



Monte Carlo Tree Search: From Playing Go to Feature Selection

Michèle Sebag

joint work: Olivier Teytaud, Sylvain Gelly, Philippe Rolet,
Romaric Gaudel

TAO, Univ. Paris-Sud

Planning to Learn, ECAI 2010, Aug. 17th





Machine Learning as Optimizing under Uncertainty

Criterion to be optimized

- Generalization Error of the learned hypothesis

Search Space

- Example set

Active learning

Rolet et al., ECML 2009

- Feature set

Feature selection

Gaudel et al., ICML 2010



Machine Learning as a Game

Similarities

- Goal : Find a sequence of moves (features, examples)

Difference : no opponent

- Optimize an expectation \equiv opponent is uniform

\Rightarrow A single-player game



Overview

- 1 Playing Go with Monte-Carlo Tree Search
- 2 Feature Selection as a Game
- 3 MCTS and Upper Confidence Tree
- 4 Extending UCT to Feature Selection : FUSE
- 5 Experimental validation



Overview

- 1 Playing Go with Monte-Carlo Tree Search
- 2 Feature Selection as a Game
- 3 MCTS and Upper Confidence Tree
- 4 Extending UCT to Feature Selection : FUSE
- 5 Experimental validation



Go as AI Challenge

Features

- Number of games $2 \cdot 10^{170} \sim$ number of atoms in universe.
- Branching factor : 200 (~ 30 for chess)
- Assessing a game ?
- Local and global features (symmetries, freedom, ...)



Gelly Silver 2007

Principles of MoGo

- A weak but unbiased assessment function : Monte Carlo-based
- Allowing the machine to play against itself and build its own strategy



Weak unbiased assessment

Monte-Carlo-based

Brügman (1993)

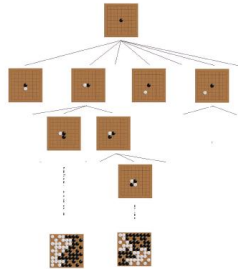
- 1 While possible, add a stone (white, black)
- 2 Compute Win(black)
- 3 Average on 1-2



Remark : The point is to be unbiased if there exists situations where you (wrongly) think you're in good shape then you go there and you're in bad shape...



Build a strategy : Monte-Carlo Tree Search



In a given situation :

Select a move

Multi-Armed Bandit

In the end :

- 1 Assess the final move
- 2 Update reward for all moves

Monte-Carlo



Select a move

Exploration vs Exploitation Dilemma



Multi-Armed Bandits

Lai, Robbins 1985

- In a casino, one wants to maximize one's gains *while playing*
- Play the best arms so far? **Exploitation**
- But there might exist better arms... **Exploration**



Multi-Armed Bandits, foll'd

Auer et al. 2001, 2002 ; Kocsis Szepesvari 2006

For each arm (move)

- Reward : Bernoulli variable $\sim \mu_i, 0 \leq \mu_i \leq 1$
- Empirical estimate : $\hat{\mu}_i \pm \text{Confidence}(n_i)$ *nb trials*

Decision : Optimism in front of unknown !

$$\text{Select } i^* = \operatorname{argmax} \hat{\mu}_i + C \sqrt{\frac{\log(\sum n_j)}{n_i}}$$

Variants

- Take into account standard deviation of $\hat{\mu}_i$
- Trade-off controlled by C
- Progressive widening



Multi-Armed Bandits, foll'd

Auer et al. 2001, 2002 ; Kocsis Szepesvari 2006

For each arm (move)

- Reward : Bernoulli variable $\sim \mu_i, 0 \leq \mu_i \leq 1$
- Empirical estimate : $\hat{\mu}_i \pm \text{Confidence}(n_i)$ *nb trials*

Decision : Optimism in front of unknown !

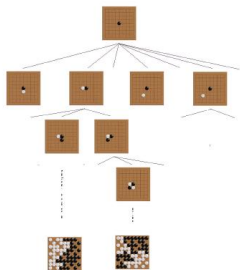
$$\text{Select } i^* = \operatorname{argmax}_i \hat{\mu}_i + C \sqrt{\frac{\log(\sum n_j)}{n_i}}$$

Variants

- Take into account standard deviation of $\hat{\mu}_i$
- Trade-off controlled by C
- Progressive widening



Monte-Carlo Tree Search



Comments : MCTS grows an asymmetrical tree

- Most promising branches are more explored
- thus their assessment becomes more precise
- Needs heuristics to deal with many arms...
- Share information among branches

MoGo : World champion in 2006, 2007, 2009

First to win over a 7th Dan player in 19×19



Overview

- 1 Playing Go with Monte-Carlo Tree Search
- 2 Feature Selection as a Game
- 3 MCTS and Upper Confidence Tree
- 4 Extending UCT to Feature Selection : FUSE
- 5 Experimental validation



Feature Selection

Optimization problem

Find $F^* = \operatorname{argmin} \mathbf{Err}(\mathcal{A}, F, \mathcal{E})$

\mathcal{F} : Set of features

F : Feature subset

\mathcal{E} : Training data set

\mathcal{A} : Machine Learning algorithm

\mathbf{Err} : Generalization error

Feature Selection Goals

- Reduced Generalization Error
- More cost-effective models
- More understandable models

Bottlenecks

- Combinatorial optimization problem : find $F \subseteq \mathcal{F}$
- Generalization error unknown



Related work

- Filter approaches [1]
 - No account for feature interdependencies
- Wrapper approaches
 - Tackling combinatorial optimization [2,3,4]
 - Exploration vs Exploitation dilemma
- Embedded approaches
 - Using the learned hypothesis [5,6]
 - Using a regularization term [7,8]
 - Restricted to linear models [7] or linear combinations of kernels [8]

[1] K. Kira, and L. A. Rendell ML'92

[2] D. Margaritis NIPS'09

[3] T. Zhang NIPS'08

[4] M. Boullé J. Mach. Learn. Res. 07

[5] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik Mach. Learn. 2002

[6] J. Rogers, and S. R. Gunn SLSFS'05

[7] R. Tibshirani Journal of the Royal Statistical Society 94

[8] F. Bach NIPS'08



FS as A Markov Decision Process

Set of features \mathcal{F}

Set of states $\mathcal{S} = 2^{\mathcal{F}}$

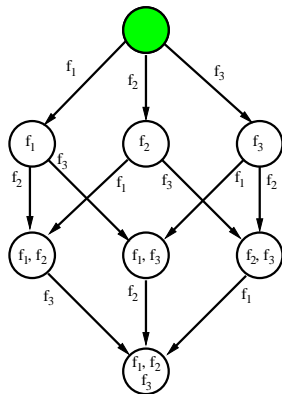
Initial state \emptyset

Set of actions $A = \{\text{add } f, f \in \mathcal{F}\}$

Final state any state

Reward function $V : \mathcal{S} \mapsto [0, 1]$

Goal : Find $\underset{F \subseteq \mathcal{F}}{\text{argmin}} \mathbf{Err}(\mathcal{A}(F, D))$





FS as a game

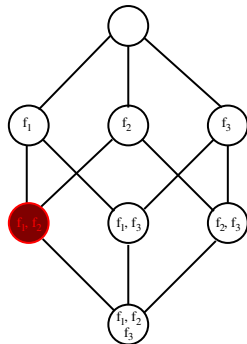
Exploration vs Exploitation tradeoff

- Virtually explore the whole lattice
- Gradually focus the search on most promising F_s
- Use a frugal, unbiased assessment of F

How ?

- Upper Confidence Tree (UCT) [1]
 - $UCT \subset Monte\text{-}Carlo\ Tree\ Search$
 - UCT tackles tree-structured optimization problems

[1] L. Kocsis, and C. Szepesvári ECML'06





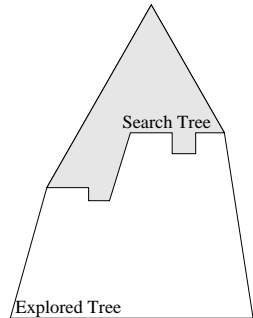
Overview

- 1 Playing Go with Monte-Carlo Tree Search
- 2 Feature Selection as a Game
- 3 MCTS and Upper Confidence Tree**
- 4 Extending UCT to Feature Selection : FUSE
- 5 Experimental validation



The UCT scheme

- Upper Confidence Tree (UCT) [1]
 - Gradually grow the search tree
 - Building Blocks
 - Select next action (bandit-based phase)
 - Add a node (leaf of the search tree)
 - Select next action bis (random phase)
 - Compute instant reward
 - Update information in visited nodes
 - Returned solution :
 - Path visited most often

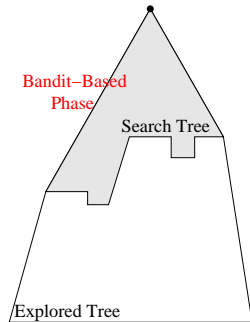


[1] L. Kocsis, and C. Szepesvári ECML'06



The UCT scheme

- Upper Confidence Tree (UCT) [1]
 - Gradually grow the search tree
 - Building Blocks
 - Select next action (bandit-based phase)
 - Add a node (leaf of the search tree)
 - Select next action bis (random phase)
 - Compute instant reward
 - Update information in visited nodes
 - Returned solution :
 - Path visited most often

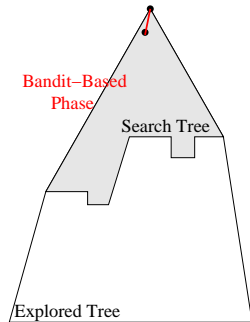


[1] L. Kocsis, and C. Szepesvári ECML'06



The UCT scheme

- Upper Confidence Tree (UCT) [1]
 - Gradually grow the search tree
 - Building Blocks
 - Select next action (bandit-based phase)
 - Add a node (leaf of the search tree)
 - Select next action bis (random phase)
 - Compute instant reward
 - Update information in visited nodes
 - Returned solution :
 - Path visited most often

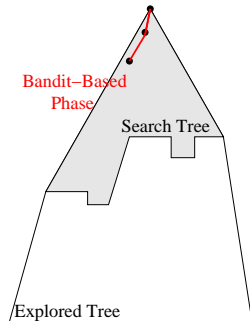


[1] L. Kocsis, and C. Szepesvári ECML'06



The UCT scheme

- Upper Confidence Tree (UCT) [1]
 - Gradually grow the search tree
 - Building Blocks
 - Select next action (bandit-based phase)
 - Add a node (leaf of the search tree)
 - Select next action bis (random phase)
 - Compute instant reward
 - Update information in visited nodes
 - Returned solution :
 - Path visited most often

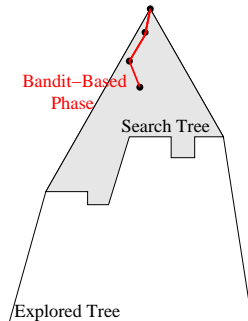


[1] L. Kocsis, and C. Szepesvári ECML'06



The UCT scheme

- Upper Confidence Tree (UCT) [1]
 - Gradually grow the search tree
 - Building Blocks
 - Select next action (bandit-based phase)
 - Add a node (leaf of the search tree)
 - Select next action bis (random phase)
 - Compute instant reward
 - Update information in visited nodes
 - Returned solution :
 - Path visited most often

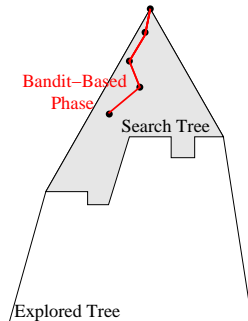


[1] L. Kocsis, and C. Szepesvári ECML'06



The UCT scheme

- Upper Confidence Tree (UCT) [1]
 - Gradually grow the search tree
 - Building Blocks
 - Select next action (bandit-based phase)
 - Add a node (leaf of the search tree)
 - Select next action bis (random phase)
 - Compute instant reward
 - Update information in visited nodes
 - Returned solution :
 - Path visited most often

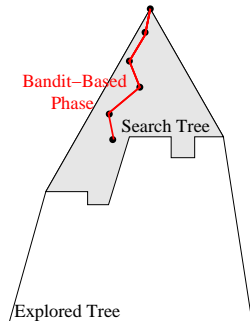


[1] L. Kocsis, and C. Szepesvári ECML'06



The UCT scheme

- Upper Confidence Tree (UCT) [1]
 - Gradually grow the search tree
 - Building Blocks
 - Select next action (bandit-based phase)
 - Add a node (leaf of the search tree)
 - Select next action bis (random phase)
 - Compute instant reward
 - Update information in visited nodes
 - Returned solution :
 - Path visited most often

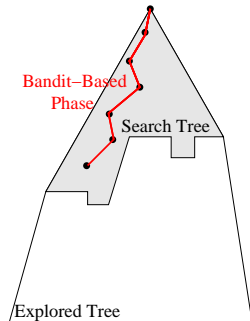


[1] L. Kocsis, and C. Szepesvári ECML'06



The UCT scheme

- Upper Confidence Tree (UCT) [1]
 - Gradually grow the search tree
 - Building Blocks
 - Select next action (bandit-based phase)
 - Add a node (leaf of the search tree)
 - Select next action bis (random phase)
 - Compute instant reward
 - Update information in visited nodes
 - Returned solution :
 - Path visited most often

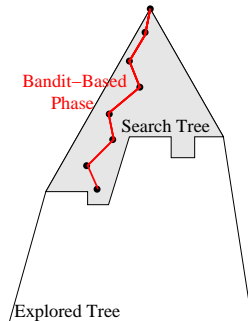


[1] L. Kocsis, and C. Szepesvári ECML'06



The UCT scheme

- Upper Confidence Tree (UCT) [1]
 - Gradually grow the search tree
 - Building Blocks
 - Select next action (bandit-based phase)
 - Add a node (leaf of the search tree)
 - Select next action bis (random phase)
 - Compute instant reward
 - Update information in visited nodes
 - Returned solution :
 - Path visited most often

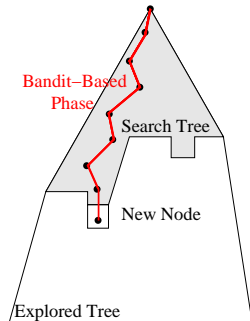


[1] L. Kocsis, and C. Szepesvári ECML'06



The UCT scheme

- Upper Confidence Tree (UCT) [1]
 - Gradually grow the search tree
 - Building Blocks
 - Select next action (bandit-based phase)
 - Add a node (leaf of the search tree)
 - Select next action bis (random phase)
 - Compute instant reward
 - Update information in visited nodes
 - Returned solution :
 - Path visited most often

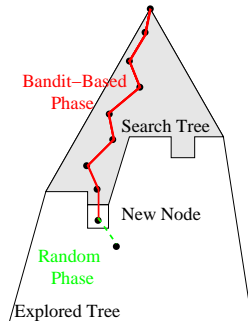


[1] L. Kocsis, and C. Szepesvári ECML'06



The UCT scheme

- Upper Confidence Tree (UCT) [1]
 - Gradually grow the search tree
 - Building Blocks
 - Select next action (bandit-based phase)
 - Add a node (leaf of the search tree)
 - Select next action bis (random phase)
 - Compute instant reward
 - Update information in visited nodes
 - Returned solution :
 - Path visited most often

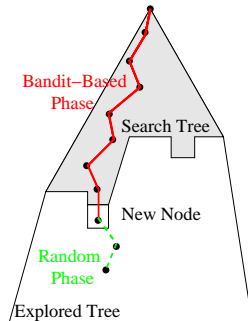


[1] L. Kocsis, and C. Szepesvári ECML'06



The UCT scheme

- Upper Confidence Tree (UCT) [1]
 - Gradually grow the search tree
 - Building Blocks
 - Select next action (bandit-based phase)
 - Add a node (leaf of the search tree)
 - Select next action bis (random phase)
 - Compute instant reward
 - Update information in visited nodes
 - Returned solution :
 - Path visited most often

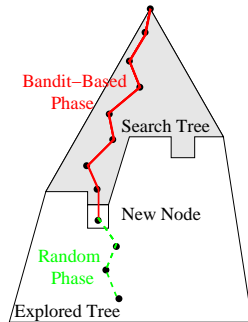


[1] L. Kocsis, and C. Szepesvári ECML'06



The UCT scheme

- Upper Confidence Tree (UCT) [1]
 - Gradually grow the search tree
 - Building Blocks
 - Select next action (bandit-based phase)
 - Add a node (leaf of the search tree)
 - Select next action bis (random phase)
 - Compute instant reward
 - Update information in visited nodes
 - Returned solution :
 - Path visited most often

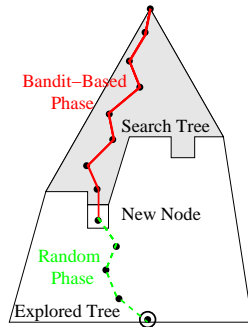


[1] L. Kocsis, and C. Szepesvári ECML'06



The UCT scheme

- Upper Confidence Tree (UCT) [1]
 - Gradually grow the search tree
 - Building Blocks
 - Select next action (bandit-based phase)
 - Add a node (leaf of the search tree)
 - Select next action bis (random phase)
 - Compute instant reward
 - Update information in visited nodes
 - Returned solution :
 - Path visited most often



[1] L. Kocsis, and C. Szepesvári ECML'06

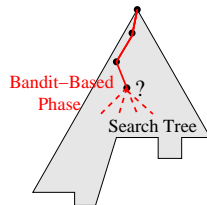


Multi-Arm Bandit-based phase

- Upper Confidence Bound (UCB1-tuned) [1]

- Select $\underset{a \in A}{\operatorname{argmax}} \hat{\mu}_a + \sqrt{\frac{c_e \log(T)}{n_a} \min\left(\frac{1}{4}, \hat{\sigma}_a^2 + \sqrt{\frac{c_e \log(T)}{t_a}}\right)}$

- T : Total number of trials in current node
- n_a : Number of trials for action a
- $\hat{\mu}_a$: Empirical average reward for action a
- $\hat{\sigma}_a^2$: Empirical variance of reward for action a





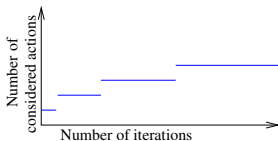
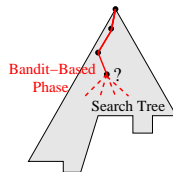
- 1 Playing Go with Monte-Carlo Tree Search
- 2 Feature Selection as a Game
- 3 MCTS and Upper Confidence Tree
- 4 Extending UCT to Feature Selection : FUSE**
- 5 Experimental validation



FUSE : bandit-based phase

The many arms problem

- Bottleneck
 - A many-armed problem (hundreds of features)
 - ⇒ need to guide UCT
- How to control the number of arms ?
 - Continuous heuristics [1]
 - Use a small exploration constant c_e
 - Discrete heuristics [2,3] : Progressive Widening
 - Consider only $\lfloor T^b \rfloor$ actions ($b < 1$)



- [1] S. Gelly, and D. Silver ICML'07
[2] R. Coulom Computer and Games 2006
[3] P. Rolet, M. Sebag, and O. Teytaud ECML'09

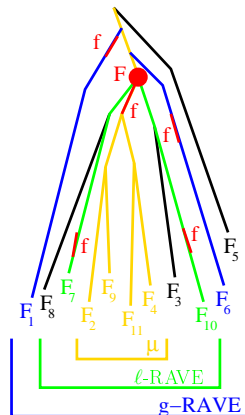


FUSE : bandit-based phase

Sharing information among nodes

- How to share information among nodes?
 - Rapid Action Value Estimation (RAVE)
[1]

$\text{RAVE}(f) = \text{average reward when } f \in F$



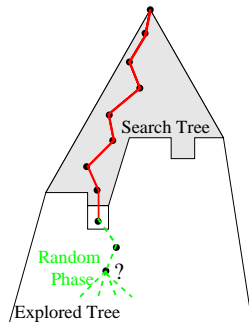
[1] S. Gelly, and D. Silver ICML'07



FUSE : random phase

Dealing with an unknown horizon

- Bottleneck
 - Finite unknown horizon
- Random phase policy
 - ↳ With probability $1 - q^{|F|}$ stop
 - | Else ● add a uniformly selected feature
 - | ● $|F| = |F| + 1$
 - | Iterate





FUSE : reward(F)

Generalization error estimate

- Requisite
 - fast (to be computed 10^4 times)
 - unbiased
- Proposed reward
 - k -NN like
 - + AUC criterion *
- Complexity : $\tilde{O}(mnd)$
 - d Number of selected features
 - n Size of the training set
 - m Size of sub-sample ($m \ll n$)



(*) Mann Whitney Wilcoxon test :

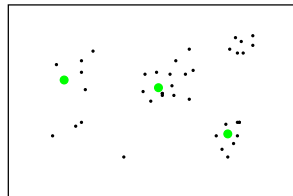
$$V(F) = \frac{|\{(x,y),(x',y')\} \in \mathcal{V}^2, \mathcal{N}_{F,k}(x) < \mathcal{N}_{F,k}(x'), y < y'\}|}{|\{(x,y),(x',y')\} \in \mathcal{V}^2, y < y'\}|}$$



FUSE : reward(F)

Generalization error estimate

- Requisite
 - fast (to be computed 10^4 times)
 - unbiased
- Proposed reward
 - k -NN like
 - + AUC criterion *
- Complexity : $\tilde{O}(mnd)$
 - d Number of selected features
 - n Size of the training set
 - m Size of sub-sample ($m \ll n$)



(*) Mann Whiton Wilcoxon test :

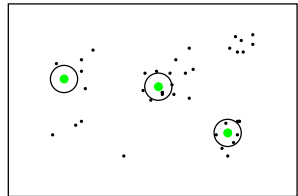
$$V(F) = \frac{|\{(x,y),(x',y') \in \mathcal{V}^2, \mathcal{N}_{F,k}(x) < \mathcal{N}_{F,k}(x'), y < y'\}|}{|\{(x,y),(x',y') \in \mathcal{V}^2, y < y'\}|}$$



FUSE : reward(F)

Generalization error estimate

- Requisite
 - fast (to be computed 10^4 times)
 - unbiased
- Proposed reward
 - k -NN like
 - + AUC criterion *
- Complexity : $\tilde{O}(mnd)$
 - d Number of selected features
 - n Size of the training set
 - m Size of sub-sample ($m \ll n$)



(*) Mann Whitney Wilcoxon test :

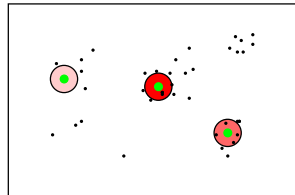
$$V(F) = \frac{|\{(x,y),(x',y') \in \mathcal{V}^2, \mathcal{N}_{F,k}(x) < \mathcal{N}_{F,k}(x'), y < y'\}|}{|\{(x,y),(x',y') \in \mathcal{V}^2, y < y'\}|}$$



FUSE : reward(F)

Generalization error estimate

- Requisite
 - fast (to be computed 10^4 times)
 - unbiased
- Proposed reward
 - k -NN like
 - + AUC criterion *
- Complexity : $\tilde{O}(mnd)$
 - d Number of selected features
 - n Size of the training set
 - m Size of sub-sample ($m \ll n$)



(*) Mann Whitney Wilcoxon test :

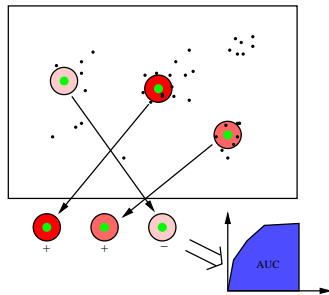
$$V(F) = \frac{|\{(x,y),(x',y') \in \mathcal{V}^2, \mathcal{N}_{F,k}(x) < \mathcal{N}_{F,k}(x'), y < y'\}|}{|\{(x,y),(x',y') \in \mathcal{V}^2, y < y'\}|}$$



FUSE : reward(F)

Generalization error estimate

- Requisite
 - fast (to be computed 10^4 times)
 - unbiased
- Proposed reward
 - k -NN like
 - + AUC criterion *
- Complexity : $\tilde{O}(mnd)$
 - d Number of selected features
 - n Size of the training set
 - m Size of sub-sample ($m \ll n$)



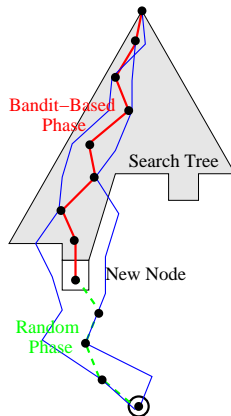
(*) Mann Whitney Wilcoxon test :

$$V(F) = \frac{|\{(x,y),(x',y')\} \in \mathcal{V}^2, \mathcal{N}_{F,k}(x) < \mathcal{N}_{F,k}(x'), y < y'\}|}{|\{(x,y),(x',y')\} \in \mathcal{V}^2, y < y'\}|}$$



FUSE : update

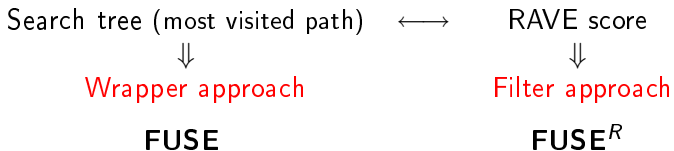
- Explore a graph
 - ⇒ Several paths to the same node
- Update only current path





The FUSE algorithm

- N iterations :
each iteration i) follows a path ; ii) evaluates a final node
- Result :



- On the feature subset, use end learner \mathcal{A}
 - Any Machine Learning algorithm
 - Support Vector Machine with Gaussian kernel in experiments



Overview

- 1 Playing Go with Monte-Carlo Tree Search
- 2 Feature Selection as a Game
- 3 MCTS and Upper Confidence Tree
- 4 Extending UCT to Feature Selection : FUSE
- 5 Experimental validation



Experimental setting

- Questions
 - FUSE vs FUSE^R
 - Continuous vs discrete exploration heuristics
 - FS performance w.r.t. complexity of the target concept
 - Convergence speed
- Experiments on

Data set	Samples	Features	Properties
Madelon [1]	2,600	500	XOR-like
Arcene [1]	200	10,000	Redundant features
Colon	62	2,000	“Easy”



Experimental setting

- Baselines
 - CFS (Constraint-based Feature Selection) [1]
 - Random Forest [2]
 - Lasso [3]
 - RAND^R : RAVE obtained by selecting 20 random features at each iteration
- Results averaged on 50 splits (10×5 fold cross-validation)
- End learner
 - Hyper-parameters optimized by 5 fold cross-validation

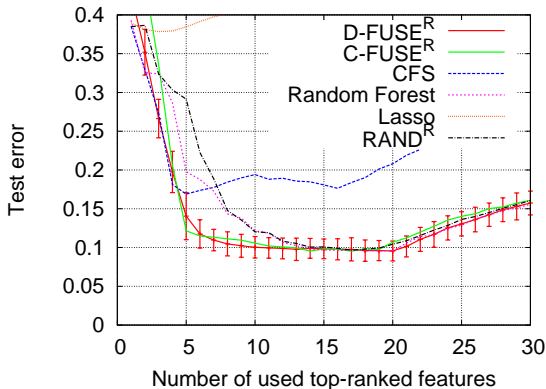
[1] M. A. Hall ICML'00

[2] J. Rogers, and S. R. Gunn SLSFS'05

[3] R. Tibshirani Journal of the Royal Statistical Society 94



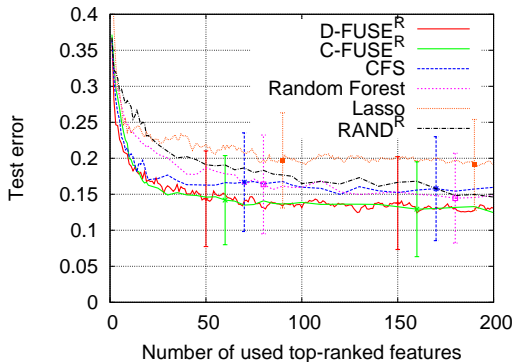
Results on Madelon after 200,000 iterations



- **Remark** : FUSE^R = best of both worlds
 - Removes redundancy (like CFS)
 - Keeps conditionally relevant features (like Random Forest)



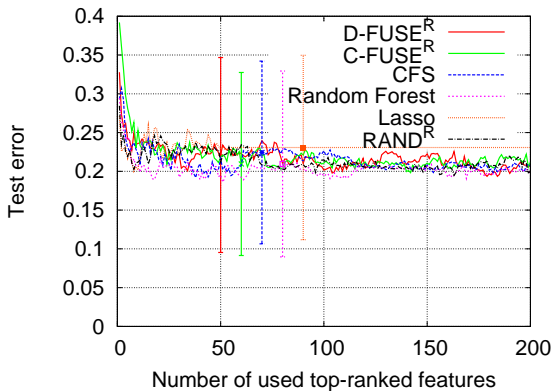
Results on Arcene after 200,000 iterations



- Remark : FUSE^R = best of both worlds
 - Removes redundancy (like CFS)
 - Keeps conditionally relevant features (like Random Forest)



Results on Colon after 200,000 iterations



- Remark

- All equivalent



NIPS 2003 Feature Selection challenge

- Test error on a disjoint test set

database	algorithm	challenge error	submitted features	irrelevant features
Madelon	FSPP2 [1]	6.22% (1 st)	12	0
	D-FUSE ^R	6.50% (24 th)	18	0
	Bayes-nn-red [2]	7.20% (1 st)	100	0
Arcene	D-FUSE ^R (on all)	8.42% (3 rd)	500	34
	D-FUSE ^R	9.42% 500 (8 th)	500	0

[1] K. Q. Shen, C. J. Ong, X. P. Li, E. P. V. Wilder-Smith Mach. Learn. 2008

[2] R. M. Neal, and J. Zhang Feature extraction, foundations and applications, Springer 2006



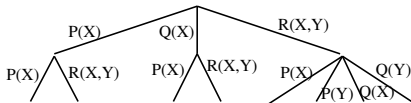
Contributions

- Formalization of Feature Selection as a Markov Decision Process
- Efficient approximation of the optimal policy (based on UCT)
 - ⇒ Any-time algorithm
- Experimental results
 - State of the art
 - High computational cost (45 minutes on Madelon)



Perspectives

- Other end learners
- Revisit the reward see (Hand 2010) about AUC
- Extend to Feature construction along [1]



[1] F. de Mesmay, A. Rimmel, Y. Voronenko, and M. Püschel | ICML'09