Order parameters and model selection in Machine Learning: model characterization and feature selection

Romaric Gaudel

Advisor: Michèle Sebag; Co-advisor: Antoine Cornuéjols

PhD, December 14, 2010

Supervised Machine Learning

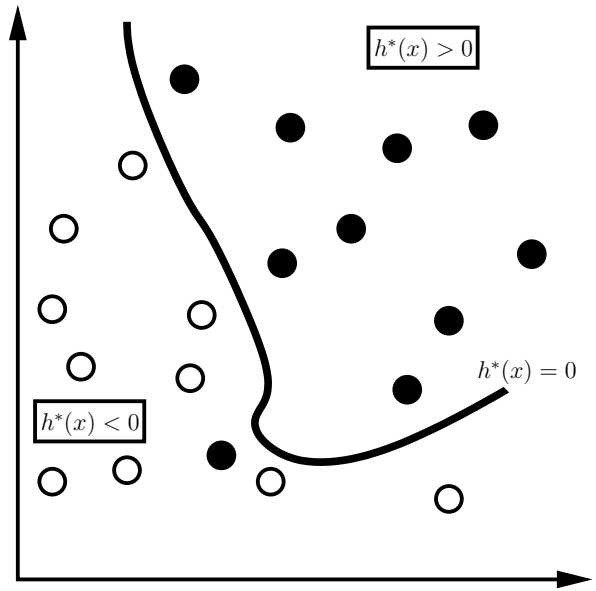
- Background
  - Unknown distribution $\mathbf{P}(x, y)$ on $\mathcal{X} \times \mathcal{Y}$
- Objective
  - Find h^* minimizing generalization error

$$\mathbf{Err}\,(h) = \mathbf{E}_{\mathbf{P}(x,y)}\left[\ell\,(h(x), y)\right]$$

    - Where $\ell\,(h(x), y)$ is the cost of error on example x
- Given
  - Training examples

$$\mathcal{L} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

    - Where $(x_i, y_i) \sim \mathbf{P}(x, y),\ i \in 1, \ldots, n$



Supervised Machine Learning 2

(Vapnik-Chervonenkis; Bottou & Bousquet, 08)

- Approximation error (a.k.a. **bias**)
  - Learned hypothesis belong to $\mathcal{H}$

$$h^*_{\mathcal{H}} = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \, \mathbf{Err}\,(h)$$

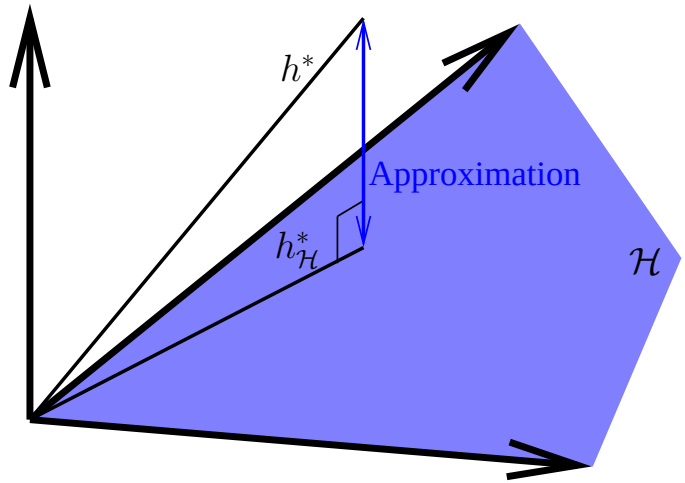
- Estimation error (a.k.a. **variance**)
  - **Err** estimated by empirical error

$$\mathbf{Err}_n\,(h) = \frac{1}{n} \sum \ell(h(x_i), y_i)$$

$$h_n = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \, \mathbf{Err}_n\,(h)$$

- Optimization error
  - Learned hypothesis returned by an optimization algorithm $\mathcal{A}$

$$\hat{h}_n = \mathcal{A}(\mathcal{L})$$



Supervised Machine Learning 2

(Vapnik-Chervonenkis; Bottou & Bousquet, 08)

- Approximation error (a.k.a. **bias**)
  - Learned hypothesis belong to $\mathcal{H}$

$$h^*_{\mathcal{H}} = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \, \mathbf{Err}\,(h)$$

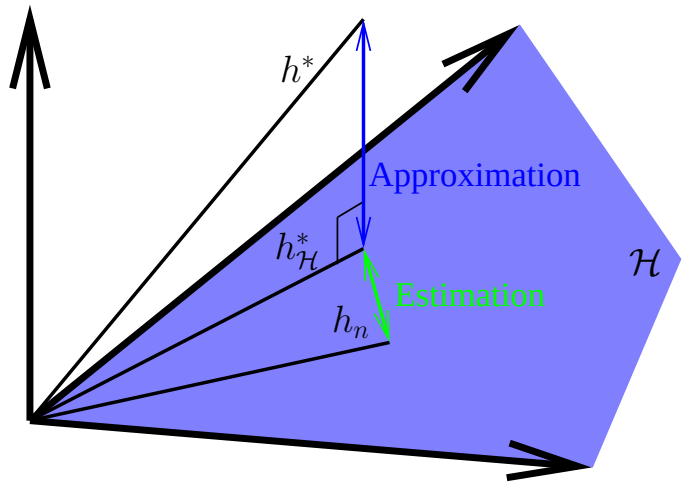
- Estimation error (a.k.a. **variance**)
  - **Err** estimated by empirical error

$$\mathbf{Err}_n\,(h) = \frac{1}{n} \sum \ell(h(x_i), y_i)$$

$$h_n = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \, \mathbf{Err}_n\,(h)$$

- Optimization error
  - Learned hypothesis returned by an optimization algorithm $\mathcal{A}$

$$\hat{h}_n = \mathcal{A}(\mathcal{L})$$



Supervised Machine Learning 2

(Vapnik-Chervonenkis; Bottou & Bousquet, 08)

- Approximation error (a.k.a. **bias**)
  - Learned hypothesis belong to $\mathcal{H}$

$$h^*_{\mathcal{H}} = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \, \mathbf{Err}\,(h)$$

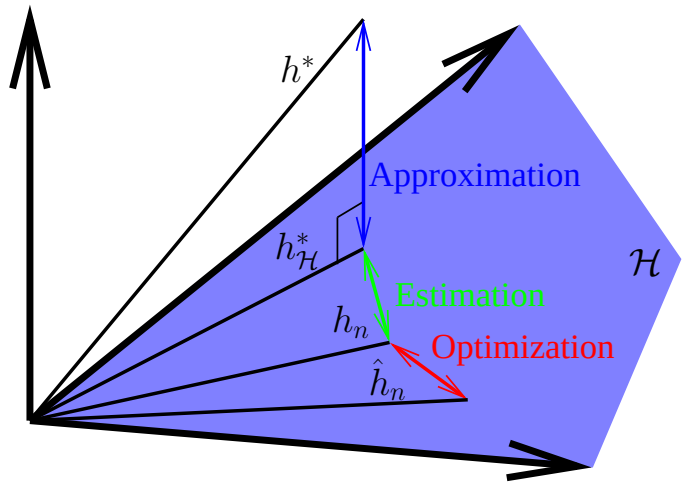
- Estimation error (a.k.a. **variance**)
  - **Err** estimated by empirical error

$$\mathbf{Err}_n\,(h) = \frac{1}{n} \sum \ell(h(x_i), y_i)$$

$$h_n = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \, \mathbf{Err}_n\,(h)$$

- Optimization error
  - Learned hypothesis returned by an optimization algorithm $\mathcal{A}$

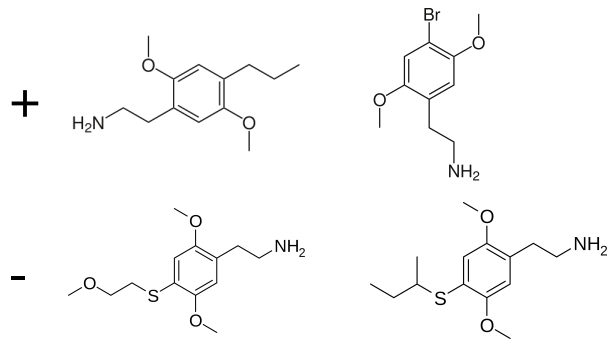
$$\hat{h}_n = \mathcal{A}(\mathcal{L})$$



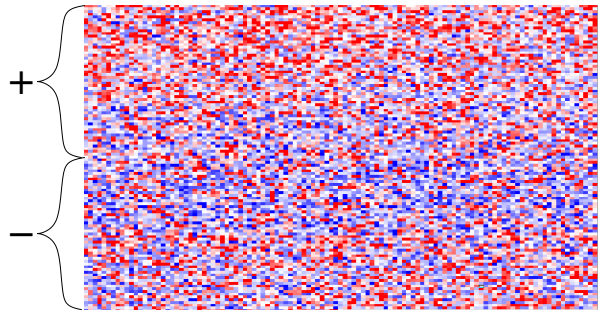
Focus of the thesis

Combinatorial optimization problems hidden in Machine Learning

- Relational representation
  - $\Longrightarrow$ Combinatorial optimization problem
    - Example: Mutagenesis database



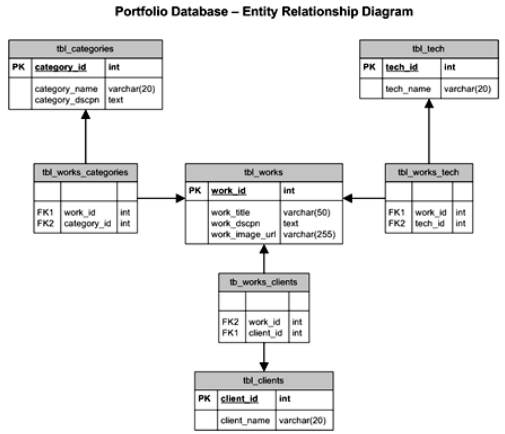
- Feature Selection
  - $\Longrightarrow$ Combinatorial optimization problem
    - Example: Microarray data



Outline

Outline

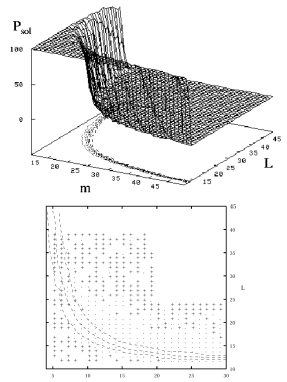
Relational Learning / Inductive Logic Programming



- Position
  - Relational database
    - $\mathcal{X}$: keys in the database
    - Background knowledge
- $\mathcal{H}$: set of logical formulas
  - Expressive language
  - Actual covering test: Constraint Satisfaction Problem (CSP)

CSP consequences within Inductive Logic Programming

- Consequences of the Phase Transition
  - Complexity
    - Worst case: NP-hard
    - Average case: “easy” except in Phase Transistion (Cheeseman et al. 91)
- Phase Transition in Inductive Logic Programming
  - Existence (Giordana & Saitta, 00)
  - Impact: fails to learn in Phase Transition region (Botta et al., 03)



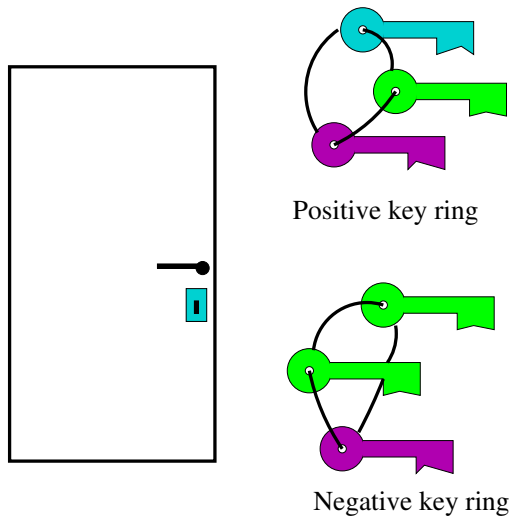
Multiple Instance Problems

The missing link between Relational and Propositional Learning

- Multiple Instance Problems (MIP) (Dietterich et al., 89)
  - An example: set of instances
  - An instance: vector of features
  - Target-concept: there exists an instance satisfying a predicate P

$$\text{pos}(x) \iff \exists I \in x,\ P(I)$$

- Example of MIP
  - A locked door
  - A positive key-ring contains a key which can unlock the door

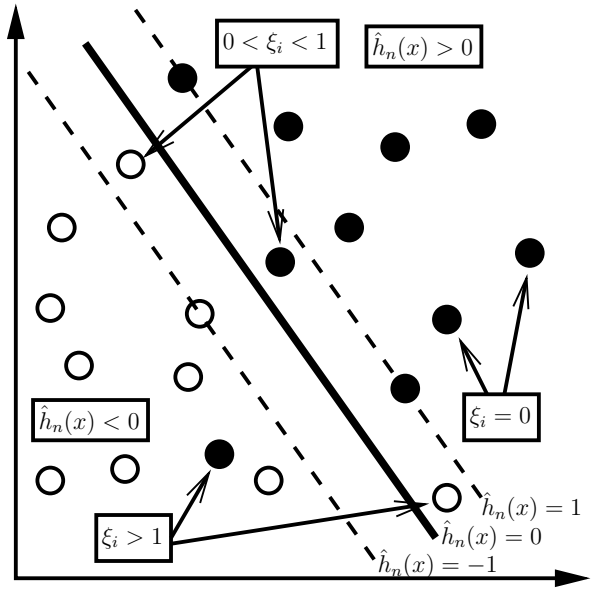


Support Vector Machine

- A Convex optimization problem

$$\underset{\alpha \in \mathbf{R}^n}{\operatorname{argmin}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

$$\text{s.t.} \begin{cases} \sum_{i=1}^{n} \alpha_i y_i = 0 \\ 0 \leqslant \alpha_i \leqslant C, \qquad i = 1, \ldots, n \end{cases}$$



- Kernel trick

$$\langle x_i, x_j \rangle \rightsquigarrow K(x_i, x_j)$$

- Kernel-based propositionalization (differs from RKHS framework)

$$\begin{cases} \mathcal{L} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \\ K \end{cases} \quad \rightsquigarrow \quad \Phi : x \longrightarrow (K(x_1, x), \ldots, K(x_n, x))$$

SVM and MIP

- Averaging-kernel for MIP (Gärtner et al., 02)
  - Given a kernel k on instances

$$K(x, x') = \frac{\sum_{x_i \in x} \sum_{x_j \in x'} k(x_i, x_j)}{\text{norm}(x)\,\text{norm}(x')}$$

Question

- MIP Target-concept: existential properties
- Averaging-Kernel: average properties

Do averaging-kernels sidestep limitations of Relational Learning?

Methodology

Inspired from Phase Transition studies

- Usual Phase Transition framework
  - Generate data after control parameters
  - Observe results
  - Draw phase diagram: results w.r.t. order parameters
- This study
  - Generalized Multiple Instance Problem
  - Experimental results of averaging-kernel-based propositionalization

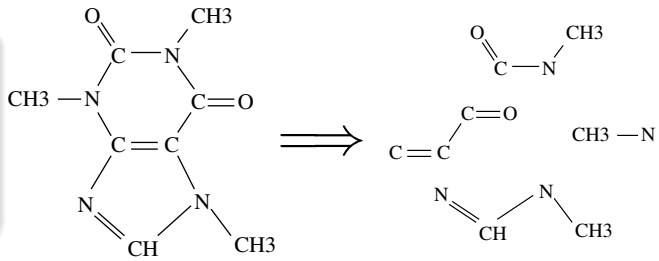
Outline

Generalized Multiple Instance Problems

- Generalized MIP (Weidmann et al., 03)
  - An example: set of instances
  - An instance: vector of features
  - Target-concept: conjunction of predicates $P_1, \ldots, P_m$

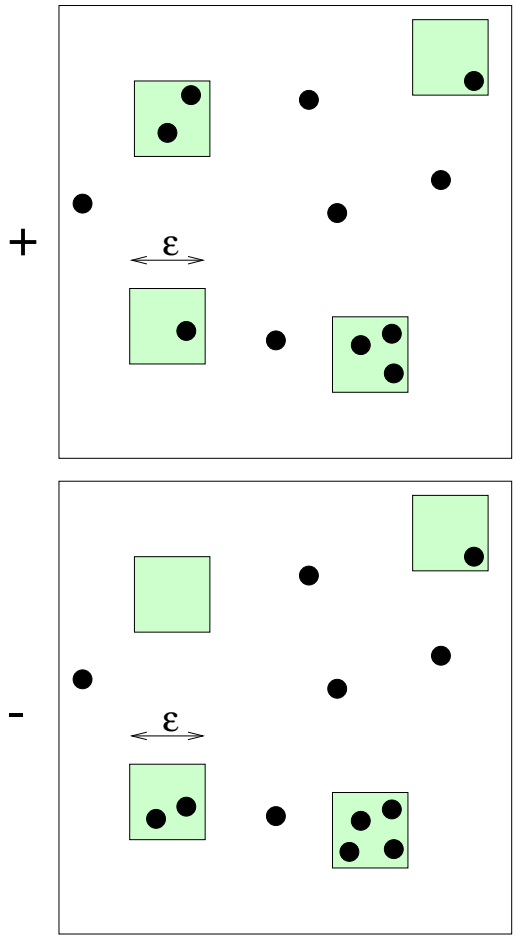
$$\text{pos}(x) \iff \exists I_1, \ldots, I_m \in x, \bigwedge_{i=1}^{m} P_i(I_i)$$

- Example of Generalized MIP
  - A molecule: set of sub-graphs
  - Bioactivity: implies several sub-graphs



Control Parameters

Category	Param.	Definition
Instances $I = (a, \boldsymbol{z})$	$\lvert\Sigma\rvert$	Size of alphabet Σ, $a \in \Sigma$
	d	number of numerical features, $\boldsymbol{z} \in [0,1]^d$
Examples	M^+	Number of instances per positive example
	M^-	Number of instances per negative example
	m^+	Number of instances in a predicate, for positive example
	m^-	Number of instances in a predicate, for negative example
	P_m	Number of predicates "missed" by each negative example
Concept	P	Number of predicate
	ε	Radius of each predicate (ε-ball)

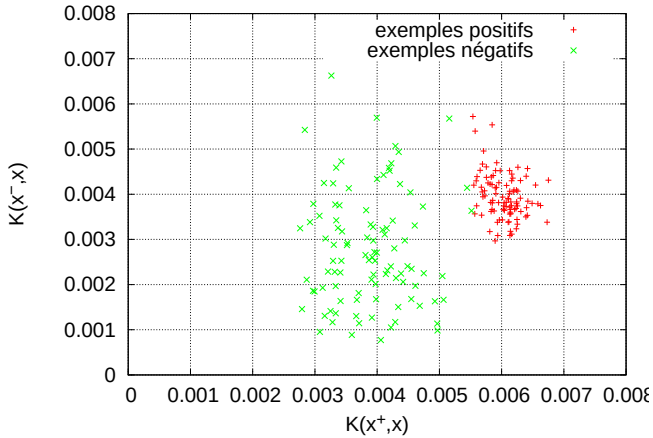


Limitation of averaging-kernels

- Theoretical analysis

Failure for $\frac{m^+}{M^+} = \frac{m^-}{M^-}$

$$\mathbf{E}_{x\sim D^+}\left[K(x_i, x)\right] = \mathbf{E}_{x\sim D^-}\left[K(x_i, x)\right]$$



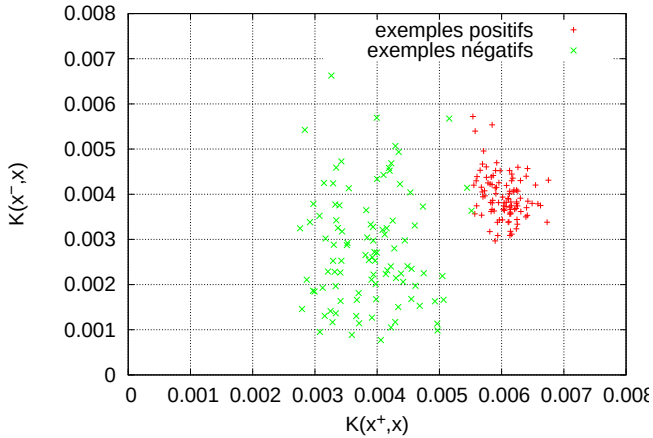
- Empirical approach
  - Generate, test and average empirical results
  - Establish a lower bound on generalization error

Limitation of averaging-kernels

- Theoretical analysis

Failure for $\frac{m^+}{M^+} = \frac{m^-}{M^-}$

$$\mathbf{E}_{x\sim D^+}\left[K(x_i, x)\right] = \mathbf{E}_{x\sim D^-}\left[K(x_i, x)\right]$$



- Empirical approach
  - Generate, test and average empirical results
  - Establish a lower bound on generalization error

Efficiency of kernel-based propositionalization

- Kernel-based propositionalization $\mathcal{H}'$ (differs from RKHS framework)

$$\begin{cases} \mathcal{L} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \\ K \end{cases} \quad \rightsquigarrow \quad \Phi : x \longrightarrow (K(x_1, x), \ldots, K(x_n, x))$$

- Question (Q): separability of test examples $\mathcal{T}$ in $\mathcal{H}'$

$\exists ?\ \alpha_i,$

$$\begin{cases} \sum_{i=1}^{n} \alpha_i y_i = 0 & & \text{SVM constraint} \\ 0 \leqslant \alpha_i \leqslant C & i = 1, \ldots, n & \text{SVM constraint} \\ \left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x') + b\right) y' \geqslant 1 & (x', y') \in \mathcal{T} & \text{test constraint} \end{cases}$$

- An optimistic criterion
  - Test examples used to define α_i

Efficiency of kernel-based propositionalization

- Kernel-based propositionalization $\mathcal{H}'$ (differs from RKHS framework)

$$\begin{cases} \mathcal{L} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \\ K \end{cases} \quad \rightsquigarrow \quad \Phi : x \longrightarrow (K(x_1, x), \ldots, K(x_n, x))$$

- Question (Q): separability of test examples $\mathcal{T}$ in $\mathcal{H}'$

$\exists ?\ \alpha_i,$

$$\begin{cases} \sum_{i=1}^{n} \alpha_i y_i = 0 & & \text{SVM constraint} \\ 0 \leqslant \alpha_i \leqslant C & i = 1, \ldots, n & \text{SVM constraint} \\ \left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x') + b\right) y' \geqslant 1 & (x', y') \in \mathcal{T} & \text{test constraint} \end{cases}$$

- An optimistic criterion
  - Test examples used to define α_i

Lower bound on the generalization error

Theorem

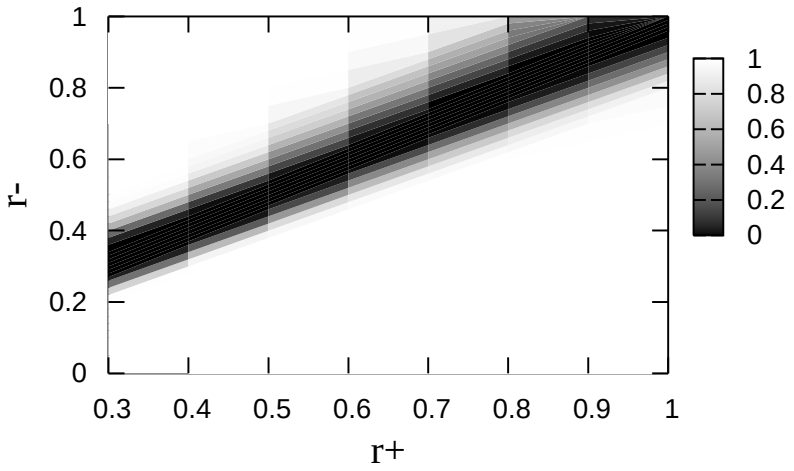
- For each setting
  - Generate T training (respectively test) datasets $\mathcal{L}$ (resp. $\mathcal{T}$)
  - Record τ: proportion of couples $(\mathcal{L}, \mathcal{T})$ s.t. (Q) is satisfiable
- Let $\mathbf{Err}_{\mathcal{L}}$ be the generalization error when learning from $\mathcal{L}$
- Then,

$$\forall \eta > 0, \text{ with probability at least } 1 - \exp(-2\eta^2 T)$$

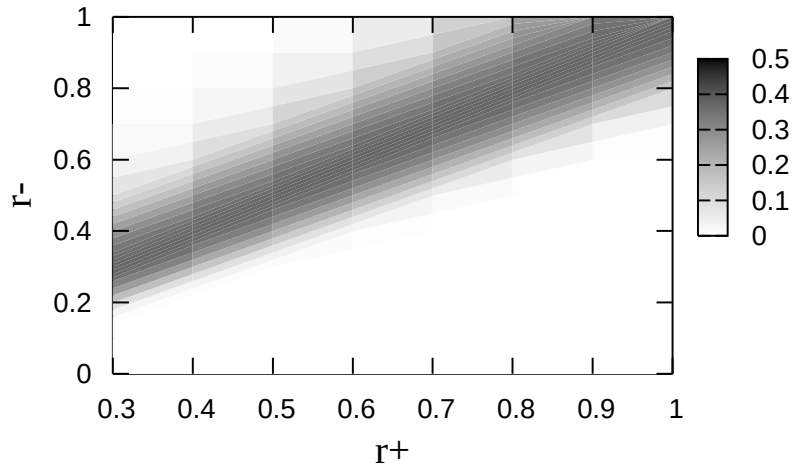
$$\mathbf{E}_{|\mathcal{L}|=n}[\mathbf{Err}_{\mathcal{L}}] > 1 - (\tau + \eta)^{\frac{1}{|\mathcal{T}|}}$$

- Remark
  - (Q) solved using Linear Programming

Empirical failure region



(Q) satisfiability



SVM test error

- Control parameters
  - Instance space: $\Sigma \times [0,1]^{30}$
  - 100 instances per example
  - 30 predicates
  - 40 couples $(\mathcal{L}, \mathcal{T})$ per setting

The averaging kernel fails when

- Small training dataset ($|\mathcal{L}| \leqslant 100$)
- $\frac{m^+}{M^+} \approx \frac{m^-}{M^-}$

Partial conclusion on Relational Kernel

Contributions

- Theoretical and empirical identification of limitations for averaging-kernels
- A lower bound on generalization error

- Perspectives
  - Failure region for other kernels
    - Claim: any kernel computable in a polynomial time leads to a failure region
    - When is the failure region small enough?

Outline

Feature Selection

- Optimization problem

$$\underset{F \subseteq \mathcal{F}}{argmin}\ \mathbf{Err}\left(\mathcal{A}\left(F, \mathcal{L}\right)\right)$$

$\mathcal{F}$: Set of features

F: Feature subset

$\mathcal{L}$: Training data set

$\mathcal{A}$: Machine Learning algorithm

Err: Generalization error

  - Feature Selection (FS)
    - Minimize the Generalization Error
    - Decrease the learning/use cost of models
    - Lead to more understandable models
  - Bottlenecks
    - Combinatorial optimization problem: find $F \subseteq \mathcal{F}$
    - Unknown objective function: generalization error

Filter approaches for Feature Selection

- Score features
- Select the best ones

Pro

- Cheap

Cons

- Cannot handle all inter-dependencies between features

- Filter approaches
  - ANOVA (Analysis of Variance)
  - RELIEFF (Kira & Rendell, 92)

Embedded approaches for Feature Selection

- Exploit the learned hypothesis
- And/Or modify the learning criterion to induce sparsity

Pro

- Based on relevance of features in the learned model

Cons

- Limited to linear models or a linear combination of kernels
- Possibly misled by feature interdependencies

- Embedded approaches
  - Lasso (Tibshirani, 94)
  - Multiple Kernel Learning (Bach, 08)
  - Gini score on Random Forest (Rogers & Gunn, 05)

Wrapper approaches for Feature Selection

- Test feature subsets
  - Actually address the combinatorial problem

Pro

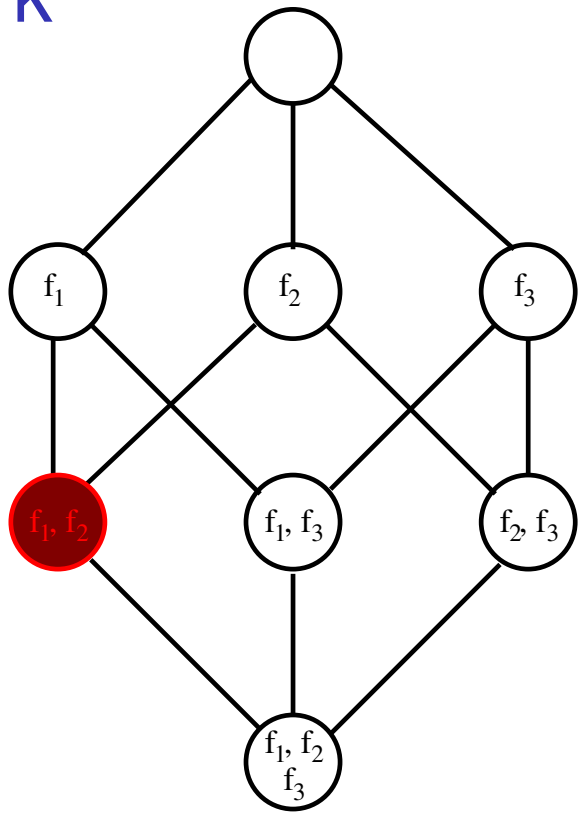
- Look for (approximate) best solution

Cons

- Computationally expensive

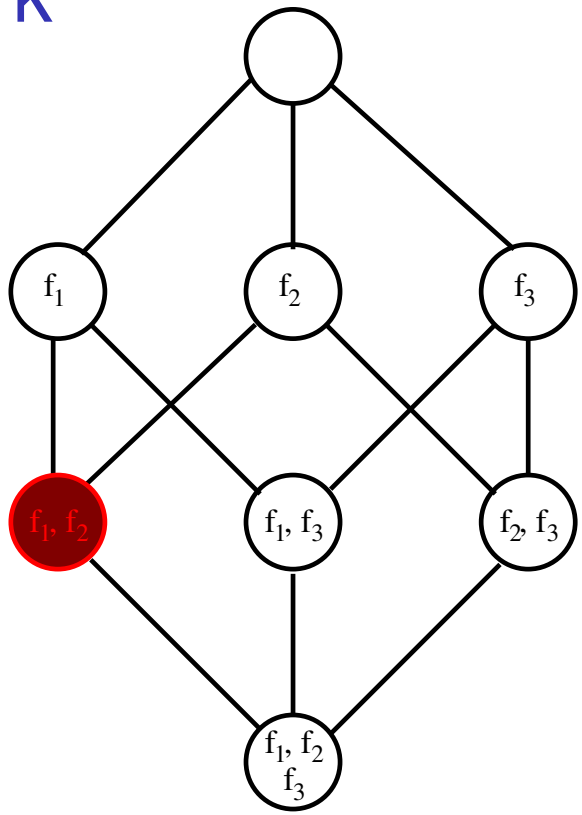
- Wrapper approaches
  - Look ahead (Margaritis, 09)
  - Mix forward/backward search (Zhang, 08)
  - Mix global/local search (Boullé, 07)

Proposed Feature Selection framework



- Goal: optimal
  - Find $\underset{F \subseteq \mathcal{F}}{argmin}$ **Err** $(\mathcal{A}(F, \mathcal{L}))$
    - Virtually explore the whole lattice
- Goal: tractable
  - Frugal, unbiased assessment of F
    - Cannot compute **Err** $(\mathcal{A}(F, \mathcal{L}))$
  - Gradually focus search on most promising subtrees
    - Exploration vs Exploitation trade-off

Proposed Feature Selection framework

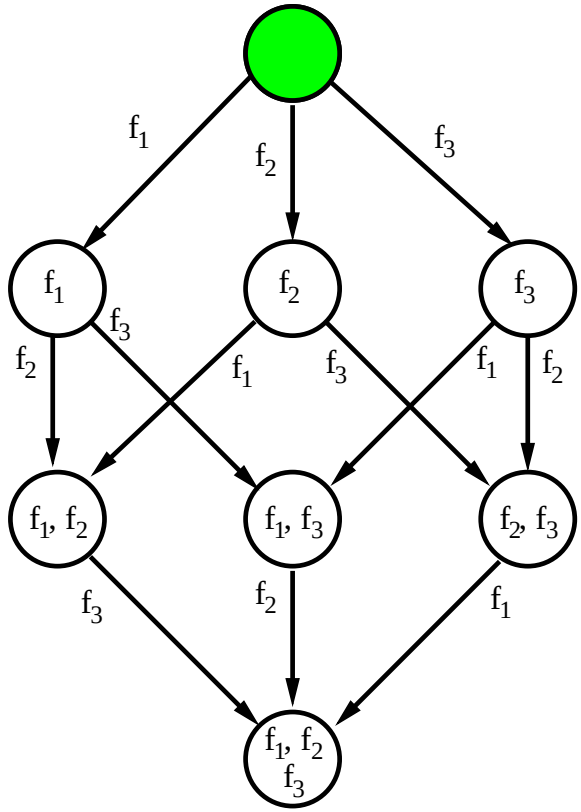


- Goal: optimal
  - Find $\underset{F \subseteq \mathcal{F}}{argmin}$ **Err** $(\mathcal{A}(F, \mathcal{L}))$
    - Virtually explore the whole lattice
- Goal: tractable
  - Frugal, unbiased assessment of F
    - Cannot compute **Err** $(\mathcal{A}(F, \mathcal{L}))$
  - Gradually focus search on most promising subtrees
    - Exploration vs Exploitation trade-off

Outline

Feature Selection as a Markov Decision Process

- From the lattice of subsets ...
  - Set of features: $\mathcal{F}$
  - Set of candidates: $2^{\mathcal{F}}$
- ... to a Markov Decision Process
  - Set of states: $\mathcal{S} = 2^{\mathcal{F}}$
  - Initial state: $\varnothing$
  - Set of actions: $A = \{\text{add } f,\ f \in \mathcal{F}\}$
  - Reward function: $V : \mathcal{S} \to [0, 1]$
    - Ideally : $V(F) = \textbf{Err}\left(\mathcal{A}\left(F, \mathcal{L}\right)\right)$
    - In practice: Fast unbiased estimate



Optimal Policy

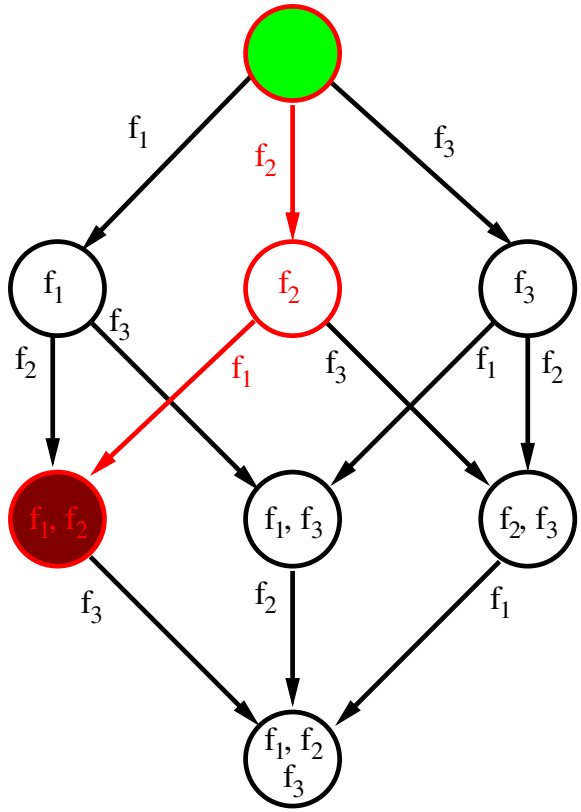
- Policy: $\pi : \mathcal{S} \rightarrow A$
- Final state following a policy: F_π
- Optimal policy: $\pi^* = \underset{\pi}{argmin}\ \mathbf{Err}\, F_\pi$

- Bellman's optimality principle

$$\pi^*(F) = \underset{f \in \mathcal{F}}{argmin}\ V^*(F \cup \{f\})$$

with

$$V^*(F) = \begin{cases} \mathbf{Err}\,(\mathbf{Err}\,(\mathcal{A}(F, \mathcal{L}))) & \text{if } final(F) \\ \underset{f \in \mathcal{F}}{min}\ V^*(F \cup \{f\}) & \text{otherwise} \end{cases}$$



Optimal Policy

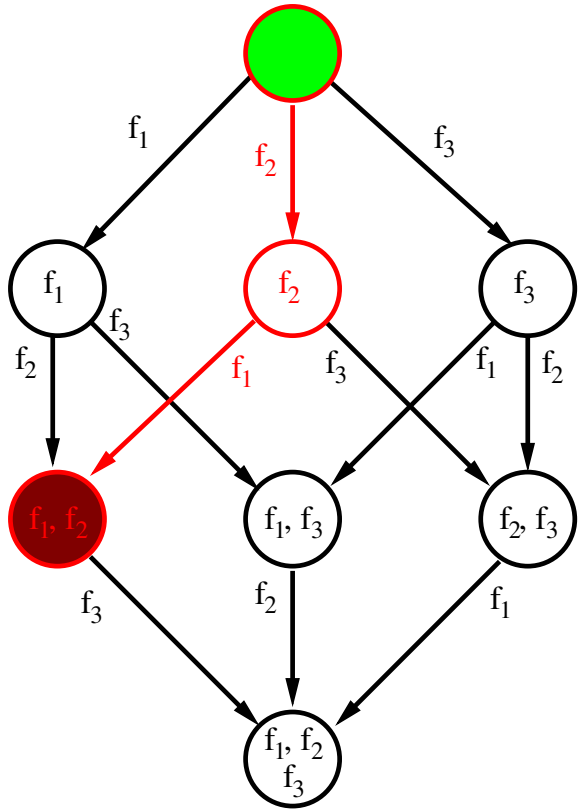
- Policy: $\pi : \mathcal{S} \rightarrow A$
- Final state following a policy: F_π
- Optimal policy: $\pi^* = \underset{\pi}{argmin}\ \mathbf{Err}\, F_\pi$

- Bellman's optimality principle

$$\pi^*(F) = \underset{f \in \mathcal{F}}{argmin}\ V^*(F \cup \{f\})$$

with

$$V^*(F) = \begin{cases} \mathbf{Err}\,(\mathbf{Err}\,(\mathcal{A}(F, \mathcal{L}))) & \text{if } final(F) \\ \underset{f \in \mathcal{F}}{min}\ V^*(F \cup \{f\}) & \text{otherwise} \end{cases}$$



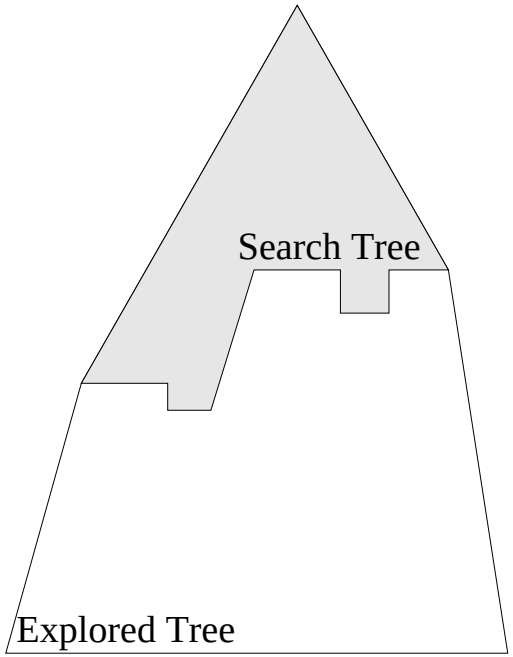
π^* intractable $\Rightarrow$ approximation using a one-player game approach

Outline

The UCT Monte-Carlo Tree Search

(Kocsis & Szepesvári, 06)

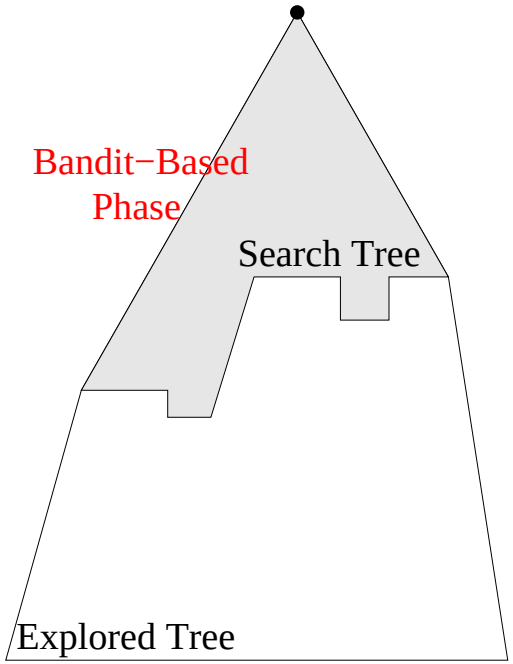
- Gradually grow a search tree
- Building Blocks
  - Select next action (bandit-based phase)
  - Add a node (leaf of the search tree)
  - Monte-Carlo exploration (random phase)
  - Compute instant reward
  - Update visited nodes
- Returned solution
  - Path visited most often



The UCT Monte-Carlo Tree Search

(Kocsis & Szepesvári, 06)

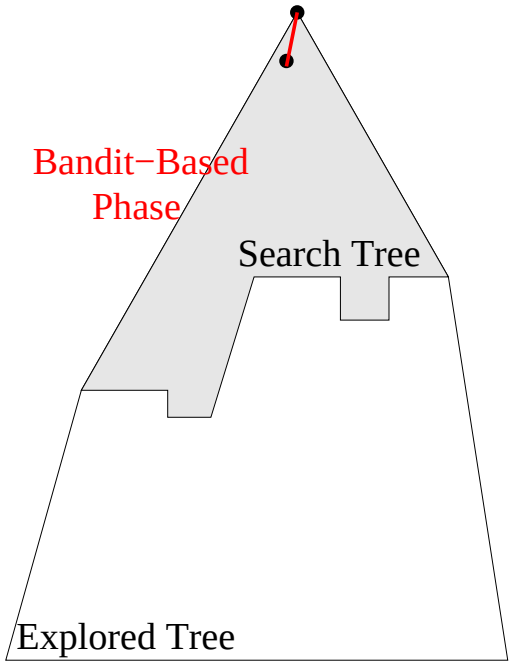
- Gradually grow a search tree
- Building Blocks
  - Select next action (bandit-based phase)
  - Add a node (leaf of the search tree)
  - Monte-Carlo exploration (random phase)
  - Compute instant reward
  - Update visited nodes
- Returned solution
  - Path visited most often



The UCT Monte-Carlo Tree Search

(Kocsis & Szepesvári, 06)

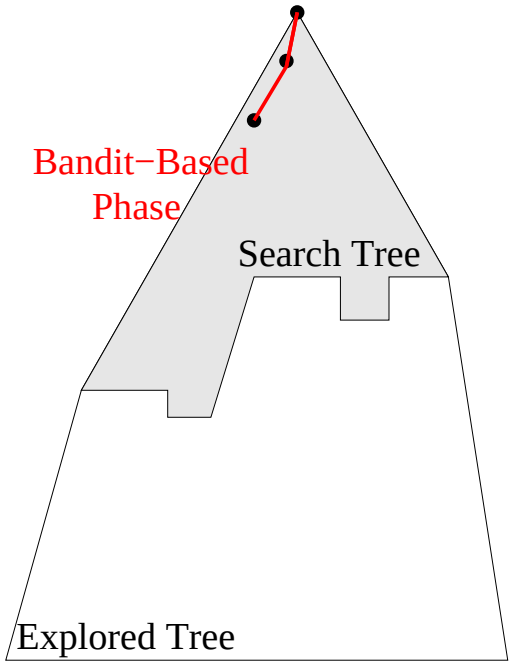
- Gradually grow a search tree
- Building Blocks
  - Select next action (bandit-based phase)
  - Add a node (leaf of the search tree)
  - Monte-Carlo exploration (random phase)
  - Compute instant reward
  - Update visited nodes
- Returned solution
  - Path visited most often



The UCT Monte-Carlo Tree Search

(Kocsis & Szepesvári, 06)

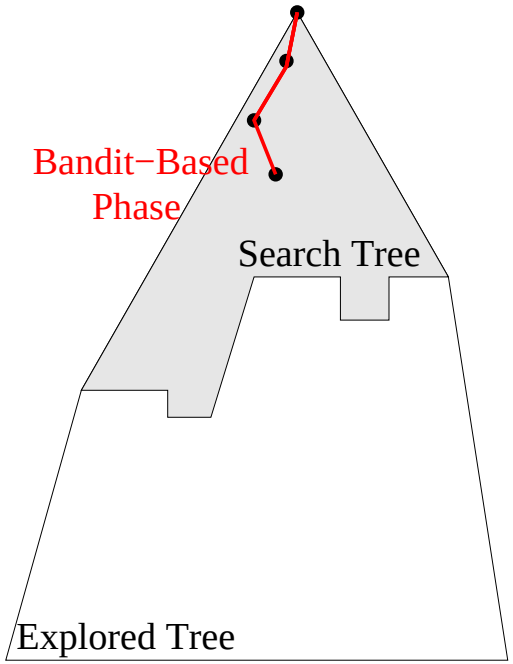
- Gradually grow a search tree
- Building Blocks
  - Select next action (bandit-based phase)
  - Add a node (leaf of the search tree)
  - Monte-Carlo exploration (random phase)
  - Compute instant reward
  - Update visited nodes
- Returned solution
  - Path visited most often



The UCT Monte-Carlo Tree Search

(Kocsis & Szepesvári, 06)

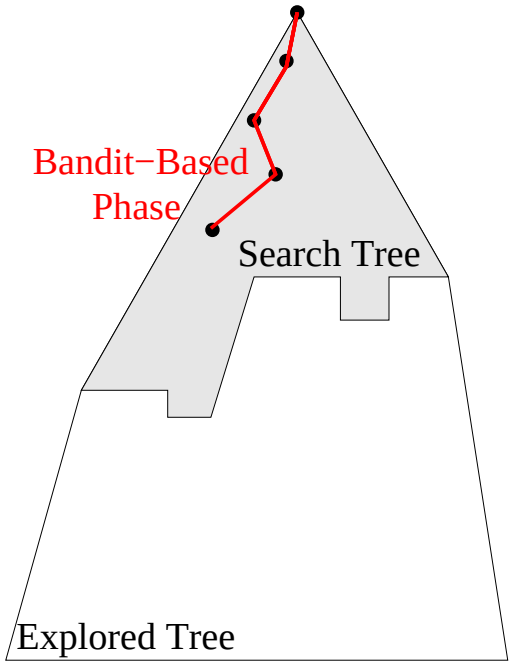
- Gradually grow a search tree
- Building Blocks
  - Select next action (bandit-based phase)
  - Add a node (leaf of the search tree)
  - Monte-Carlo exploration (random phase)
  - Compute instant reward
  - Update visited nodes
- Returned solution
  - Path visited most often



The UCT Monte-Carlo Tree Search

(Kocsis & Szepesvári, 06)

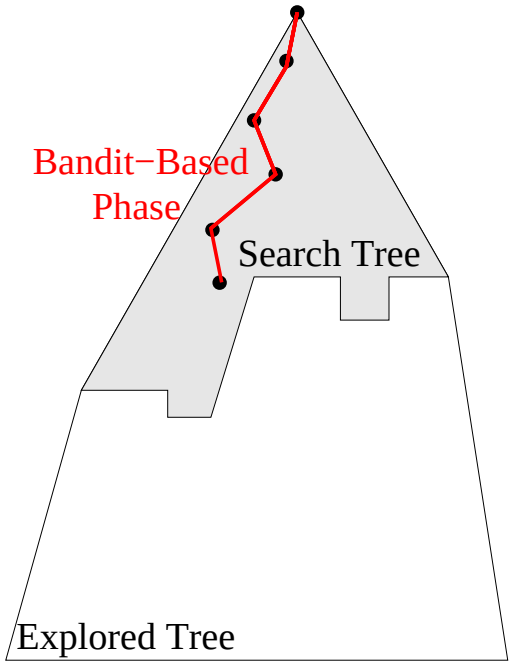
- Gradually grow a search tree
- Building Blocks
  - Select next action (bandit-based phase)
  - Add a node (leaf of the search tree)
  - Monte-Carlo exploration (random phase)
  - Compute instant reward
  - Update visited nodes
- Returned solution
  - Path visited most often



The UCT Monte-Carlo Tree Search

(Kocsis & Szepesvári, 06)

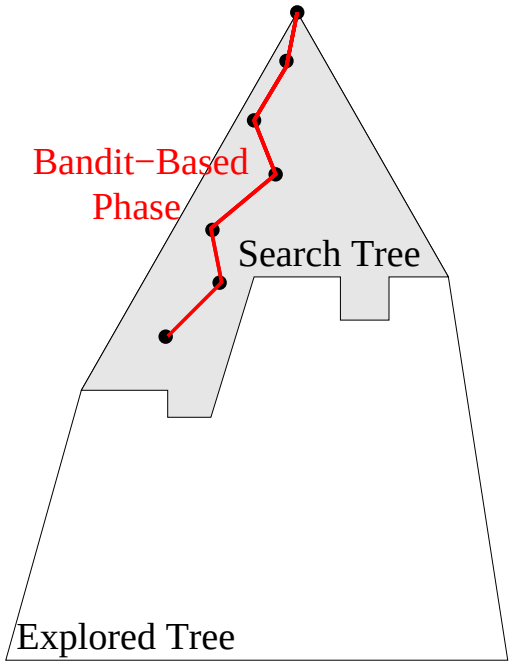
- Gradually grow a search tree
- Building Blocks
  - Select next action (bandit-based phase)
  - Add a node (leaf of the search tree)
  - Monte-Carlo exploration (random phase)
  - Compute instant reward
  - Update visited nodes
- Returned solution
  - Path visited most often



The UCT Monte-Carlo Tree Search

(Kocsis & Szepesvári, 06)

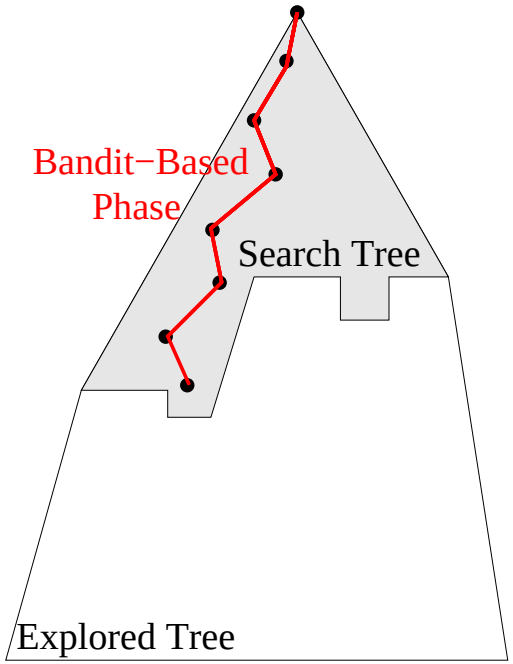
- Gradually grow a search tree
- Building Blocks
  - Select next action (bandit-based phase)
  - Add a node (leaf of the search tree)
  - Monte-Carlo exploration (random phase)
  - Compute instant reward
  - Update visited nodes
- Returned solution
  - Path visited most often



The UCT Monte-Carlo Tree Search

(Kocsis & Szepesvári, 06)

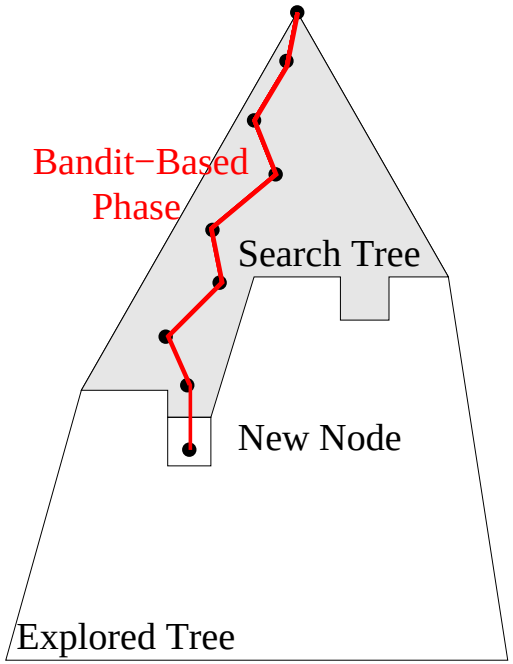
- Gradually grow a search tree
- Building Blocks
  - Select next action (bandit-based phase)
  - Add a node (leaf of the search tree)
  - Monte-Carlo exploration (random phase)
  - Compute instant reward
  - Update visited nodes
- Returned solution
  - Path visited most often



The UCT Monte-Carlo Tree Search

(Kocsis & Szepesvári, 06)

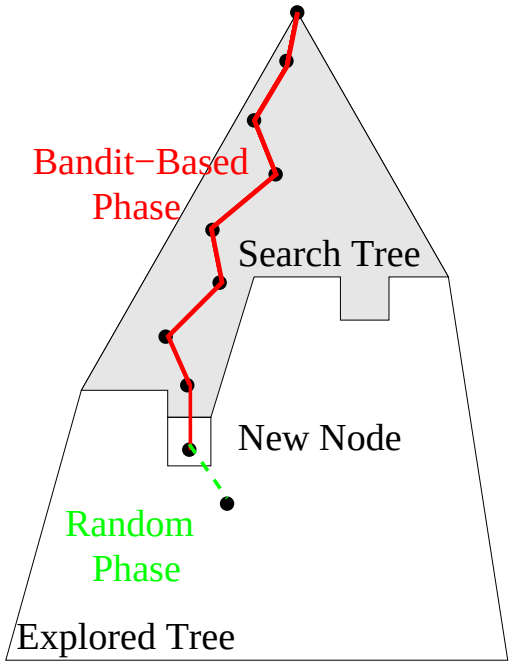
- Gradually grow a search tree
- Building Blocks
  - Select next action (bandit-based phase)
  - Add a node (leaf of the search tree)
  - Monte-Carlo exploration (random phase)
  - Compute instant reward
  - Update visited nodes
- Returned solution
  - Path visited most often



The UCT Monte-Carlo Tree Search

(Kocsis & Szepesvári, 06)

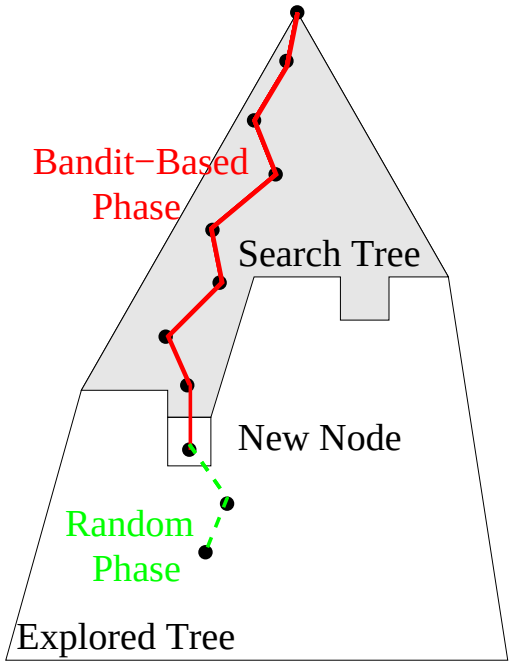
- Gradually grow a search tree
- Building Blocks
  - Select next action (bandit-based phase)
  - Add a node (leaf of the search tree)
  - Monte-Carlo exploration (random phase)
  - Compute instant reward
  - Update visited nodes
- Returned solution
  - Path visited most often



The UCT Monte-Carlo Tree Search

(Kocsis & Szepesvári, 06)

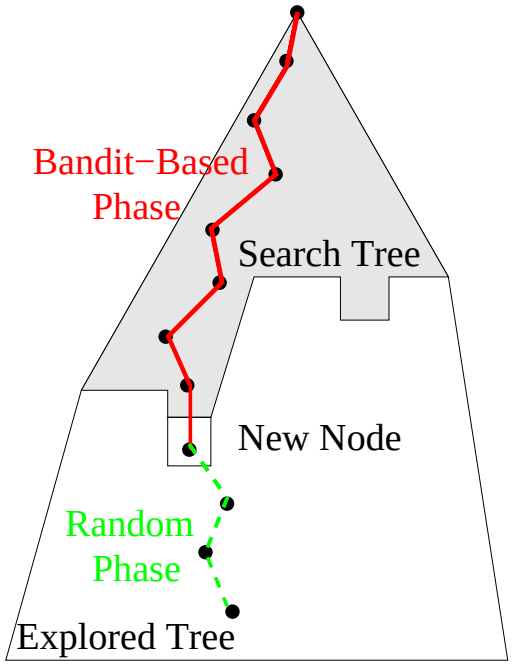
- Gradually grow a search tree
- Building Blocks
  - Select next action (bandit-based phase)
  - Add a node (leaf of the search tree)
  - Monte-Carlo exploration (random phase)
  - Compute instant reward
  - Update visited nodes
- Returned solution
  - Path visited most often



The UCT Monte-Carlo Tree Search

(Kocsis & Szepesvári, 06)

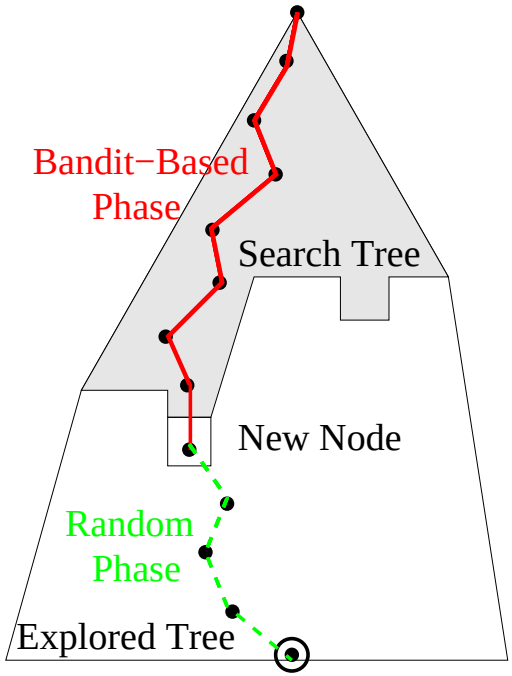
- Gradually grow a search tree
- Building Blocks
  - Select next action (bandit-based phase)
  - Add a node (leaf of the search tree)
  - Monte-Carlo exploration (random phase)
  - Compute instant reward
  - Update visited nodes
- Returned solution
  - Path visited most often



The UCT Monte-Carlo Tree Search

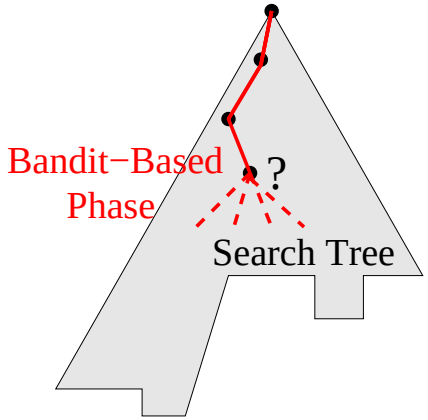
(Kocsis & Szepesvári, 06)

- Gradually grow a search tree
- Building Blocks
  - Select next action (bandit-based phase)
  - Add a node (leaf of the search tree)
  - Monte-Carlo exploration (random phase)
  - Compute instant reward
  - Update visited nodes
- Returned solution
  - Path visited most often



Multi-Arm Bandit-based phase

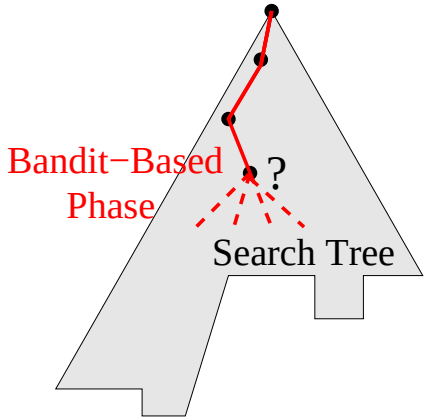
Upper Confidence Bound (UCB1-tuned) (Auer et al., 02)



- Exploration vs Exploitation trade-off
- Select $\underset{a \in A}{\text{argmax}}\ \hat{\mu}_a + \sqrt{\frac{c_e \log(T)}{t_a} min\left(\frac{1}{4}, \hat{\sigma}_a^2 + \sqrt{\frac{c_e \log(T)}{t_a}}\right)}$
  - $\hat{\mu}_a$: Empirical average reward for action a
  - $\hat{\sigma}_a^2$: Empirical variance of reward for action a
  - T: Total number of trials in current node
  - t_a: Number of trials for action a
  - c_e: Parameter

Multi-Arm Bandit-based phase

External information

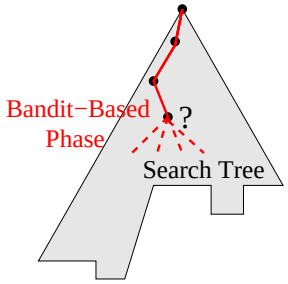


- Mixing UCB with
  - Priors on actions (Rolet et al., 09)
  - Information learned during iterations (Gelly & Silver, 07 ; Auer, 02 ; Filippi et al., 10)

Outline

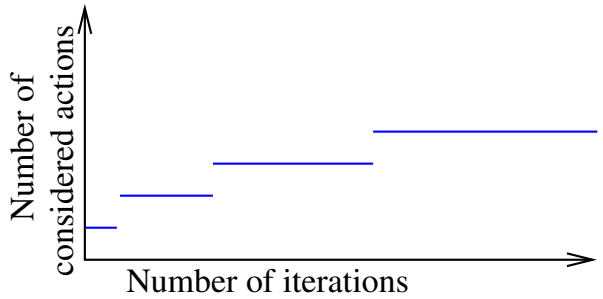
FUSE: bandit-based phase

A many-armed bandit problem



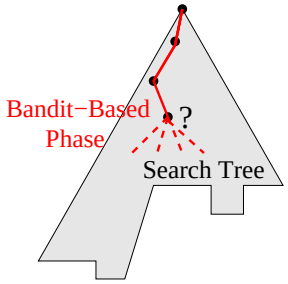
- Bottleneck
  - UCT degenerates to pure exploration as the number of arms increases (several hundred features)

  $\Rightarrow$
  - Control the number of arms
  - Select the arms
- How to control the number of arms?
  - Continuous heuristics (Gelly & Silver, 07)
    - Use a small exploration constant c_e (10^{-2}, 10^{-4})
  - Discrete heuristics (Coulom, 06; Rolet et al., 09)
    - Progressive Widening
    - Select a new action whenever $\lfloor T^b \rfloor$ increases ($b = \frac{1}{2}$ in experiments)



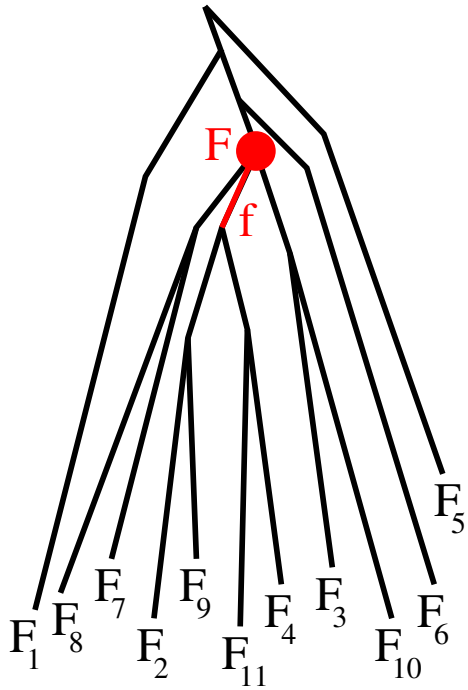
FUSE: bandit-based phase

Sharing information among nodes



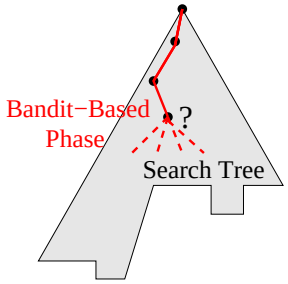
- How to share information among nodes?
  - Rapid Action Value Estimation (RAVE)
    (Gelly & Silver, 07)

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



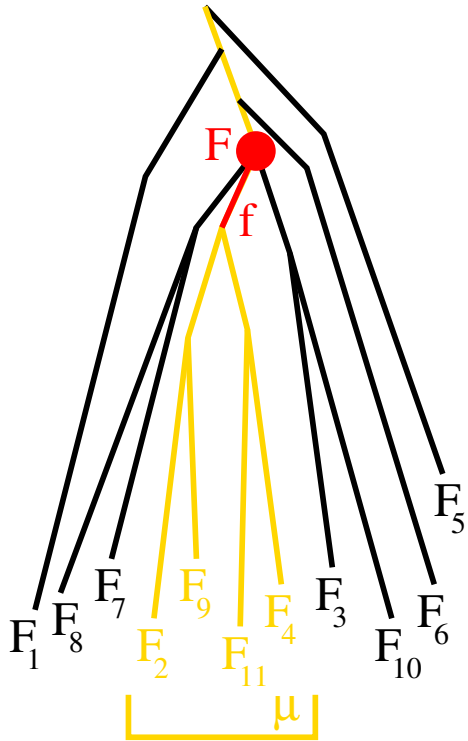
FUSE: bandit-based phase

Sharing information among nodes



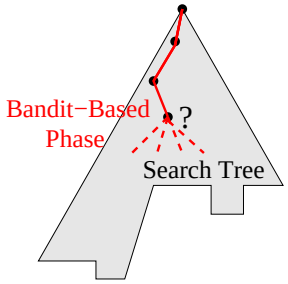
- How to share information among nodes?
  - Rapid Action Value Estimation (RAVE)
    (Gelly & Silver, 07)

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



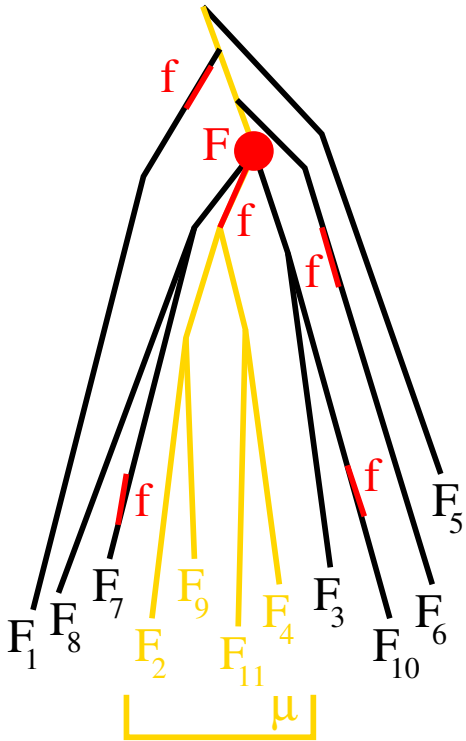
FUSE: bandit-based phase

Sharing information among nodes



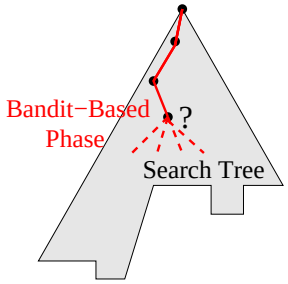
- How to share information among nodes?
  - Rapid Action Value Estimation (RAVE)
    (Gelly & Silver, 07)

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



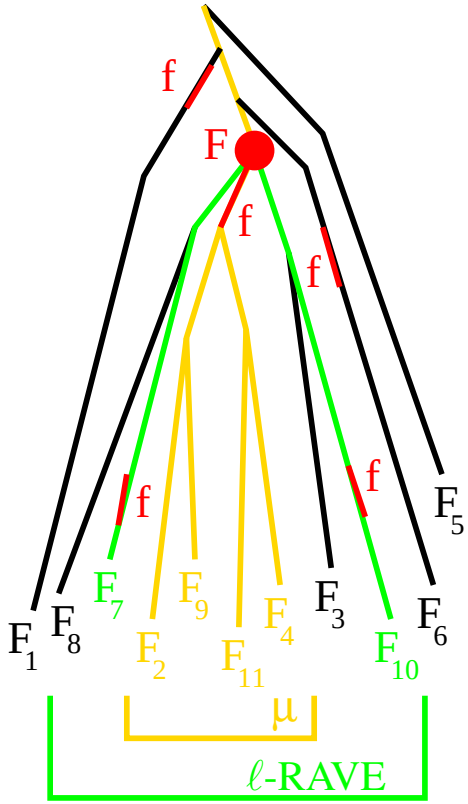
FUSE: bandit-based phase

Sharing information among nodes



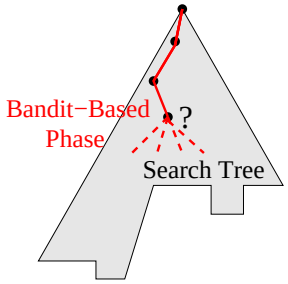
- How to share information among nodes?
  - Rapid Action Value Estimation (RAVE)
    (Gelly & Silver, 07)

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



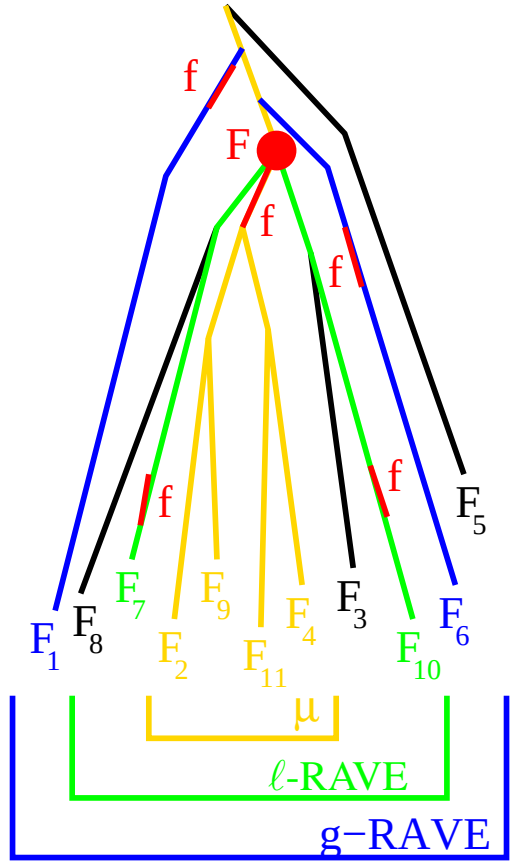
FUSE: bandit-based phase

Sharing information among nodes



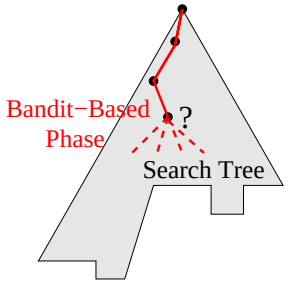
- How to share information among nodes?
  - Rapid Action Value Estimation (RAVE)
    (Gelly & Silver, 07)

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



FUSE: bandit-based phase

Sharing information among nodes



- Guiding search with RAVE
  - Continuous heuristics
    - Generalizing the empirical reward

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

with

$$\alpha=\frac{c}{c+t_{F,f}}$$

$$\beta=\frac{c'}{c'+t'_{F,f}}$$

$t_{F,f}$ = Number of trials for feature f from state F

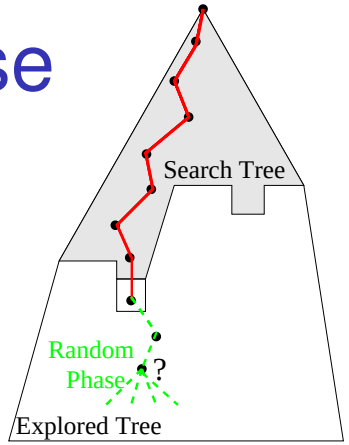
$t'_{F,f}$ = Number of trials for feature f after visiting state F

c, c': Parameters

  - Discrete heuristics
    - New arm $=\underset{f\in\mathcal{F}}{\text{argmax}}\ (1-\beta)\cdot\ell\text{-RAVE}(F,f)+\beta\cdot\text{g-RAVE}(f)$

FUSE: random phase

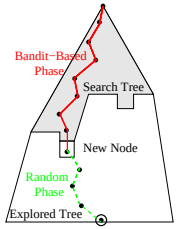
Dealing with an unknown horizon



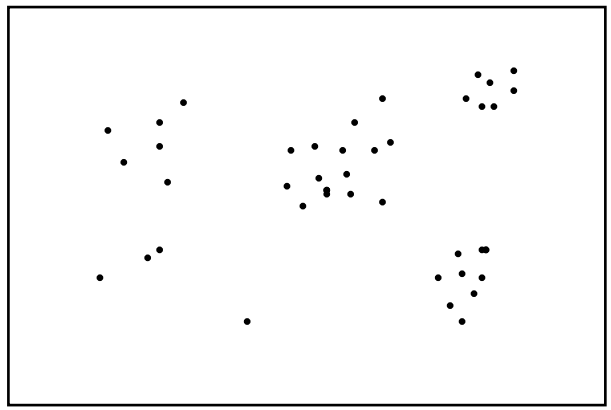
- Bottleneck
  - Finite unknown horizon (= number of relevant features)
- Random phase policy
  - With probability $1 - q^{|F|}$ stop
  - Else • add a uniformly selected feature
    - $|F| = |F| + 1$
  - Iterate

FUSE: reward(F)

Generalization error estimate



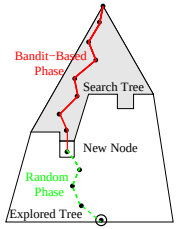
- Requisite
  - fast (to be computed 10^4 times)
  - unbiased
- Proposed reward
  - k-NN: strong consistency results (Cover & Hart, 67)
  - + AUC criterion *
- Complexity: $\tilde{O}(mnd)$
  - d Number of selected features
  - n Size of the training set
  - m Size of sub-sample ($m \ll n$)



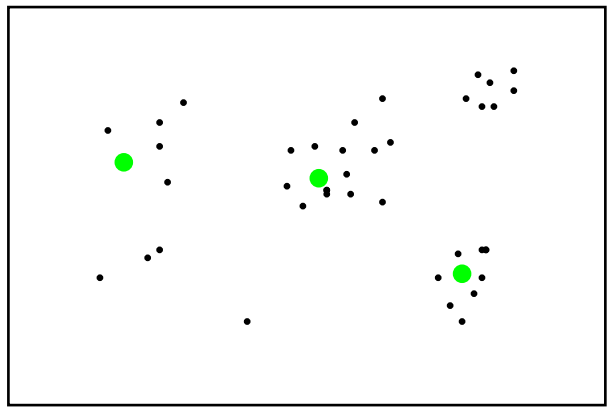
* Mann Whitney Wilcoxon test: $V(F) = \frac{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ \mathcal{N}_{F,k}(x) < \mathcal{N}_{F,k}(x'),\ y < y'\}|}{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ y < y'\}|}$

FUSE: reward(F)

Generalization error estimate



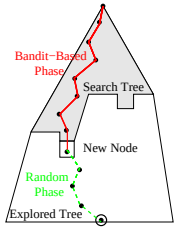
- Requisite
  - fast (to be computed 10^4 times)
  - unbiased
- Proposed reward
  - k-NN: strong consistency results (Cover & Hart, 67)
  - + AUC criterion *
- Complexity: $\tilde{O}(mnd)$
  - d Number of selected features
  - n Size of the training set
  - m Size of sub-sample ($m \ll n$)



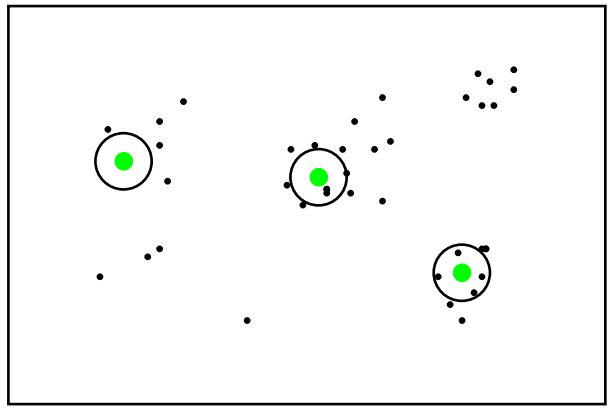
* Mann Whitney Wilcoxon test: $V(F) = \frac{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ \mathcal{N}_{F,k}(x) < \mathcal{N}_{F,k}(x'),\ y < y'\}|}{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ y < y'\}|}$

FUSE: reward(F)

Generalization error estimate



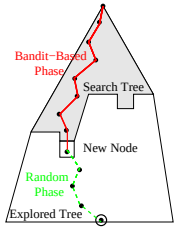
- Requisite
  - fast (to be computed 10^4 times)
  - unbiased
- Proposed reward
  - k-NN: strong consistency results (Cover & Hart, 67)
  - + AUC criterion *
- Complexity: $\tilde{O}(mnd)$
  - d Number of selected features
  - n Size of the training set
  - m Size of sub-sample ($m \ll n$)



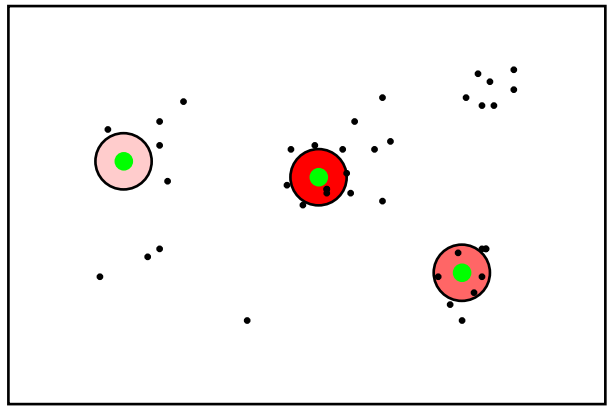
* Mann Whitney Wilcoxon test: $V(F) = \frac{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ \mathcal{N}_{F,k}(x) < \mathcal{N}_{F,k}(x'),\ y < y'\}|}{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ y < y'\}|}$

FUSE: reward(F)

Generalization error estimate



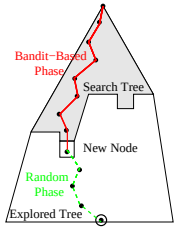
- Requisite
  - fast (to be computed 10^4 times)
  - unbiased
- Proposed reward
  - k-NN: strong consistency results (Cover & Hart, 67)
  - + AUC criterion *
- Complexity: $\tilde{O}(mnd)$
  - d Number of selected features
  - n Size of the training set
  - m Size of sub-sample ($m \ll n$)



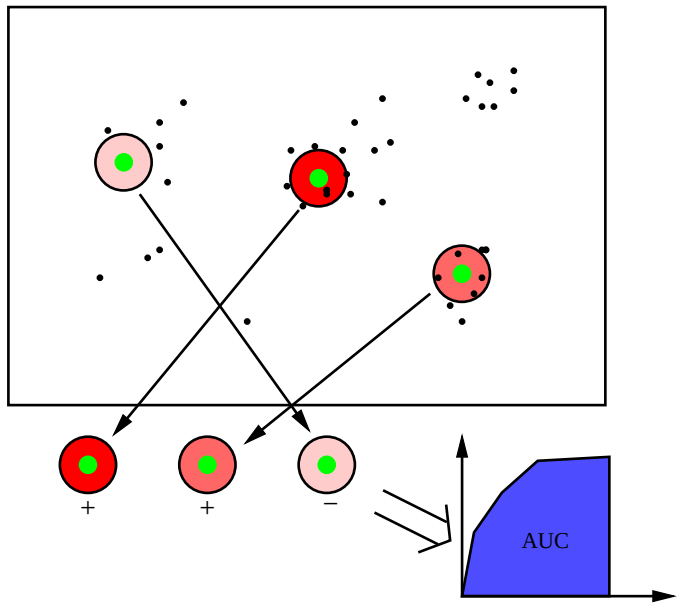
* Mann Whitney Wilcoxon test: $V(F) = \frac{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ \mathcal{N}_{F,k}(x) < \mathcal{N}_{F,k}(x'),\ y < y'\}|}{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ y < y'\}|}$

FUSE: reward(F)

Generalization error estimate

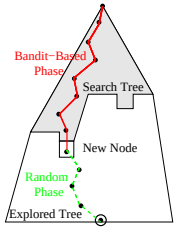


- Requisite
  - fast (to be computed 10^4 times)
  - unbiased
- Proposed reward
  - k-NN: strong consistency results (Cover & Hart, 67)
  - + AUC criterion *
- Complexity: $\tilde{O}(mnd)$
  - d Number of selected features
  - n Size of the training set
  - m Size of sub-sample ($m \ll n$)

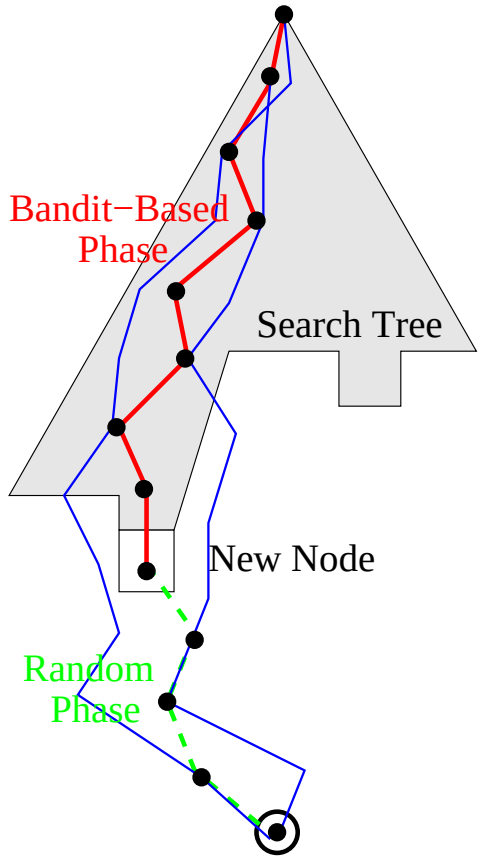


* Mann Whitney Wilcoxon test: $V(F) = \frac{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ \mathcal{N}_{F,k}(x) < \mathcal{N}_{F,k}(x'),\ y < y'\}|}{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ y < y'\}|}$

FUSE: update



- Explore a graph
  - $\Rightarrow$ Several paths to the same node
- Update followed path only



From UCT to Feature Selection to Learning

- Algorithm
  - N iterations: each iteration $\begin{cases} 1. \text{ Follows a path} \\ 2. \text{ Evaluates a final node} \end{cases}$
- Output

Search tree	$\longleftrightarrow$	RAVE score
$\Downarrow$		$\Downarrow$
FUSE		**FUSE**^{R}
Wrapper approach		Filter approach
Most visited path		Using RAVE as score

- End learner
  - Any Machine Learning algorithm
  - Support Vector Machine with Gaussian kernel in experiments

Outline

Experimental setting

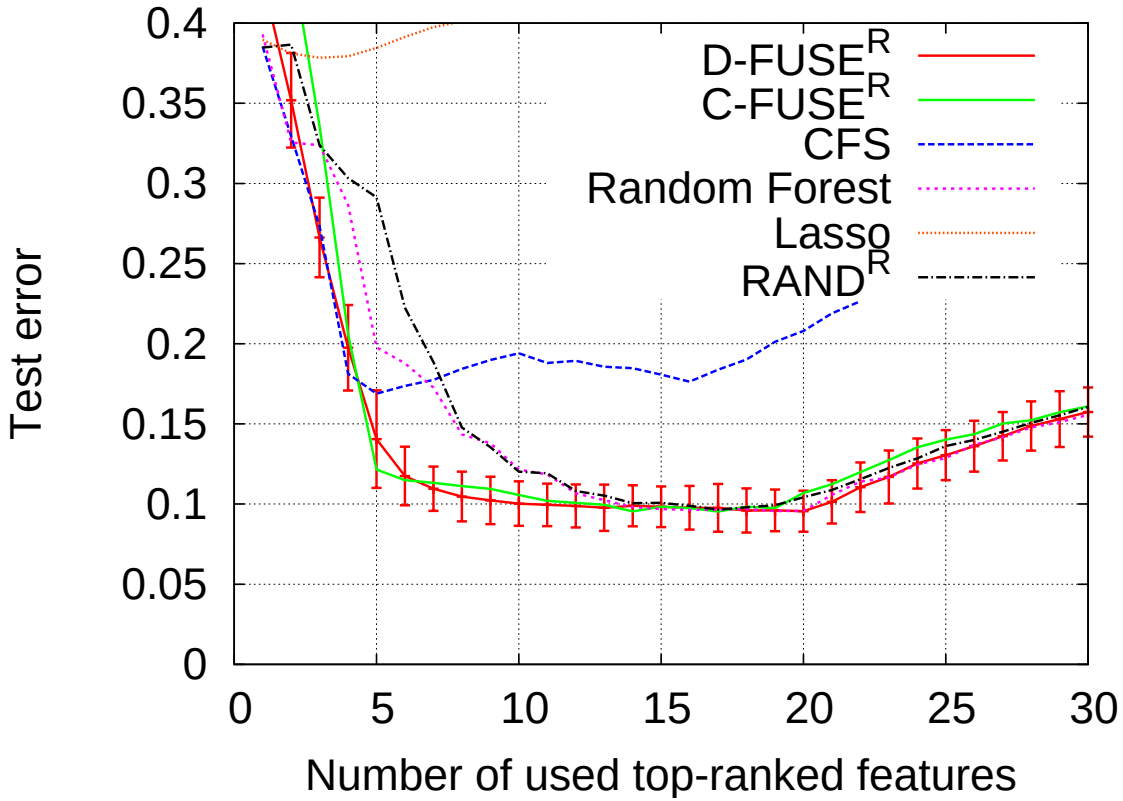
- Questions
  - FUSE vs FUSE^R
  - Continuous vs discrete exploration heuristics
  - FS performance w.r.t. complexity of the target concept
  - Convergence speed
- Datasets from NIPS'03 Feature Selection challenge

Data set	Samples	Features	Properties
Madelon	2,600	500	XOR-like
Arcene	200	10,000	Redundant features
Colon	62	2,000	"Easy"

Experimental setting

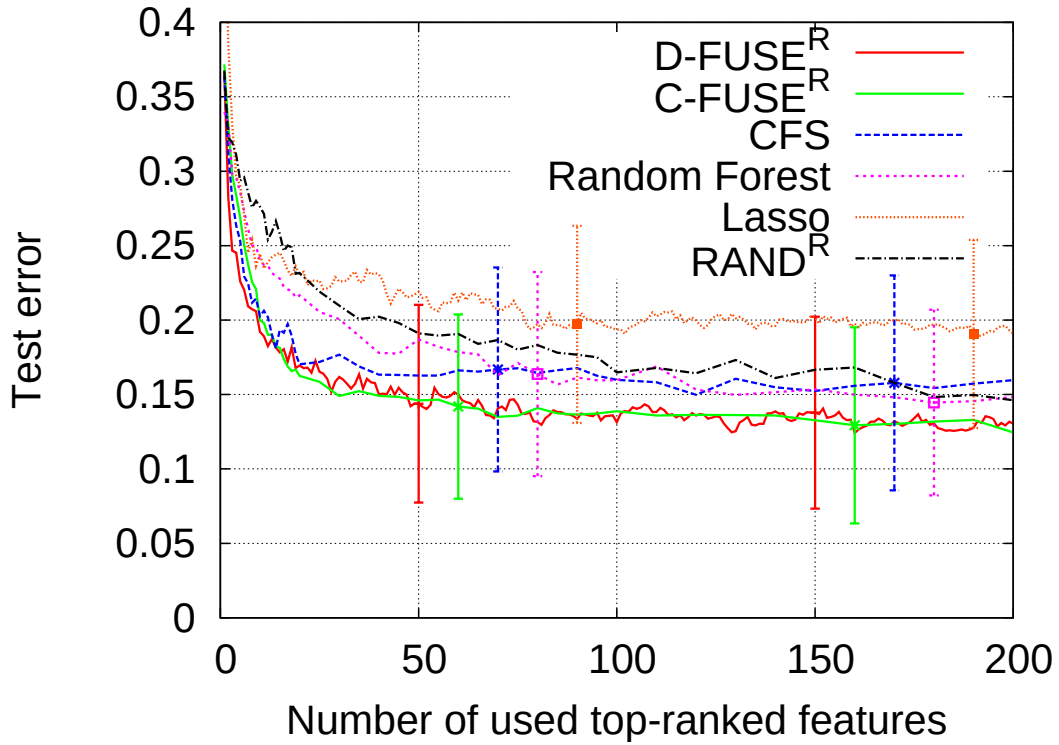
- Baselines
  - CFS (Constraint-based Feature Selection) (Hall, 00)
  - Random Forest (Rogers & Gunn, 05)
  - Lasso (Tibshirani, 94)
  - RAND^R: RAVE obtained by selecting 20 random features at each iteration
- Results averaged on 50 splits (10 $\times$ 5 fold cross-validation)
- Gaussian SVM
  - Hyper-parameters optimized by 5 fold cross-validation

Results on Madelon after 200,000 iterations



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

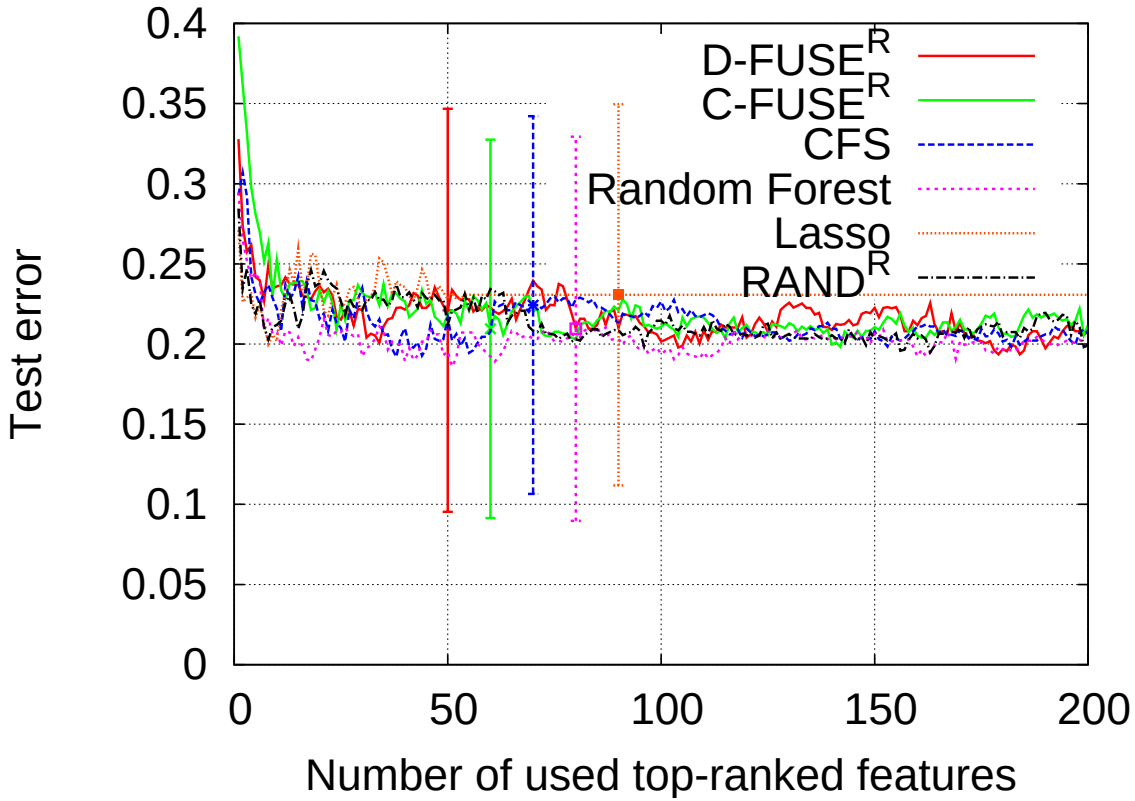
Results on Arcene after 200,000 iterations



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

T-test "CFS vs. FUSE^R" with 100 features: p-value=0.036

Results on Colon after 200,000 iterations



- Comment
  - All equivalent

NIPS 2003 Feature Selection challenge

- Test error on the NIPS 2003 Feature Selection challenge
  - On an 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
ARCENE	BAYES-NN-RED [2]	7.20% (1^{st})	100	0
	D-FUSE^R(ON ALL)	8.42% (3^{rd})	500	34
	D-FUSE^R	9.42% 500 (8^{th})	500	0

- Comment
  - Accurate w.r.t Feature Selection

[1] K. Q. Shen, C. J. Ong, X. P. Li, E. P. V. Wilder-Smith *Feature selection via sensitivity analysis of SVM probabilistic outputs*. Mach. Learn. 2008

[2] R. M. Neal, and J. Zhang *Chap. High Dimensional Classification with Bayesian Neural Networks and Dirichlet Diffusion Trees*. Feature extraction, foundations and applications, Springer 2006

Partial conclusion on Feature Selection

Contributions

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

- Perspectives
  - Proof of convergence (including heuristics) (Berthier et al., 10)
  - Include other improvements from Reinforcement Learning community
    - Function approximation of the Q-value (Melo et al., 08; Auer, 02)
    - Biased random phase (Rimmel & Teytaud, 10)

Conclusion

- Focus on combinatorial optimization problems hidden in Machine Learning
- Relational Learning
  - Theoretical and empirical limitations of averaging-kernels
- Feature Selection
  - Exploration of the feature lattice using a Monte-Carlo tree search approach

$\Longrightarrow$ refining wrapper approaches using a frugal assessment of candidate subsets

Perspective 1: Constructive Induction

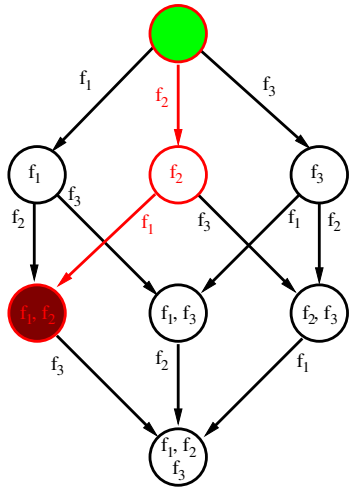
- Context
  - Relational Learning / Inductive Logic Programming
- Goal
  - Find a relevant set of primitives / queries

  $\Longrightarrow$ combinatorial optimization problem
- Proposed approach
  - Extending FUSE to grammar structured search spaces (de Mesmay et al., 09)
- Motivating applications
  - Customer Relationship Management

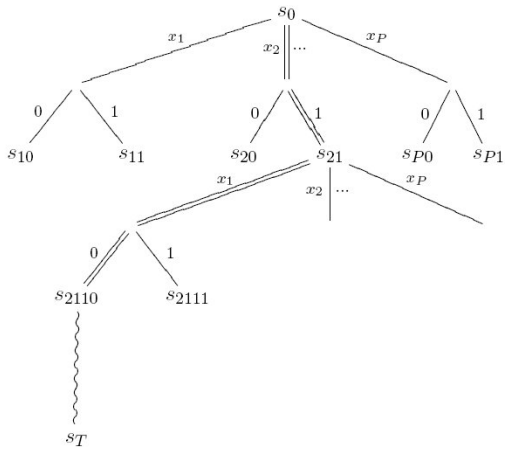
Perspective 2: Feature/Example Selection

- Context
  - FUSE: Feature Selection based on UCT
  - BAAL: Active Learning based on UCT
    (Rolet et al. 09)

  $\Longrightarrow$ Can we mix both approaches?



- Goal
  - Local Feature Selection
  - Local Distance Metric Learning
    (Weinberger & Saul, 09)



Bibliography 1/3

P. Auer *Using confidence bounds for exploitation-exploration trade-offs*. JMLR'02
P. Auer, N. Cesa-Bianchi, and P. Fischer *Finite-time analysis of the Multiarmed Bandit Problem*. ML'02
F. Bach *Exploring large feature spaces with hierarchical Multiple Kernel Learning*. NIPS'08
V. Berthier, H. Doghmen, and O. Teytaud *Consistency Modifications for Automatically Tuned Monte-Carlo Tree Search*. CAP'10
M. Botta, A. Giordana, L. Saitta, and M. Sebag *Relational Learning as search in a critical region*. JMLR'03
L. Bottou, and O. Bousquet *The Tradeoffs of Large Scale Learning*. NIPS'08
M. Boullé *Compression-based averaging of selective Naive Bayes classifiers*. J. Mach. Learn. Res. 07
L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen *Classification and regression trees*. Taylor & Francis, Inc., 84
P. Cheeseman, B. Kanefsky, and W. M. Taylor *Where the really hard problems are*. IJCAI'91
R. Coulom *Efficient selectivity and backup operators in Monte-Carlo tree search*. Computer and Games 06
T. Cover, and P. Hart *Nearest neighbor pattern classification*. IEEE Transactions on Information Theory 1967
R. Féraud, M. Boullé, F. Clérot, F. Fessant, V. Lemaire *The Orange Customer Analysis Platform*. ICDM'10

Bibliography 2/3

S. Filippi, O. Cappé, A. Garivier and C. Szepesvári *Parametric Bandits: The Generalized Linear Case*. NIPS'10
S. Gelly, and D. Silver *Combining online and offline knowledge in UCT*. ICML'07
A. Giordana, and L. Saitta *Phase Transitions in Relational Learning*. Mach. Learn. 00
M. A. Hall *Correlation-based Feature Selection for discrete and numeric class Machine Learning*. ICML'00
K. Kira, and L. A. Rendell *A practical approach to feature selection*. ML'92
L. Kocsis, and C. Szepesvári *Bandit based Monte-Carlo planning*. ECML'06
D. Margaritis *Toward provably correct Feature Selection in arbitrary domains*. NIPS'09
F. Melo, S. Meyn, and I. Ribeiro *An Analysis of Reinforcement Learning with Function Approximation*. ICML'08
F. de Mesmay, A. Rimmel, Y. Voronenko, and M. Püschel *Bandit-based optimization on graphs with application to library performance tuning*. ICML'09
R. M. Neal, and J. Zhang *Chap. High Dimensional Classification with Bayesian Neural Networks and Dirichlet Diffusion Trees*. Feature extraction, foundations and applications, Springer 2006
J. Rogers, and S. R. Gunn *Identifying feature relevance using a Random Forest*. SLSFS'05
P. Rolet, M. Sebag, and O. Teytaud *Boosting Active Learning to optimality: a tractable Monte-Carlo, Billiard-based algorithm*. ECML'09
K. Q. Shen, C. J. Ong, X. P. Li, E. P. V. Wilder-Smith *Feature selection via sensitivity analysis of SVM probabilistic outputs*. Mach. Learn. 08

Bibliography 3/3

R. Tibshirani *Regression shrinkage and selection via the Lasso*. Journal of the Royal Statistical Society 94

K. Q. Weinberger, L. K. Saul *Distance Metric Learning for Large Margin Nearest Neighbor Classification*. JMLR'09

T. Zhang *Adaptive forward-backward greedy algorithm for sparse learning with linear models*. NIPS'08

Relational Kernels

- Position
  - Relational data
    - $\mathcal{X}$: keys in a database
- Bottlenecks
  - $\mathcal{H}$: set of logical formula
  - h: logical formula
- Support Vector Machine: the solution ?
  - A propositionalization
    - Use only relations between examples

$\Longrightarrow$ argmin easy to solve

Contribution

On Multiple-Instance datasets, averaging-kernels miss some concepts

- Theoretical/empirical identification of failure region + lower bound on generalization error

Feature Selection

- Position
  - Thousands of features (& Only estimation of **Err** $(\mathcal{A}(h, \mathcal{L}))$)

  $\Longrightarrow$ Overfitting: small error on training data / large generalization error

- Solution: Feature Selection

$$\underset{F \subseteq \mathcal{F}}{\text{argmin}} \ \textbf{Err}\left(\mathcal{A}(F, \mathcal{L})\right)$$

$\mathcal{F}$: Set of features
F: Feature subset
$\mathcal{L}$: Training data set
$\mathcal{A}$: ML algorithm

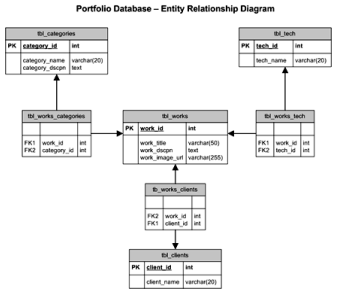
- Bottlenecks
  - Combinatorial optimization problem: find $F \subseteq \mathcal{F}$
  - Unknown objective function: generalization error

Contribution

Actually handle combinatorial optimization

- Use a Monte-Carlo Tree Search algorithm: UCT (Kocsis & Szepesvári, 06)

Relational Kernels



- Position
  - Relational data
    - $\mathcal{X}$: keys in a database
- $\mathcal{H}$: set of logical formula

$\Longrightarrow$
  - Expressive language
  - Value of a hypothesis on an example: NP-hard
  - Number of hypothesis to test: exponential

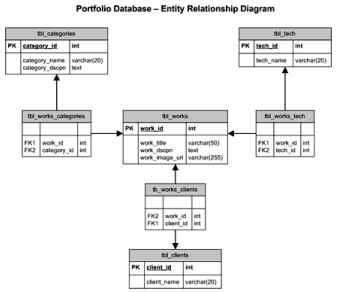
- Support Vector Machine (SVM)
  - Only based on relations between examples

$\Longrightarrow$
  - Value of a hypothesis on one example: linear in # examples
  - Best hypothesis search $\equiv$ convex problem

Question

Does SVM have the same expressiveness as logical formula?

Relational Kernels



- Position
  - Relational data
    - $\mathcal{X}$: keys in a database

Multiple Instance data

- $\mathcal{H}$: set of logical formula

$\Longrightarrow$
- Expressive language
- Value of a hypothesis on an example: NP-hard
- Number of hypothesis to test: exponential

- Support Vector Machine (SVM)
  - Only based on relations between examples

Averaging kernel

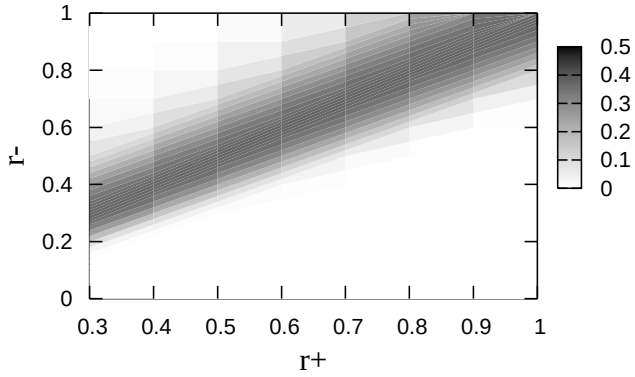
$\Longrightarrow$
- Value of a hypothesis on one example: linear in # examples
- Best hypothesis search $\equiv$ convex problem

Question

Does SVM have the same expressiveness as logical formula?

Relational Kernels failure

- A Phase Transition-based study
  - Identify order parameters
  - Generate artificial problems
  - Identify difficult region



Contribution

On Multiple-Instance data, averaging-kernels miss some concepts

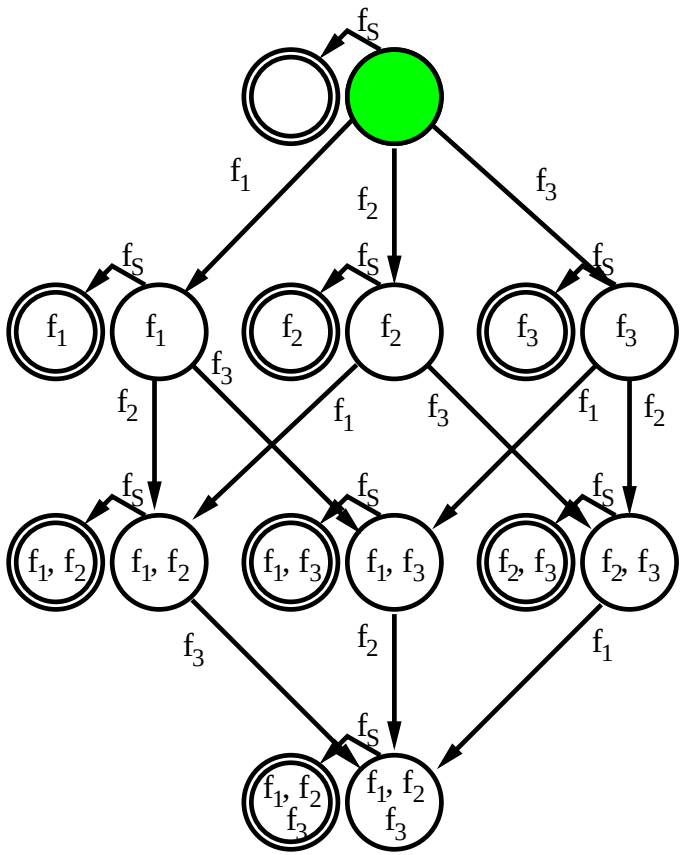
- Theoretical demonstration of relational kernels failure region
- New criterion leading to a lower bound on generalization error
- Empirical visualization of failure region

- Discussion
  - What about other kernels?

Stopping feature

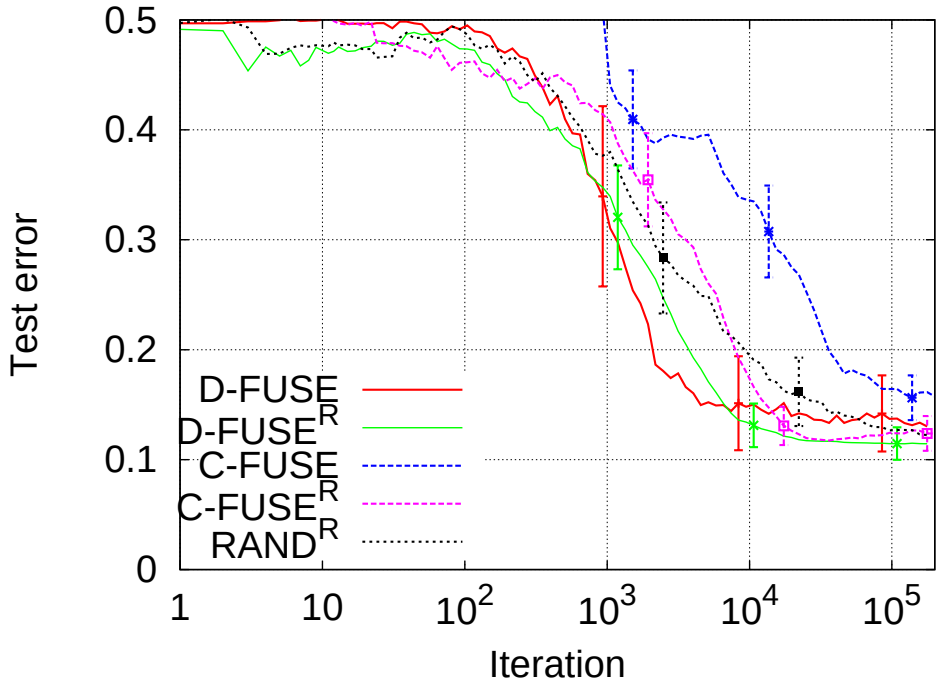
Dealing with an unknown horizon

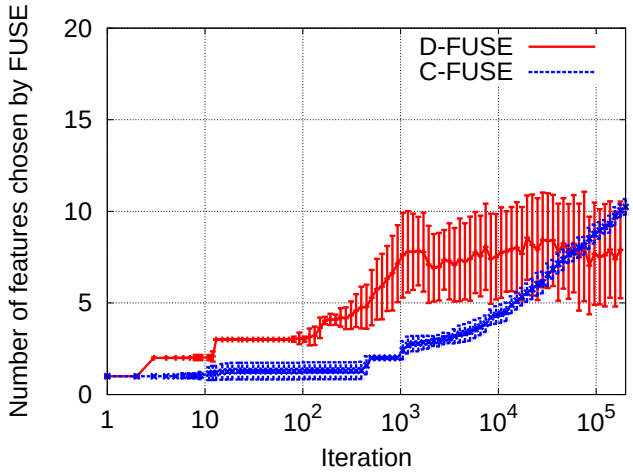
- Any state can be final or not
  - Final(F) = "$f_s \in F$"
  - f_s: A virtual stopping feature
- RAVE(f_s)
  - g-RAVE($f_s^{(d)}$) = *average* $\{V(F_t), |F_t| = d + 1\}$
    - $V(F_t)$: Reward of Feature Subset F_t selected at iteration t
    - d: When RAVE(f_s) is used, d is set to the number of features in current state



Sensitivity of FUSE to the Computational Effort

Madelon





- Comments
  - FUSE: not enough features
  - FUSE^R: 10 times faster than RAND^R

FUSE hyperparameters

			How to restrict exploration		
			Disc. heur.	Continuous heuristics	
Param.	k-NN	q	b	c_e	c, c'
Value i	5-NN	$1-10^i$ $\{-1,-3,-5\}$	1/2	10^i $\{-4,-2,0,2\}$	10^i $\{-\infty,2,4\}$
Best value					
Arcene	5-NN	$1-10^{-1}$	1/2		
	5-NN	$1-10^{-1}$		10^{-2}	Any
	5-NN	$1-10^{-3}$		10^{-4}	Almost any
Madelon	5-NN	$1-10^{-3}$	1/2		
	5-NN	$1-10^{-1}$		10^{-2}	$\{(10^2,0),(10^4,0)\}$
Colon	5-NN	$1-10^{-5}$	1/2		
	5-NN	$1-10^{-5}$		Any	Almost any