Master Recherche IAC
Option 2
Robotique et agents autonomes

Jamal Atif – Michèle Sebag
LRI

Feb. 7th, 2014

Overview

Starting point: gathering the data

	AGE	SEX	BMI	BP	··· Serum Measurements ···						Response
Patient	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	y
1	59	2	32.1	101	157	93.2	38	4	4.9	87	151
2	48	1	21.6	87	183	103.2	70	3	3.9	69	75
3	72	2	30.5	93	156	93.6	41	4	4.7	85	141
4	24	1	25.3	84	198	131.4	40	5	4.9	89	206
5	50	1	23.0	101	192	125.4	52	4	4.3	80	135
6	23	1	22.6	89	139	64.8	61	2	4.2	68	97
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
441	36	1	30.0	95	201	125.2	42	5	5.1	85	220
442	36	1	19.6	71	250	133.2	97	3	4.6	92	57

Find features

Before learning: describe the examples

- Too poor a description $\Rightarrow$ nothing possible
- Too rich $\Rightarrow$ feature pruning is required

Why ?

- ML is not a well-posed problem
- $\Longrightarrow$ Adding useless features (the captain's age) can deteriorate the hypotheses

Feature Selection, Position of the problem

Context

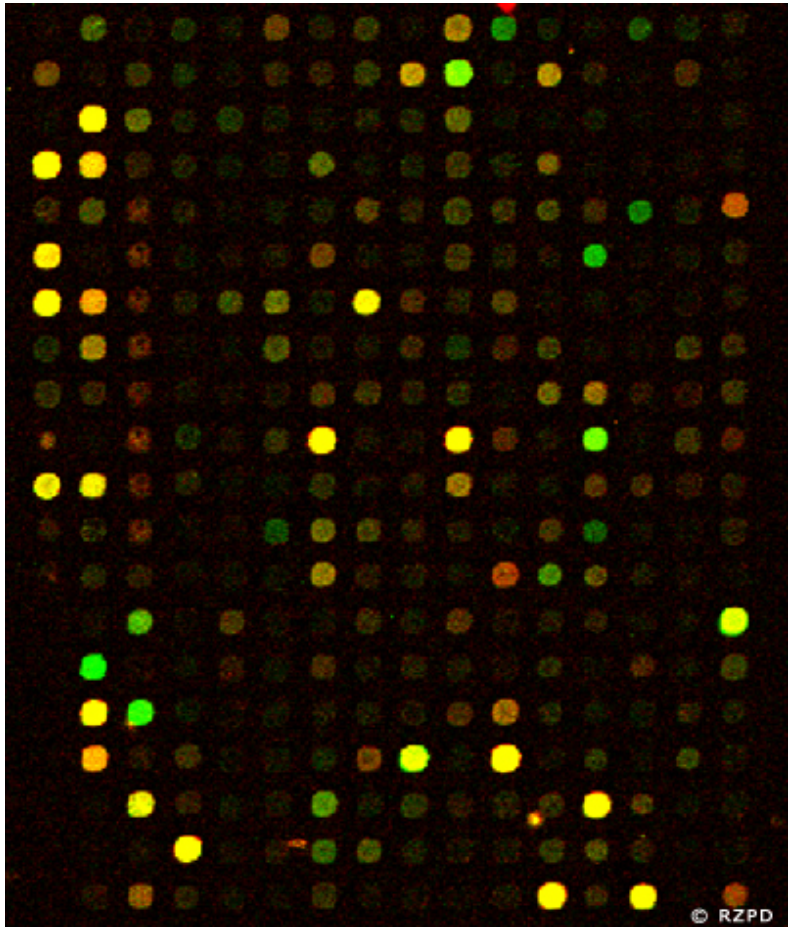
- Too many features wrt number of examples
  - Remove — Feature Selection
  - Build new features — Feature Construction
  - Project on few features — Dimensionality Reduction
- A particular case, first-order logic: — Propositionalisation

The hidden goal: select or build features ?

- Feature Construction : build good features
- .. makes learning easier...
- Best features: good hypotheses.

When learning boils down to feature selection

Bio-informatics



- 30 000 genes
- few samples (expensive)
- goal: find genes relevant to diseases, resilience, ...

Position of the problem

Goals

- Selection: find a subset of features
- Ranking: order features by increasing relevance

Formalization

Given $\mathcal{A} = \{a_1, ..a_d\}$. Define

$$\begin{aligned} \mathcal{F} : \mathcal{P}(\mathcal{A}) &\mapsto \mathbb{R} \\ A \subset \mathcal{A} &\mapsto Err(A) = \text{ min error of hypotheses built from } A \end{aligned}$$

Find $Argmin(\mathcal{F})$

Challenge

- A combinatorial optimization problem (2^d)
- An unknown optimization function $\mathcal{F}$

Feature selection: the filter approach

Univariate approach

Define $score(a_i)$; iteratively add features by decreasing *score* order
or iteratively remove features with increasing *score*

PROS simple, inexpensive

CONS very local optima

Backtrack possible

- Given current solution $\mathcal{A}$
- Add a_i to $\mathcal{A}$
- Examine whether removing a_j is relevant

Backtrack = less greedy, better optima, much more expensive

Feature selection: the wrapping approach

Multivariate approach

Measure the quality of a feature subset:

$$\text{estimate } \mathcal{F}(a_{i1}, \ldots a_{ik})$$

CONS

Expensive: an estimate = solving an ML problem.

PROS

Better optima

Feature selection: embedded approach

Principle (beforehand)

An ML criterion which favors hypotheses with few features

For instance: find w, $h(x) = < w, x >$, $=$ argmin

$$\sum_i (h(x_i) - y_i)^2 + ||w||_1$$

data fitting favor w with many null coordinates

Principle – a posteriori

Given

$$h(x) = < w, x > = \sum_{j=1}^{d} w_j x_j$$

If $|w_j|$ small, the j-th feature is unimportant

Remove and restart the learning.

Filter approaches, 1

Notations

Training set: $\mathcal{E} = \{(x_i, y_i), i = 1..n, y_i \in \{-1, 1\}\}$

$a(x_i)$= value of feature a for example (x_i)

Correlation

$$corr(a) = \frac{\sum_i a(x_i).y_i}{\sqrt{\sum_i (a(x_i))^2 \times \sum_i y_i^2}} \propto \sum_i a(x_i).y_i = \ < a, y >$$

Limitations

Correlated features

Non linear dependencies

Filter approaches, 2

Correlation and projection

Stoppiglia et al. 2003

Repeat

- select $a^* =$ feature most correlated to target

$$a^* = argmax\{\sum_i a(x_i)y_i,\ a \in \mathcal{A}\}$$

- Project all other features on orthogonal space:

$$\forall b \in \mathcal{A} \qquad b \to \quad b - \frac{<a^*,b>}{<a^*,a^*>}\ a^*$$

$$b(x_i) \to \quad b(x_i) - \frac{\sum_j a^*(x_j)b(x_j)}{\sqrt{\sum_j a^*(x_j)^2}\sqrt{\sum_j b(x_j)^2}} a^*(x_i)$$

Correlation and projection, cont

- Project y on orthogonal space too

$$y \rightarrow y - \frac{<a^*, y>}{<a^*, a^*>} a^*$$

$$y_i \rightarrow y_i - - \frac{\sum_j a^*(x_j) y_j}{\sum_j a^*(x_j)^2} a^*(x_i)$$

- Until stopping criterion
  - Add random features ($r(x_i) = \pm 1$) *probe*
  - When probes are selected, stop.

Limitations

does not work well when there are more than 6-7 relevant features (numerical noise).

Filter approaches, 3

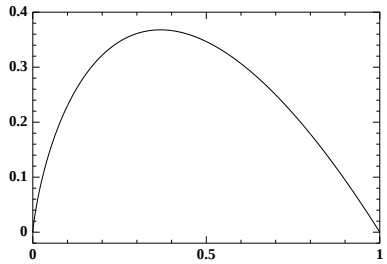
Information gain

decision trees

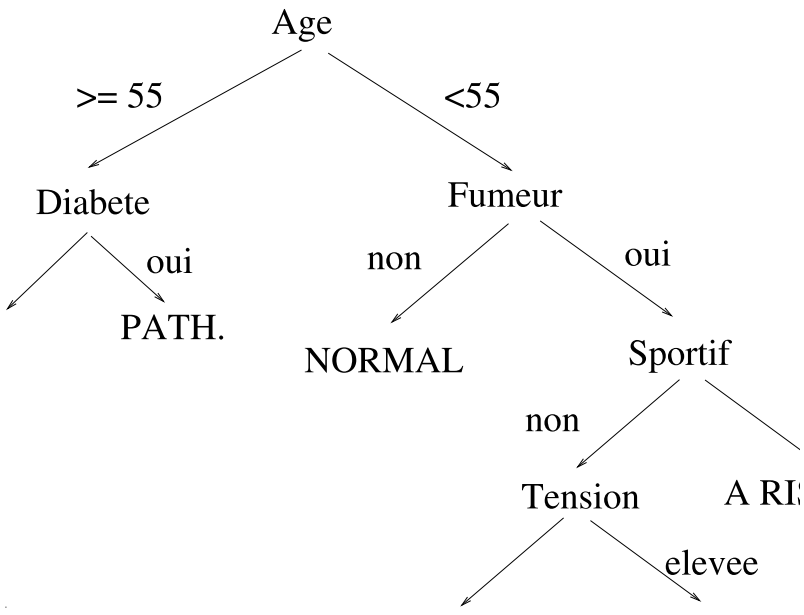
$$p([a = v]) = Pr(y = 1|a(x_i) = v)$$

$$QI([a = v]) = -p([a = v]) \log p([a = v])$$

$$QI(a) = \sum_v Pr(a(x_i) = v) QI([a = v])$$



Information gain, contd



Limitations

- Myopic criterion — the XOR case
- Favors many-valued features
- Not well-suited to numerical features

Scores

in text mining, supervised learning

Notations : c_i a class a_k a word or term

Criteria

1. Conditional probability $P(c_i|a_k)$
2. Mutual information $P(c_i, a_k)Log(\frac{P(c_i,a_k)}{P(c_i)P(a_k)})$
3. Chi-2 $\frac{(P(t,c)P(\neg t,\neg c)-P(t,\neg c)P(\neg t,c))^2}{P(t)P(\neg t)P(c)P(\neg c)}$
4. Relevance $\frac{P(t,c)+d}{P(\neg t,\neg c)+d}$

Wrapper approaches

Principle: Generate and test

Given a list of candidate subsets $\mathcal{L} = \{A_1, .., A_p\}$

- Generate a new candidate A
- Compute $\mathcal{F}(A)$
  - learn h_A from $\mathcal{E}_{|A}$
  - test h_A on a test set $= \hat{\mathcal{F}}(A)$
- Update $\mathcal{L}$.

Algorithms

- hill-climbing / multiple restart
- genetic algorithms
- genetic programming

Embedded approaches, 2

Principle

- Build a hypothesis
- Detect irrelevant features
- Prune them
- Iterate

Algorithm : SVM Recursive Feature Elimination Guyon et al. 03

- Linear SVM $\rightarrow h(x) = sign(\sum w_i.a_i(x) + b)$
- relevance(a_i) approx $|w_i|$
- Prune the bottom-k features
- Iterate.

Overview

Dimensionality Reduction – Intuition

Degrees of freedom

- Image: 4096 pixels; but not independent
- Robotics: (# camera pixels + # infra-red) $\times$ time; but not independent

Goal

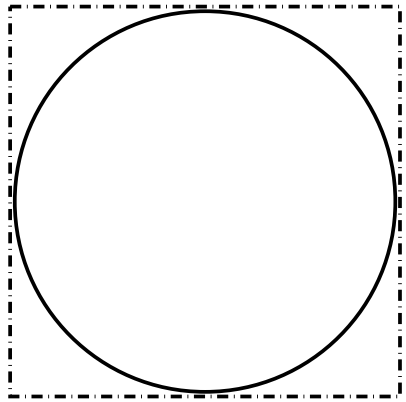
Find the (low-dimensional) structure of the data:

- Images
- Robotics
- Genes

Dimensionality Reduction

In high dimension

- Everybody lives in the corners of the space
  Volume of Sphere $V_n = \frac{2\pi r^2}{n} V_{n-2}$
- All points are far from each other



Approaches

- Linear dimensionality reduction
  - Principal Component Analysis
  - Random Projection
- Non-linear dimensionality reduction

Criteria

- Complexity/Size
- Prior knowledge

e.g., relevant distance

Linear Dimensionality Reduction

Training set *unsupervised*

$$\mathcal{E} = \{(\mathbf{x}_k), \mathbf{x}_k \in \mathbb{R}^D, k = 1 \dots N\}$$

Projection from $\mathbb{R}^D$ onto $\mathbb{R}^d$

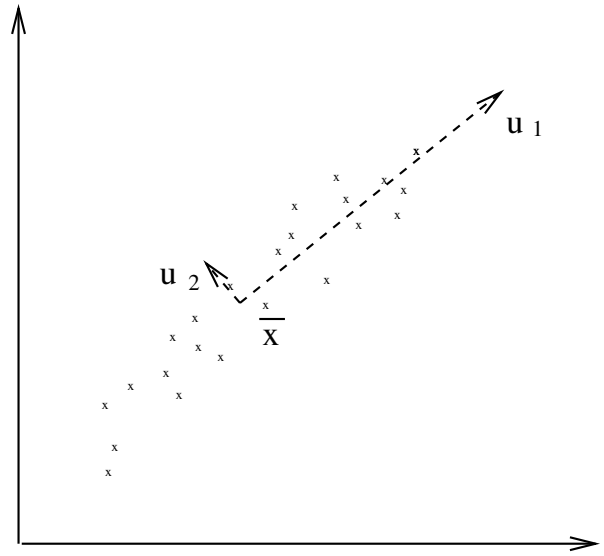
$$\mathbf{x} \in \mathbb{R}^D \rightarrow \quad h(\mathbf{x}) \in \mathbb{R}^d,\ d << D$$
$$h(\mathbf{x}) = A\mathbf{x}$$

$$s.t.\ \text{minimize} \quad \sum_{k=1}^{N} ||\mathbf{x}_k - h(\mathbf{x}_k)||^2$$

Principal Component Analysis

Covariance matrix S

Mean $\mu_i = \frac{1}{N}\sum_{k=1}^{N} X_i(\mathbf{x}_k)$



$$S_{ij} = \frac{1}{N}\sum_{k=1}^{N}(X_i(\mathbf{x}_k) - \mu_i)(X_j(\mathbf{x}_k) - \mu_j)$$

symmetric $\Rightarrow$ can be diagonalized

$$S = U\Delta U' \quad \Delta = Diag(\lambda_1, \ldots \lambda_D)$$

Thm: Optimal projection in dimension d

projection on the first d eigenvectors of S

Let u_i the eigenvector associated to eigenvalue λ_i $\lambda_i > \lambda_{i+1}$

$$h : \mathbb{R}^D \mapsto \mathbb{R}^d, h(\mathbf{x}) = < \mathbf{x}, u_1 > u_1 + \ldots + < \mathbf{x}, u_d > u_d$$

Sketch of the proof

1. Maximize the variance of $h(\mathbf{x}) = A\mathbf{x}$

$\sum_k ||\mathbf{x}_k - h(\mathbf{x}_k)||^2 = \sum_k ||\mathbf{x}_k||^2 - \sum_k ||h(\mathbf{x}_k)||^2$

$$\text{Minimize } \sum_k ||\mathbf{x}_k - h(\mathbf{x}_k)||^2 \Rightarrow \text{ Maximize } \sum_k ||h(\mathbf{x}_k)||^2$$

$$Var(h(\mathbf{x})) = \frac{1}{N}\left(\sum_k ||h(\mathbf{x}_k)||^2 - ||\sum_k h(\mathbf{x}_k)||^2\right)$$

As

$$||\sum_k h(\mathbf{x}_k)||^2 = ||A\sum_k \mathbf{x}_k||^2 = N^2||A\mu||^2$$

where $\mu = (\mu_1, \ldots . \mu_D)$.
Assuming that $\mathbf{x}_k$ are centered ($\mu_i = 0$) gives the result.

Sketch of the proof, 2

2. Projection on eigenvectors u_i of S

Assume $h(\mathbf{x}) = A\mathbf{x} = \sum_{i=1}^{d} < \mathbf{x}, v_i > v_i$ and show $v_i = u_i$.

$$Var(AX) = (AX)(AX)' = A(XX')A' = ASA' = A(U\Delta U')A'$$

Consider $d = 1,\ v_1 = \sum w_i u_i$ $\qquad$ $\sum w_i^2 = 1$

remind $\lambda_i > \lambda_{i+1}$

$$Var(AX) = \sum \lambda_i w_i^2$$

maximized for $w_1 = 1, w_2 = \ldots = w_N = 0$

that is, $v_1 = u_i$.

Principal Component Analysis, Practicalities

Data preparation

- Mean centering the dataset

$$\begin{aligned} \mu_i &= \frac{1}{N}\sum_{k=1}^{N} X_i(\mathbf{x}_k) \\ \sigma_i &= \sqrt{\frac{1}{N}\sum_{k=1}^{N} X_i(\mathbf{x}_k)^2 - \mu_i^2} \\ z_k &= \left(\frac{1}{\sigma_i}(X_i(\mathbf{x}_k) - \mu_i)\right)_{i=1}^{D} \end{aligned}$$

Matrix operations

- Computing the covariance matrix

$$S_{ij} = \frac{1}{N}\sum_{k=1}^{N} X_i(z_k)X_j(z_k)$$

- Diagonalizing $S = U'\Delta U$ might be not affordable... Complexity $\mathcal{O}(D^3)$

Random projection

Random matrix

$$A : \mathbb{R}^D \mapsto \mathbb{R}^d \qquad A[d, D] \qquad A_{i,j} \sim \mathcal{N}(0, 1)$$

define

$$h(\mathbf{x}) = \frac{1}{\sqrt{d}} A\mathbf{x}$$

Property: h preserves the norm in expectation

$$E[||h(\mathbf{x})||^2] = ||\mathbf{x}||^2$$

With high probability $1 - 2exp\{-(\varepsilon^2 - \varepsilon^3)\frac{d}{4}\}$

$$(1 - \varepsilon)||\mathbf{x}||^2 \leq ||h(\mathbf{x})||^2 \leq (1 + \varepsilon)||\mathbf{x}||^2$$

Random projection

Proof

$$
\begin{aligned}
h(\mathbf{x}) = & \quad \frac{1}{\sqrt{d}} A\mathbf{x} \\
E(||h(\mathbf{x})||^2) = & \quad \frac{1}{d} E\left[\sum_{i=1}^{d}\left(\sum_{j=1}^{D} A_{i,j} X_j(\mathbf{x})\right)^2\right] \\
= & \quad \frac{1}{d}\sum_{i=1}^{d} E\left[\left(\sum_{j=1}^{D} A_{i,j} X_j(\mathbf{x})\right)^2\right] \\
= & \quad \frac{1}{d}\sum_{i=1}^{d}\sum_{j=1}^{D} E[A_{i,j}^2] E[X_j(\mathbf{x})^2] \\
= & \quad \frac{1}{d}\sum_{i=1}^{d}\sum_{j=1}^{D} \frac{||\mathbf{x}||^2}{D} \\
= & \quad ||\mathbf{x}||^2
\end{aligned}
$$

Random projection, 2

Johnson Lindenstrauss Lemma

For $d > \frac{9 \ln N}{\varepsilon^2 - \varepsilon^3}$, with high probability

$$(1-\varepsilon)||\mathbf{x}_i - \mathbf{x}_j||^2 \leq ||h(\mathbf{x}_i) - h(\mathbf{x}_j)||^2 \leq (1+\varepsilon)||\mathbf{x}_i - \mathbf{x}_j||^2$$

More:

`http://www.cs.yale.edu/clique/resources/RandomProjectionMethod.pdf`

Overview

Latent Semantic Analysis

1. Motivation
2. Algorithm
3. Discussion

Example

- c1: ***Human*** machine ***interface*** for ABC ***computer*** applications
- c2: A ***survey*** of ***user*** opinion of ***computer system response time***
- c3: The ***EPS user interface*** management ***system***
- c4: ***System*** and ***human system*** engineering testing of ***EPS***
- c5: Relation of ***user*** perceived ***response time*** to error measurement

- m1: The generation of random, binary, ordered ***trees***
- m2: The intersection ***graph*** of paths in ***trees***
- m3: ***Graph minors*** IV: Widths of ***trees*** and well-quasi-ordering
- m4: ***Graph minors***: A ***survey***

Example, cont

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

LSA, 2

Motivations

- Context : bag of words
- Curse of dimensionality $\mathbb{R}^D$
- Synonymy / Polysemy

Goals

- Dimensionality reduction $\mathbb{R}^d$
- A good topology (distance, similarity)

Remark

- First solution: cosine similarity
- Why not ?

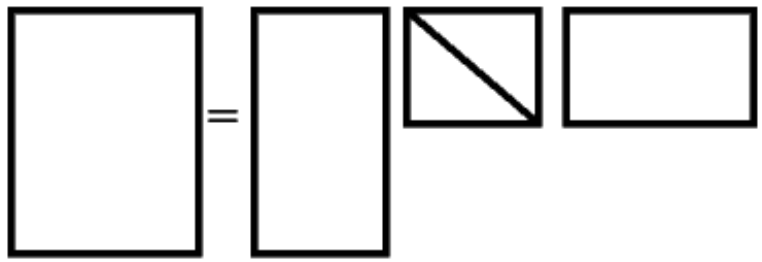
More

`http://lsa.colorado.edu`

LSA, 3

Input

Matrix X = words $\times$ documents



Principle

1. Change of coordinates from words and documents to concepts
2. Dimensionality reduction

Difference with Principal Component Analysis

LSA $\equiv$ Singular Value Decomposition

Input

Matrix X = words $\times$ documents $m \times d$

$$X = U' \; S \; V$$

with • U: change of word basis $m \times r$

• V: change of document basis $r \times d$

• S: diagonal matrix $r \times r$

Dimensionality reduction

• S Order by decreasing eigenvalue

• $S' = S$ cancel out all eigenvalues but the first (300) ones.

$$X' = U'S'V$$

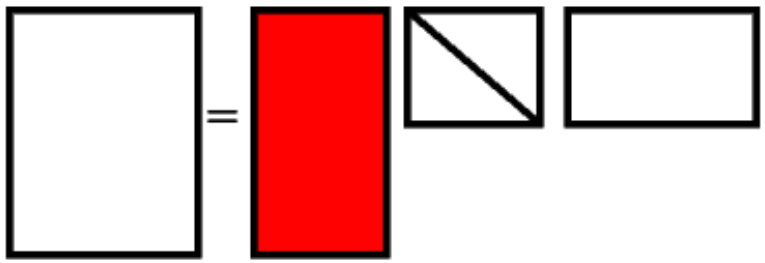
Intuition

$$X = \begin{pmatrix} & m_1 & m_2 & m_3 & m_4 \\ d_1 & 0 & 1 & 1 & 1 \\ d_2 & 1 & 1 & 1 & 0 \end{pmatrix}$$

m_1 and m_4 are not present in the same documents, but are together with same words; "hence" they are somewhat related'... After SVD + Reduction,

$$X = \begin{pmatrix} & m_1 & m_2 & m_3 & m_4 \\ d_1 & \epsilon & 1 & 1 & 1 \\ d_2 & 1 & 1 & 1 & \epsilon \end{pmatrix}$$

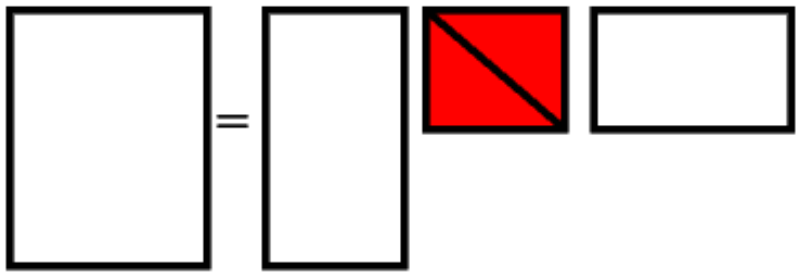
Algorithm



Singular value Decomposition of the words by contexts matrix

0.22	-0.11	0.29	-0.41	-0.11	-0.34	0.52	-0.06	-0.41
0.20	-0.07	0.14	-0.55	0.28	0.50	-0.07	-0.01	-0.11
0.24	0.04	-0.16	-0.59	-0.11	-0.25	-0.30	0.06	0.49
0.40	0.06	-0.34	0.10	0.33	0.38	0.00	0.00	0.01
0.64	-0.17	0.36	0.33	-0.16	-0.21	-0.17	0.03	0.27
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.30	-0.14	0.33	0.19	0.11	0.27	0.03	-0.02	-0.17
0.21	0.27	-0.18	-0.03	-0.54	0.08	-0.47	-0.04	-0.58
0.01	0.49	0.23	0.03	0.59	-0.39	-0.29	0.25	-0.23
0.04	0.62	0.22	0.00	-0.07	0.11	0.16	-0.68	0.23
0.03	0.45	0.14	-0.01	-0.30	0.28	0.34	0.68	0.18

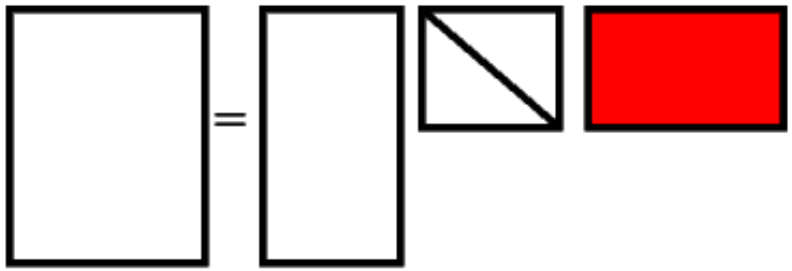
Algorithm, 2



Singular value Decomposition of the words by contexts matrix

3.34
2.54
2.35
1.64
1.50
1.31
0.85
0.56
0.36

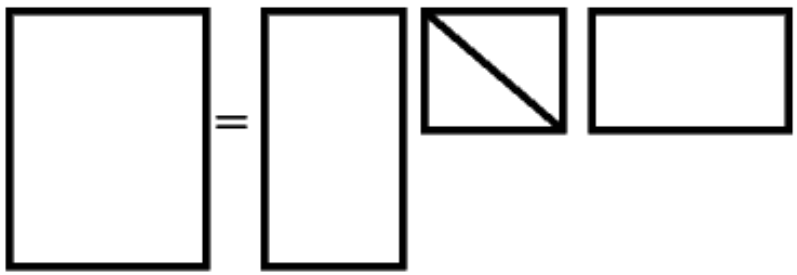
Algorithm, 3



Singular value Decomposition of the words by contexts matrix

0.20	0.61	0.46	0.54	0.28	0.00	0.01	0.02	0.08
-0.06	0.17	-0.13	-0.23	0.11	0.19	0.44	0.62	0.53
0.11	-0.50	0.21	0.57	-0.51	0.10	0.19	0.25	0.08
-0.95	-0.03	0.04	0.27	0.15	0.02	0.02	0.01	-0.03
0.05	-0.21	0.38	-0.21	0.33	0.39	0.35	0.15	-0.60
-0.08	-0.26	0.72	-0.37	0.03	-0.30	-0.21	0.00	0.36
0.18	-0.43	-0.24	0.26	0.67	-0.34	-0.15	0.25	0.04
-0.01	0.05	0.01	-0.02	-0.06	0.45	-0.76	0.45	-0.07
-0.06	0.24	0.02	-0.08	-0.26	-0.62	0.02	0.52	-0.45

Algorithm, 4

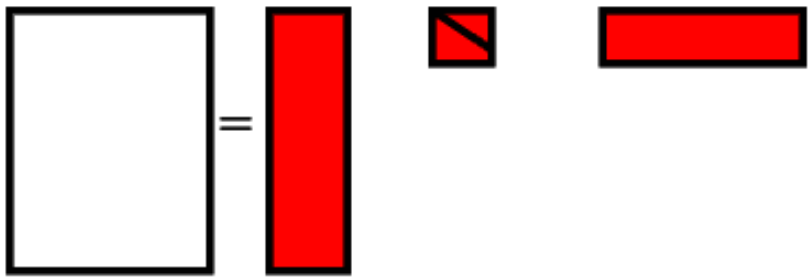


Singular value Decomposition of the words by contexts matrix

3.34

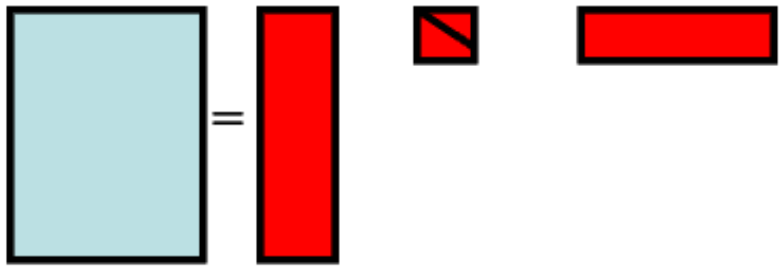
2.54

Algorithm, 5



Singular value
Decomposition of the
words by contexts matrix

Algorithm, 6



Singular value Decomposition of the words by contexts matrix

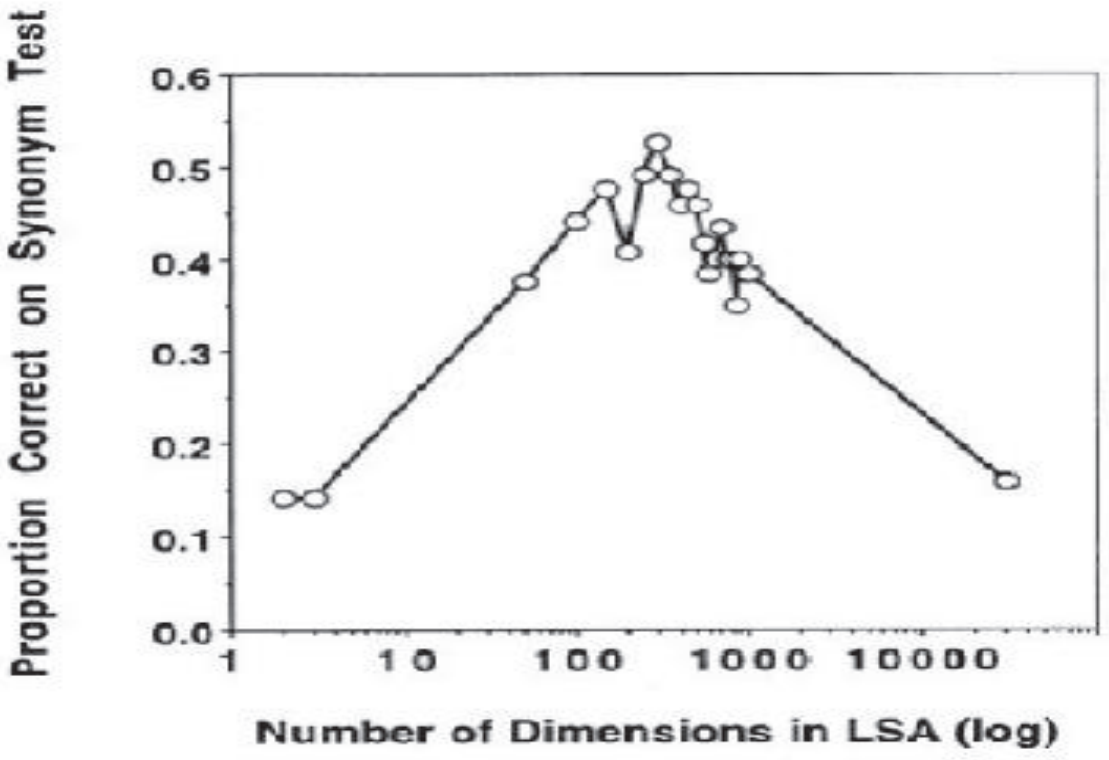
	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

Discussion

An application

Synonymy test TOEFL



Setting the number of dimensions

Trial and error :-(

Remarks

Negation apparently does not matter

More: Google hits P. Turney

Some applications

- Educational Text Selection
- Essay Scoring
- Summary Scoring & Revision
- Cross Language Retrieval

LSA – Principal Component Analysis

Similarities

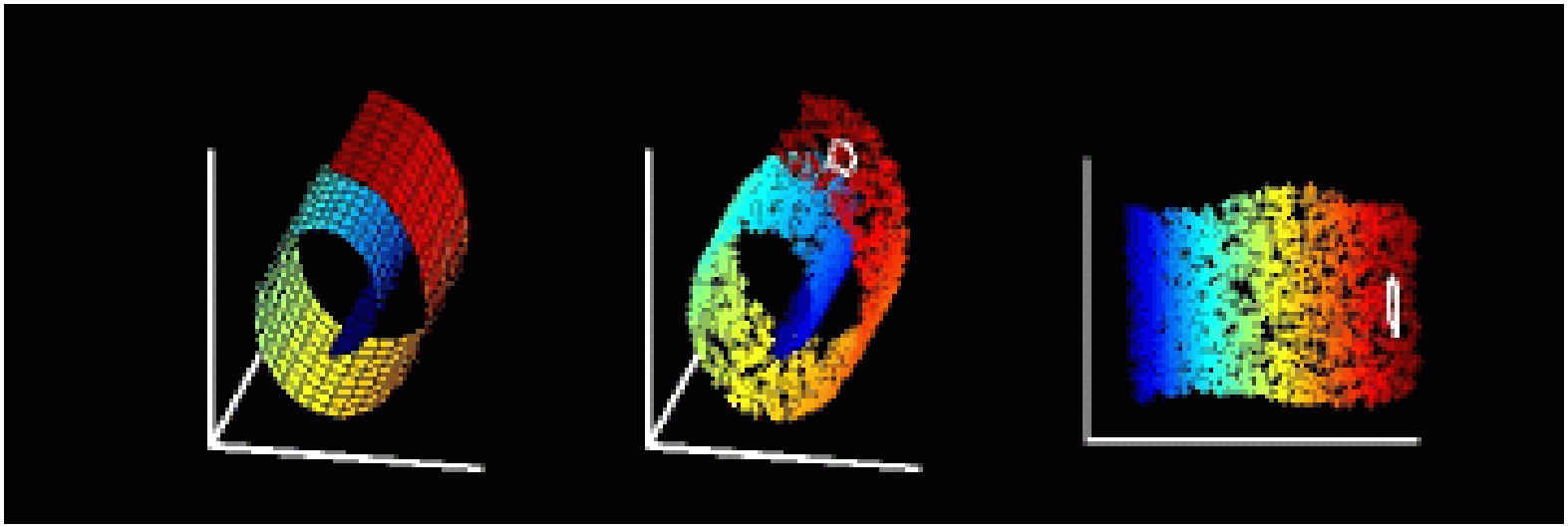
- Input: matrix
- Diagonalizing
- Cancel all eigenvalues but the highest ones
- Projection on the corresponding eigenvectors

Differences

	ACP	LSA
Matrix	covariance attributs	words $\times$ documents
d	2-3	100-300

Overview

Non-Linear Dimensionality Reduction



Conjecture

Examples live in a manifold of dimension $d << D$

Goal: consistent projection of the dataset onto $\mathbb{R}^d$

Consistency:

- Preserve the structure of the data
- e.g. preserve the distances between points

Multi-Dimensional Scaling

Position of the problem

- Given $\{\mathbf{x}_1, \ldots, \mathbf{x}_N,\ \mathbf{x}_i \in \mathbb{R}^D\}$
- Given $sim(\mathbf{x}_i, \mathbf{x}_j) \in \mathbb{R}^+$
- Find projection Φ onto $\mathbb{R}^d$

$$\begin{array}{rl} x \in \mathbb{R}^D \rightarrow & \Phi(x) \in \mathbb{R}^d \\ sim(\mathbf{x}_i, \mathbf{x}_j) \sim & sim(\Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)) \end{array}$$

Optimisation

Define $X,\ X_{i,j} = sim(\mathbf{x}_i, \mathbf{x}_j)$; $X^\Phi,\ X^\Phi_{i,j} = sim(\Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j))$

Find Φ minimizing $||X - X'||$

Rq : Linear Φ = Principal Component Analysis

But linear MDS does not work: preserves all distances, while

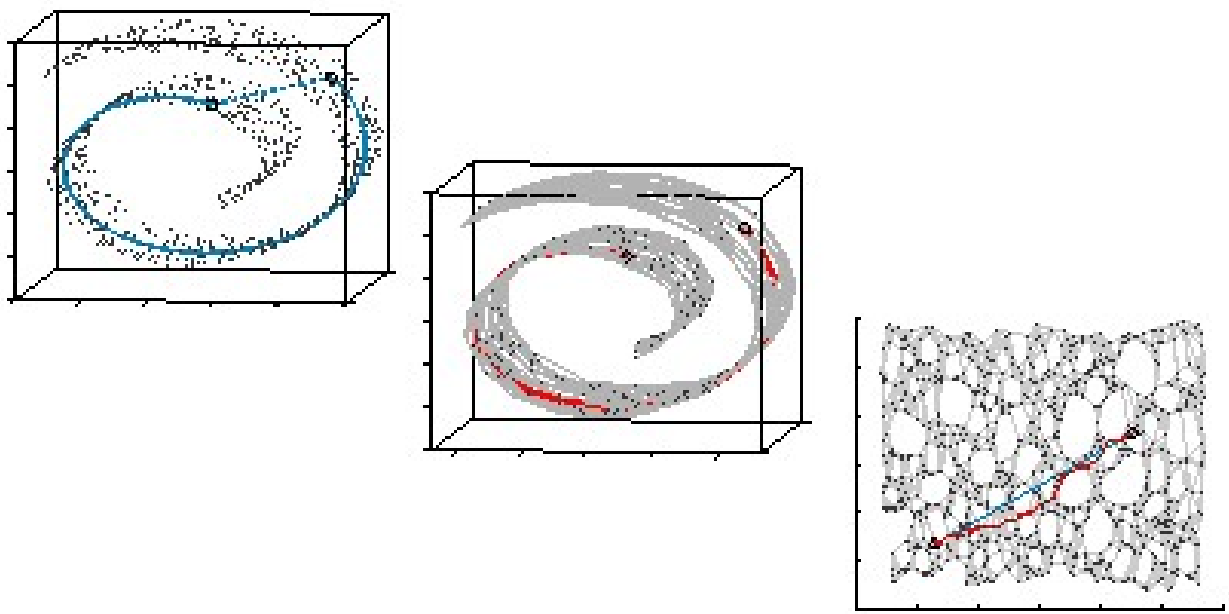
only local distances are meaningful

Non-linear projections

Approaches

- Reconstruct global structures from local ones and find global projection — **Isomap**
- Only consider local structures — **LLE**

Intuition: locally, points live in $\mathbb{R}^d$



Isomap

Tenenbaum, da Silva, Langford 2000

http://isomap.stanford.edu

Estimate $d(x_i, x_j)$

- Known if $\mathbf{x}_i$ and $\mathbf{x}_j$ are close
- Otherwise, compute the shortest path between $\mathbf{x}_i$ and $\mathbf{x}_j$

geodesic distance (dynamic programming)

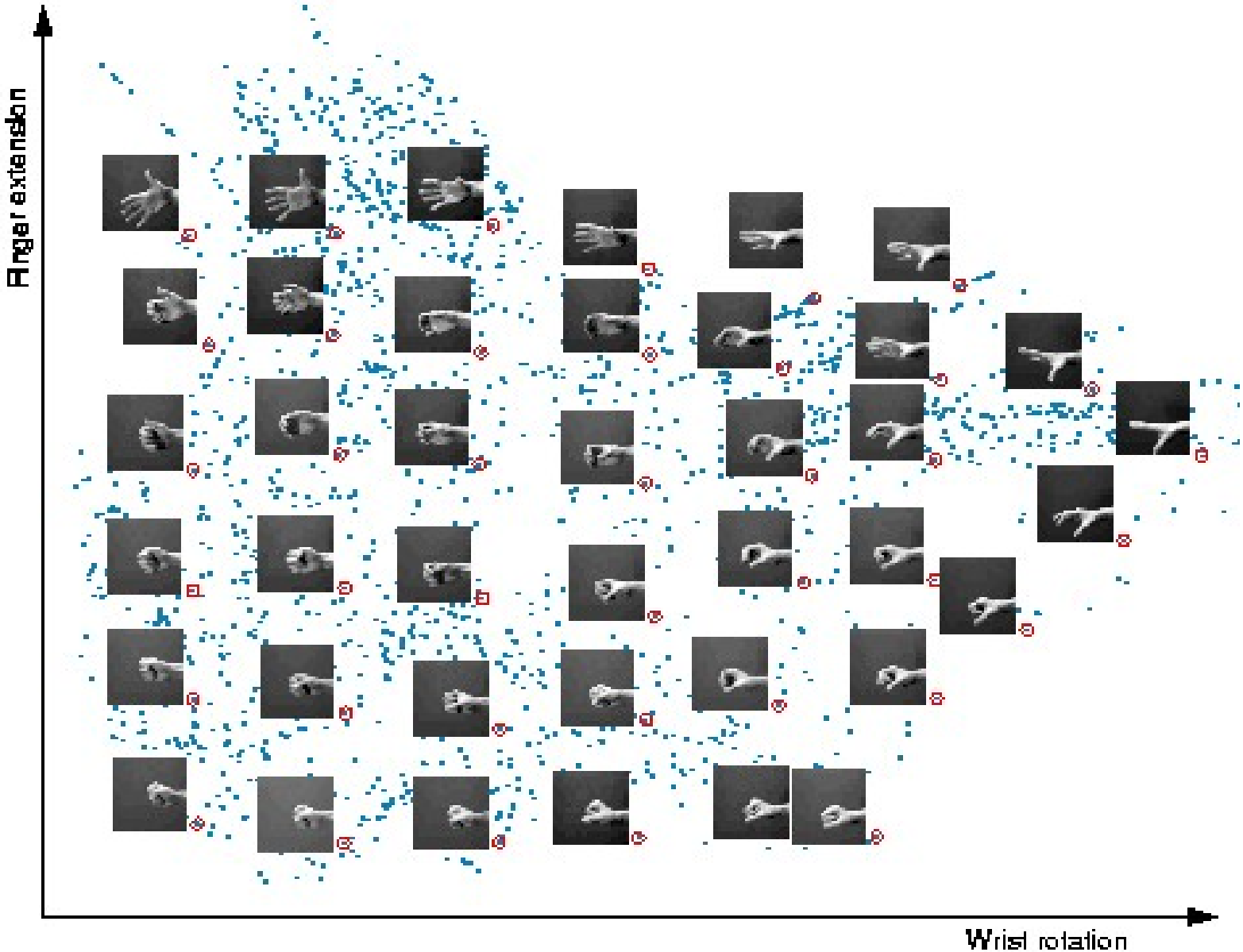
Requisite

If data points sampled in a convex subset of $\mathbb{R}^d$,
then geodesic distance $\sim$ Euclidean distance on $\mathbb{R}^d$.

General case

- Given $d(\mathbf{x}_i, \mathbf{x}_j)$, estimate $< \mathbf{x}_i, \mathbf{x}_j >$
- Project points in $\mathbb{R}^d$

Isomap, 2



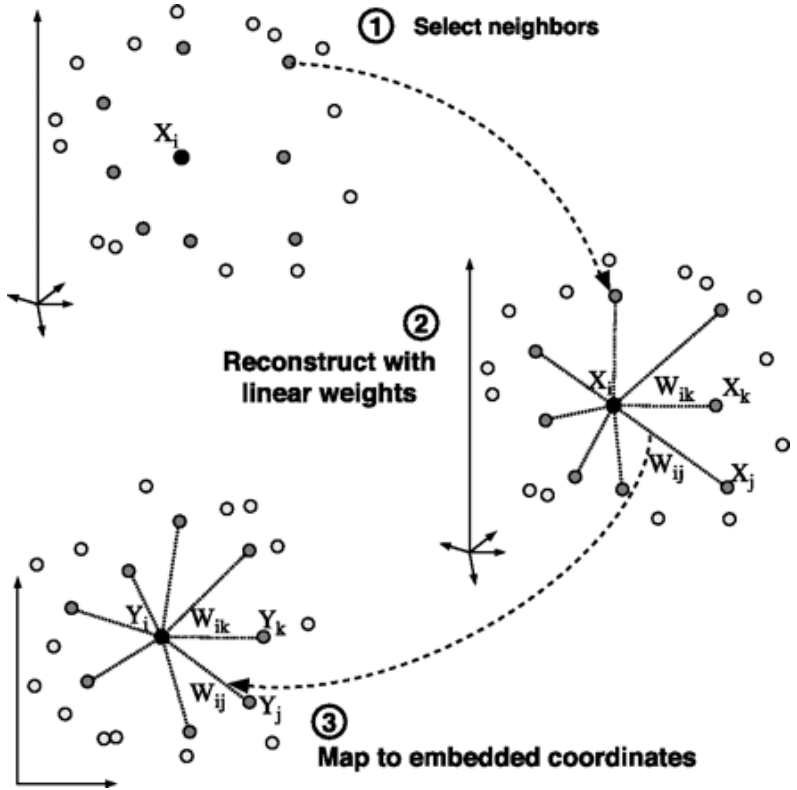
Locally Linear Embedding

Roweiss and Saul, 2000

http://www.cs.toronto.edu/~roweis/lle/

Principle

- Find local description for each point: depending on its neighbors



Local Linear Embedding, 2

Find neighbors

For each $\mathbf{x}_i$, find its nearest neighbors $\mathcal{N}(i)$

Parameter: number of neighbors

Change of representation

Goal Characterize $\mathbf{x}_i$ wrt its neighbors:

$$\mathbf{x}_i = \sum_{j \in \mathcal{N}(i)} w_{i,j} \mathbf{x}_j \quad \text{with} \quad \sum_{j \in \mathcal{N}(i)} w_{ij} = 1$$

Property: invariance by translation, rotation, homothety

How Compute the local covariance matrix:

$$C_{j,k} = < x_j - x_i, x_k - x_i >$$

Find vector w_i s.t. $Cw_i = 1$

Local Linear Embedding, 3

Algorithm

Local description: Matrix W such that $\sum_j w_{i,j} = 1$

$$W = argmin\{\sum_{i=1}^{N} ||\mathbf{x}_i - \sum_j w_{i,j}\mathbf{x}_j||^2\}$$

Projection: Find $\{z_1, \ldots, z_n\}$ *in* $\mathbb{R}^d$ minimizing

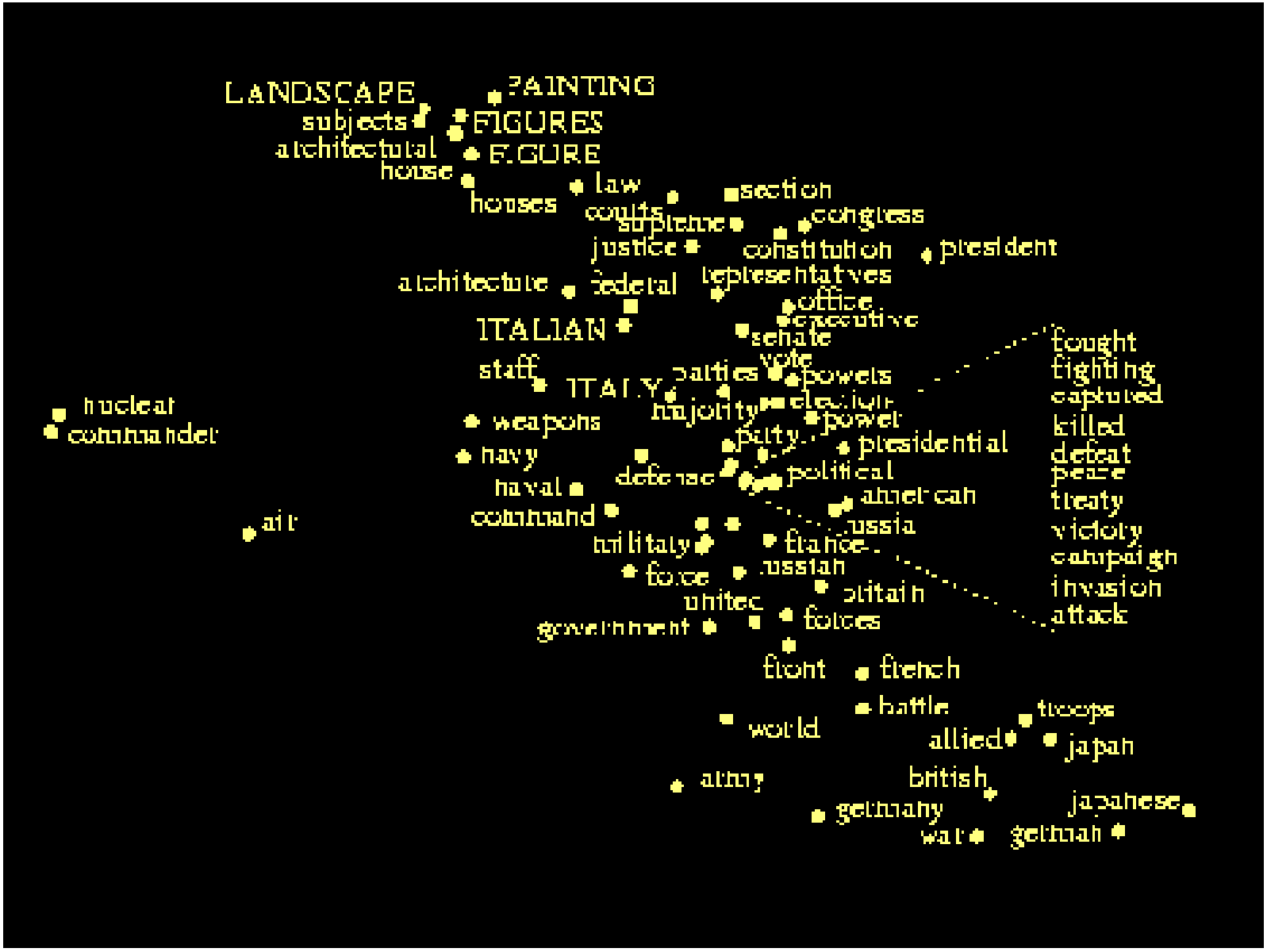
$$\sum_{i=1}^{N} ||z_i - \sum_j w_{i,j} z_j||^2$$

$$Minimize\ ((I - W)Z)'((I - W)Z) = Z'(I - W)'(I - W)Z$$

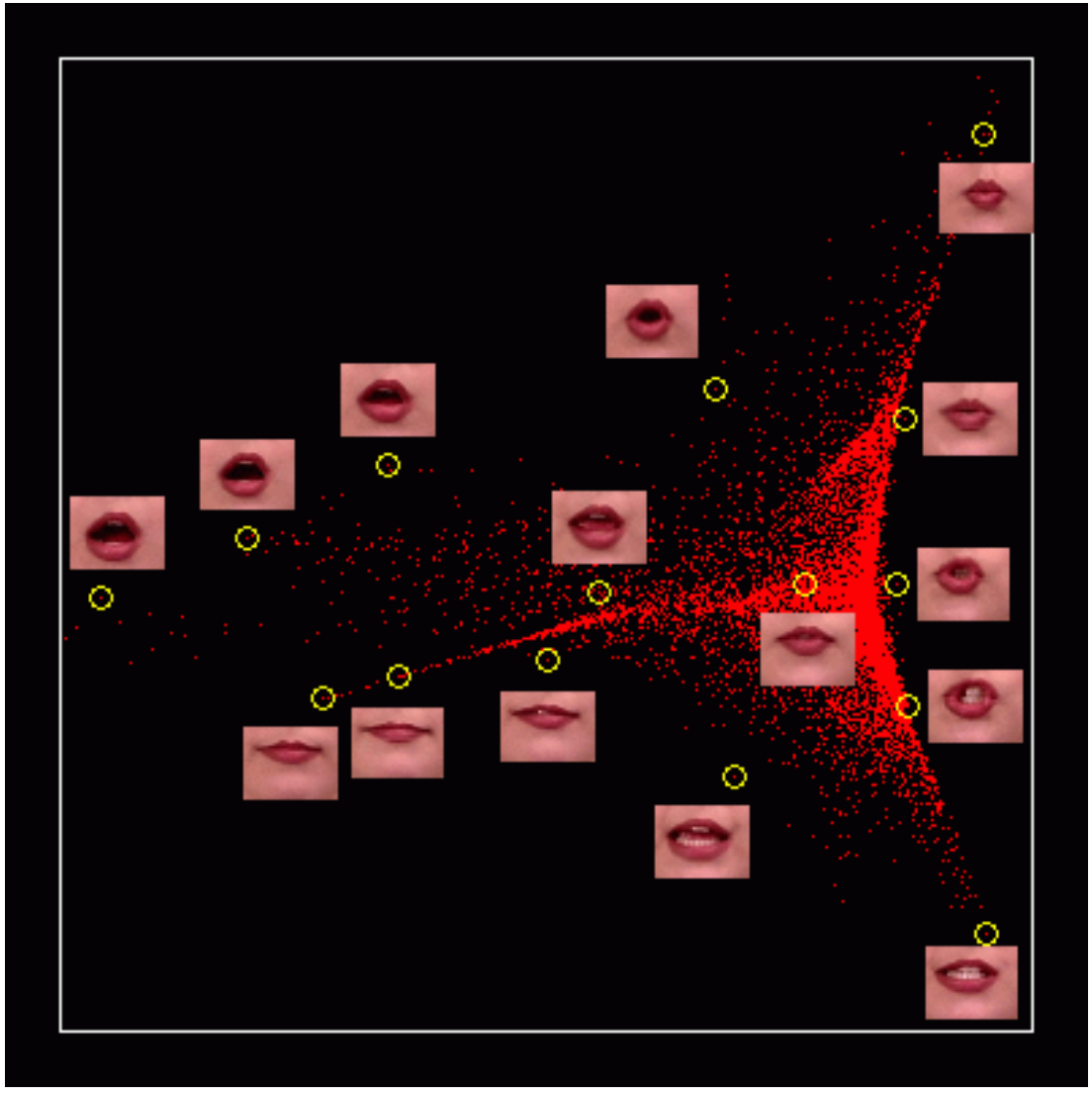
Solutions: vectors z_i are eigenvectors of $(I - W)'(I - W)$

- Keeping the d eigenvectors with lowest eigenvalues > 0

Example, Texts



Example, Images



LLE

Overview

Feature Selection as a one-player game

Romaric Gaudel[1,2,3] and **Michèle Sebag**[1,2,3]

[1] Univ. Paris-Sud, LRI, UMR8623
[2] CNRS [3] INRIA-Saclay

ICML, June 2010

Feature Selection

- Optimization problem

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

$\mathcal{F}$: Set of features

F: Feature subset

D: 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

Feature Selection: state of the art / drawbacks

- Filter approaches [1]
  - ★ No account for all feature interdependencies
- Wrapper approaches
  - ▸ Tackling combinatorial optimization [2,3,4]
    - ★ Tractability vs. exhaustivity tradeoff
- 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 *A practical approach to feature selection*. ML'92

[2] D. Margaritis *Toward provably correct Feature Selection in arbitrary domains*. NIPS'09

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

[4] M. Boullé *Compression-based averaging of selective Naive Bayes classifiers*. J. Mach. Learn. Res. 07

[5] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik *Gene selection for cancer classification using Support Vector Machines*. Mach. Learn. 2002

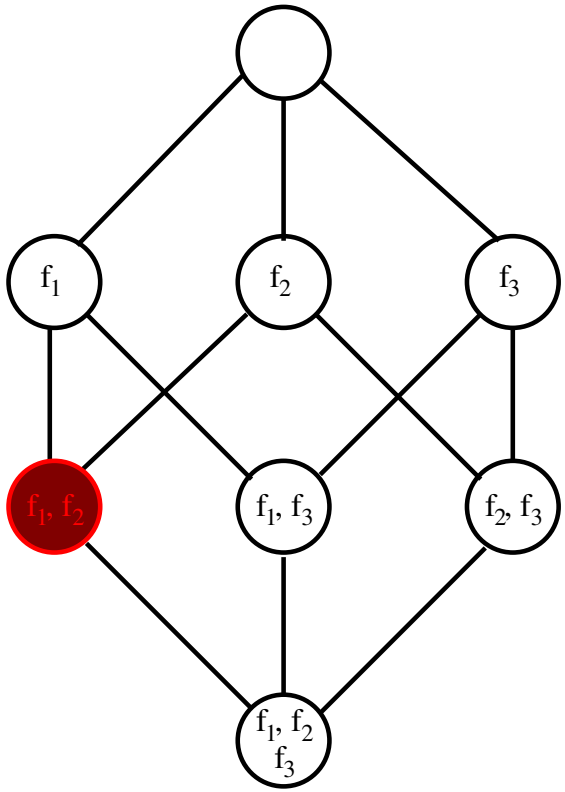
[6] J. Rogers, and S. R. Gunn *Identifying feature relevance using a Random Forest*. SLSFS'05

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

[8] F. Bach *Exploring large feature spaces with hierarchical Multiple Kernel Learning*. NIPS'08

The one-player game approach

- Goal
  - Find $\underset{F \subseteq \mathcal{F}}{argmin}$ **Err** $(\mathcal{A}(F, D))$
- Exploration vs Exploitation tradeoff
  - Virtually explore the whole lattice
  - Gradually focus the search on most promising Fs
  - Use a frugal, unbiased assessment of F
- How? tractability vs. optimality tradeoff
  - Upper Confidence Tree (UCT) [1]
    - ★ UCT $\subset$ Monte-Carlo Tree Search
    - ★ UCT tackles tree-structured optimization problems



[1] L. Kocsis, and C. Szepesvári *Bandit based Monte-Carlo planning*. ECML'06

Outline

Feature Selection as a one-player game

A Markov Decision Process

Set of features $\mathcal{F}$

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

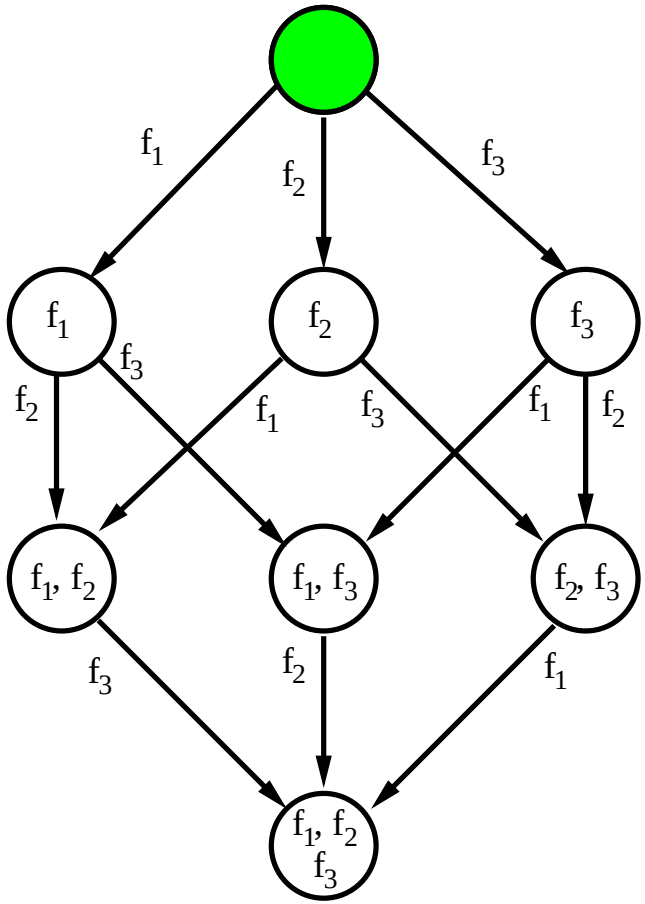
Initial state $\varnothing$

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

Final state any state

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

- Ideally : $V(F) = \textbf{Err}(\mathcal{A}(F, D))$
- In practice: Fast unbiased estimate of $\textbf{Err}(\mathcal{A}(F, D))$



Optimal Policy

Policy $\pi : \mathcal{S} \rightarrow A$

Final state following a policy F_π

Optimal policy

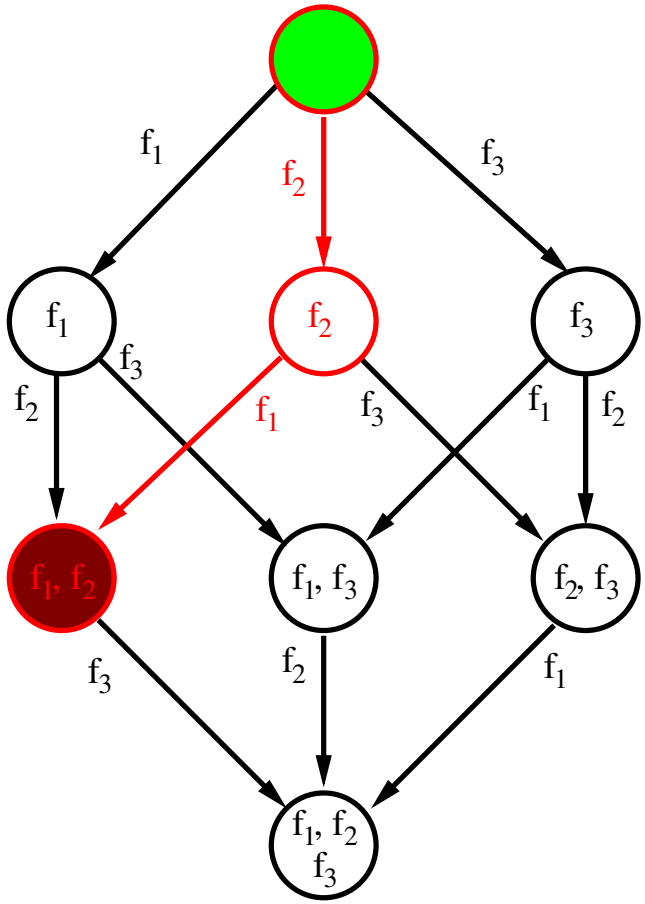
$$\pi^\star = \underset{\pi}{argmin}\ \mathbf{Err}\left(\mathcal{A}\left(F_\pi, D\right)\right)$$

Bellman's optimality principle

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

with

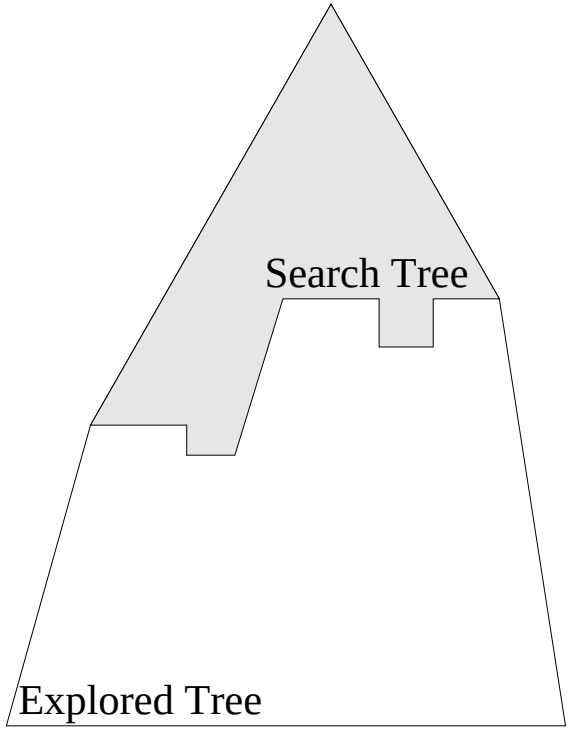
$$V^\star(F) = \begin{cases} \mathbf{Err}(\mathcal{A}(F)) & \text{if } final(F) \\ \underset{f \in \mathcal{F}}{min}\ V^\star(F \cup \{f\}) & \text{otherwise} \end{cases}$$



$\pi^\star$ intractable $\Rightarrow$ approximation using UCT

Upper Confidence Tree

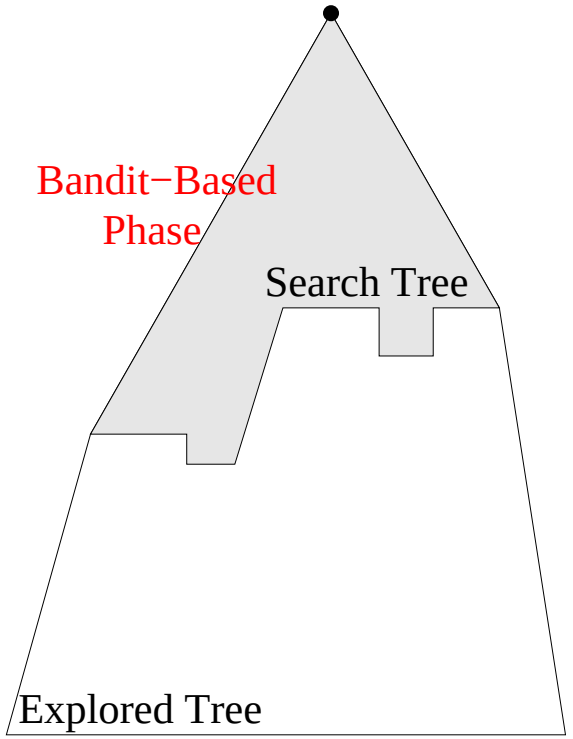
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - ★ Select next action (bandit-based phase)
    - ★ Add a node (leaf of the search tree)
    - ★ Select next action bis (random phase)
    - ★ Compute instant reward
    - ★ Update information in visited nodes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



[1] L. Kocsis, and C. Szepesvári *Bandit based Monte-Carlo planning*. ECML'06

Upper Confidence Tree

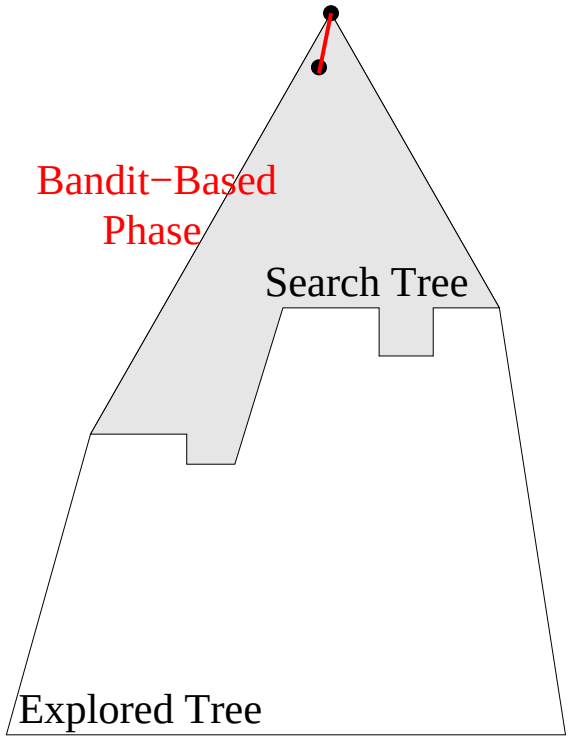
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - ★ Select next action (bandit-based phase)
    - ★ Add a node (leaf of the search tree)
    - ★ Select next action bis (random phase)
    - ★ Compute instant reward
    - ★ Update information in visited nodes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



[1] L. Kocsis, and C. Szepesvári *Bandit based Monte-Carlo planning*. ECML'06

Upper Confidence Tree

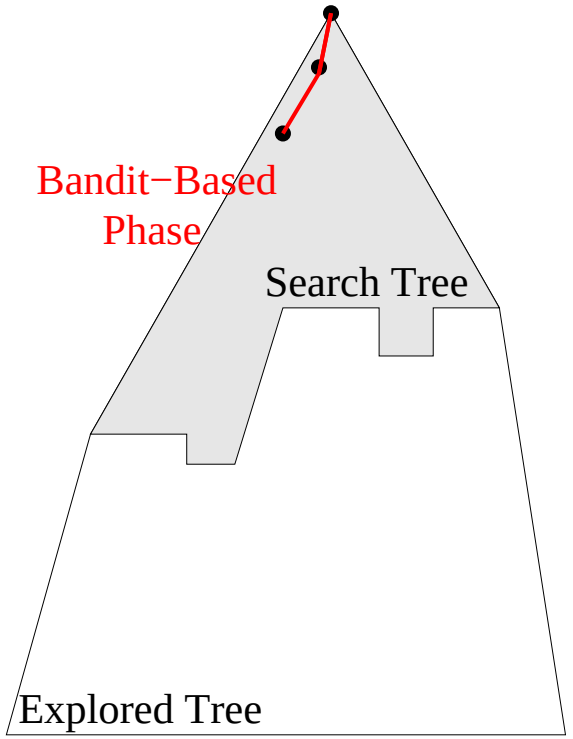
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - ★ Select next action (bandit-based phase)
    - ★ Add a node (leaf of the search tree)
    - ★ Select next action bis (random phase)
    - ★ Compute instant reward
    - ★ Update information in visited nodes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



[1] L. Kocsis, and C. Szepesvári *Bandit based Monte-Carlo planning*. ECML'06

Upper Confidence Tree

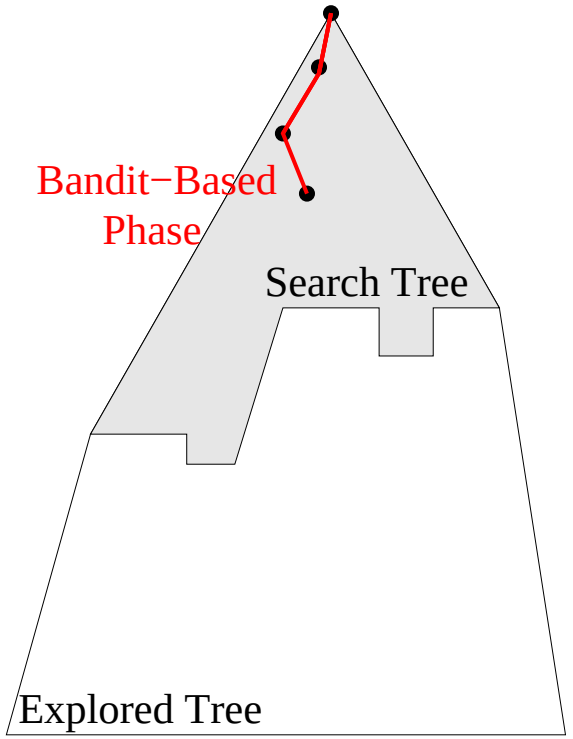
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - ★ Select next action (bandit-based phase)
    - ★ Add a node (leaf of the search tree)
    - ★ Select next action bis (random phase)
    - ★ Compute instant reward
    - ★ Update information in visited nodes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



[1] L. Kocsis, and C. Szepesvári *Bandit based Monte-Carlo planning*. ECML'06

Upper Confidence Tree

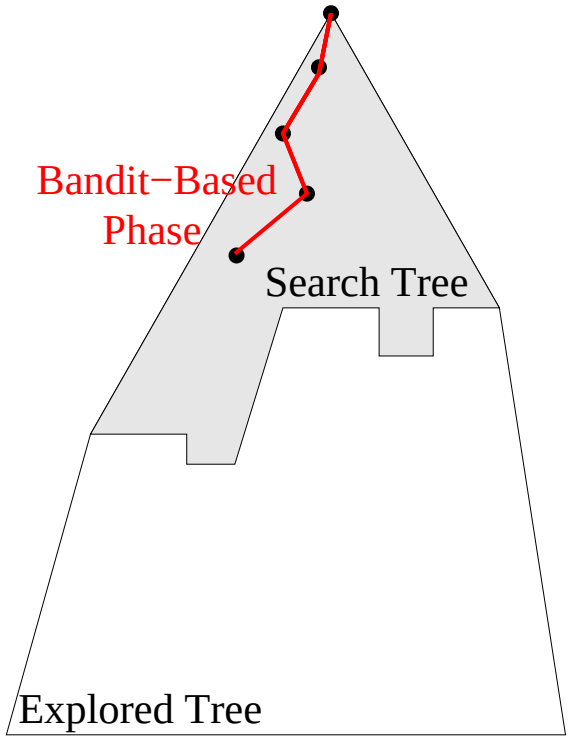
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - ★ Select next action (bandit-based phase)
    - ★ Add a node (leaf of the search tree)
    - ★ Select next action bis (random phase)
    - ★ Compute instant reward
    - ★ Update information in visited nodes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



[1] L. Kocsis, and C. Szepesvári *Bandit based Monte-Carlo planning*. ECML'06

Upper Confidence Tree

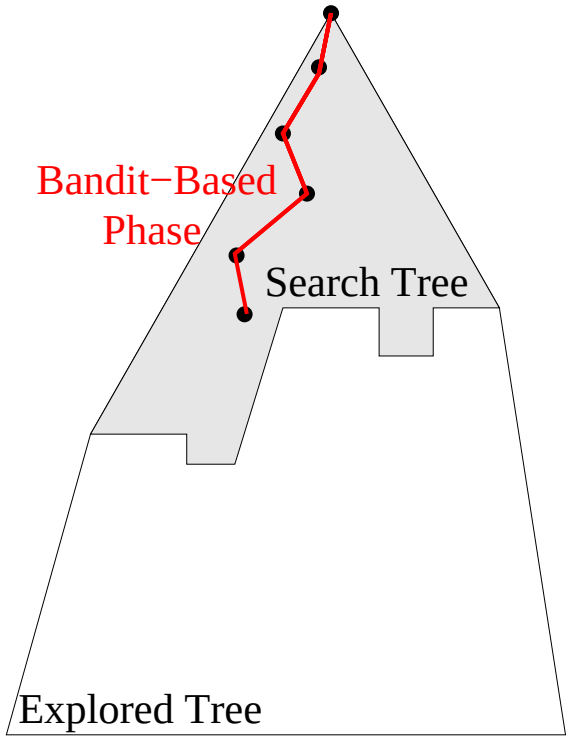
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - ★ Select next action (bandit-based phase)
    - ★ Add a node (leaf of the search tree)
    - ★ Select next action bis (random phase)
    - ★ Compute instant reward
    - ★ Update information in visited nodes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



[1] L. Kocsis, and C. Szepesvári *Bandit based Monte-Carlo planning*. ECML'06

Upper Confidence Tree

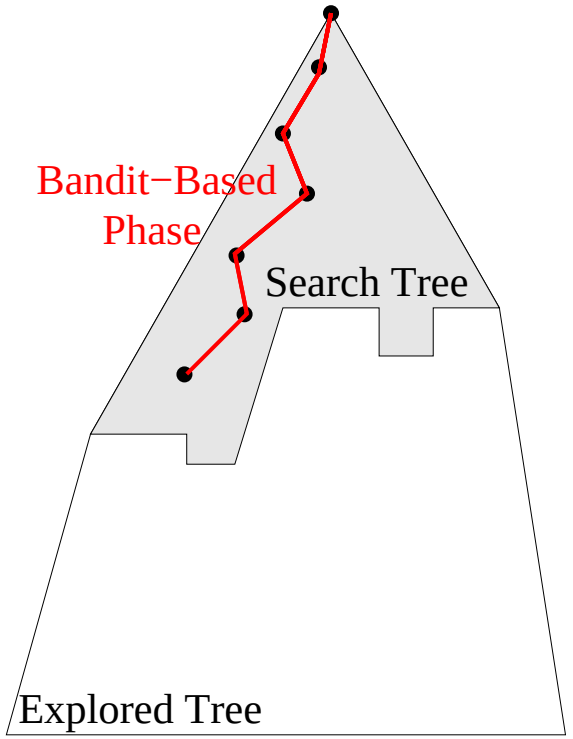
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - ★ Select next action (bandit-based phase)
    - ★ Add a node (leaf of the search tree)
    - ★ Select next action bis (random phase)
    - ★ Compute instant reward
    - ★ Update information in visited nodes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



[1] L. Kocsis, and C. Szepesvári *Bandit based Monte-Carlo planning*. ECML'06

Upper Confidence Tree

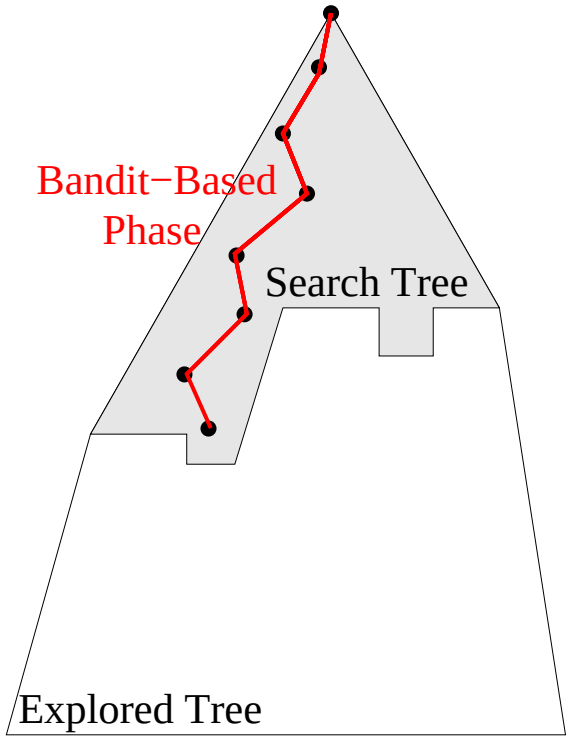
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - ★ Select next action (bandit-based phase)
    - ★ Add a node (leaf of the search tree)
    - ★ Select next action bis (random phase)
    - ★ Compute instant reward
    - ★ Update information in visited nodes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



[1] L. Kocsis, and C. Szepesvári *Bandit based Monte-Carlo planning*. ECML'06

Upper Confidence Tree

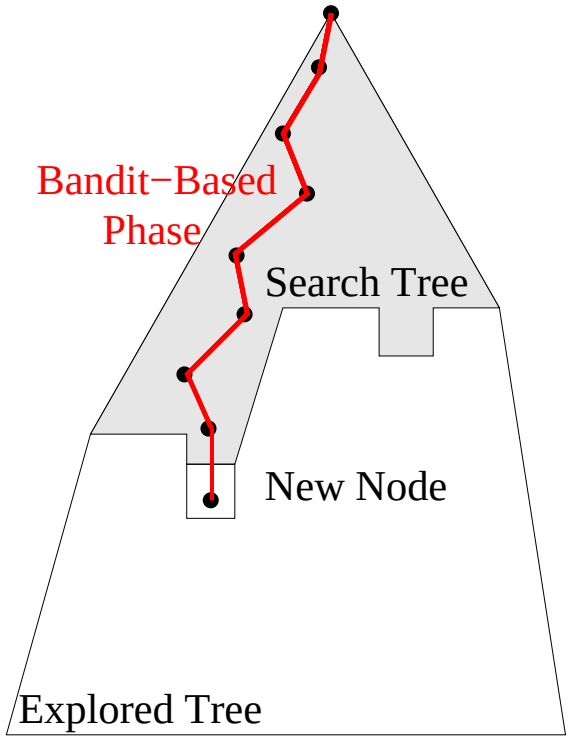
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - ★ Select next action (bandit-based phase)
    - ★ Add a node (leaf of the search tree)
    - ★ Select next action bis (random phase)
    - ★ Compute instant reward
    - ★ Update information in visited nodes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



[1] L. Kocsis, and C. Szepesvári *Bandit based Monte-Carlo planning*. ECML'06

Upper Confidence Tree

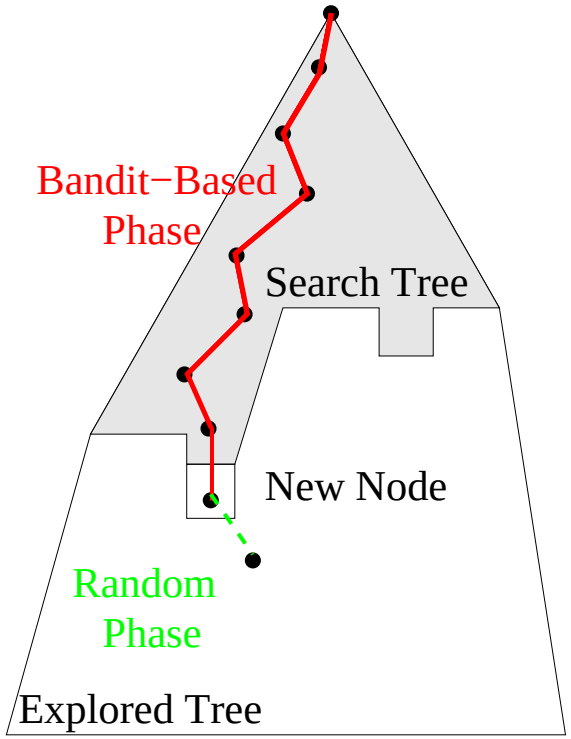
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - ★ Select next action (bandit-based phase)
    - ★ Add a node (leaf of the search tree)
    - ★ Select next action bis (random phase)
    - ★ Compute instant reward
    - ★ Update information in visited nodes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



[1] L. Kocsis, and C. Szepesvári *Bandit based Monte-Carlo planning*. ECML'06

Upper Confidence Tree

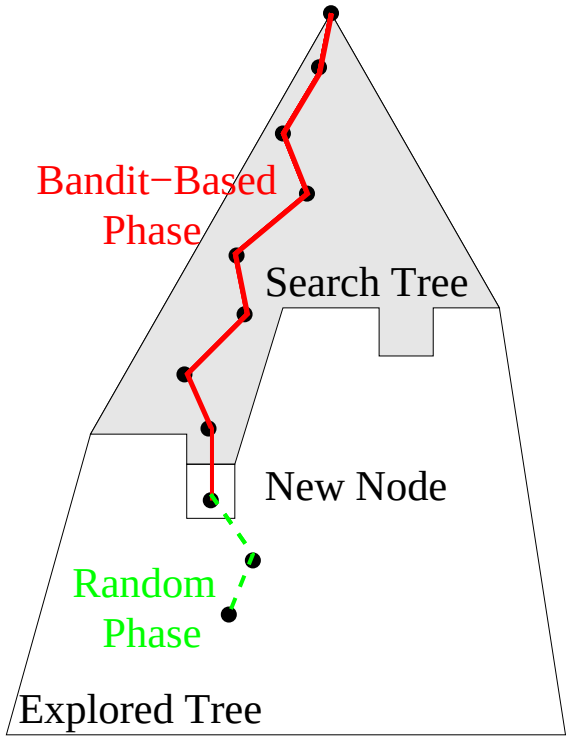
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - ★ Select next action (bandit-based phase)
    - ★ Add a node (leaf of the search tree)
    - ★ Select next action bis (random phase)
    - ★ Compute instant reward
    - ★ Update information in visited nodes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



[1] L. Kocsis, and C. Szepesvári *Bandit based Monte-Carlo planning*. ECML'06

Upper Confidence Tree

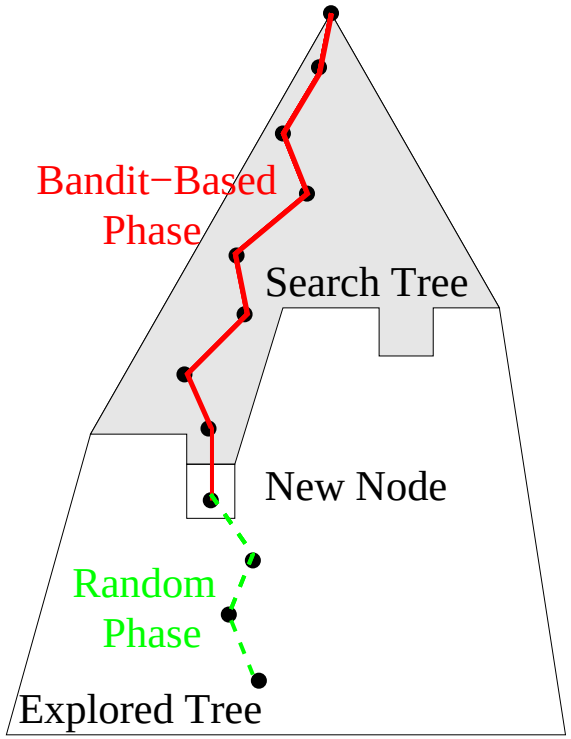
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - ★ Select next action (bandit-based phase)
    - ★ Add a node (leaf of the search tree)
    - ★ Select next action bis (random phase)
    - ★ Compute instant reward
    - ★ Update information in visited nodes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



[1] L. Kocsis, and C. Szepesvári *Bandit based Monte-Carlo planning*. ECML'06

Upper Confidence Tree

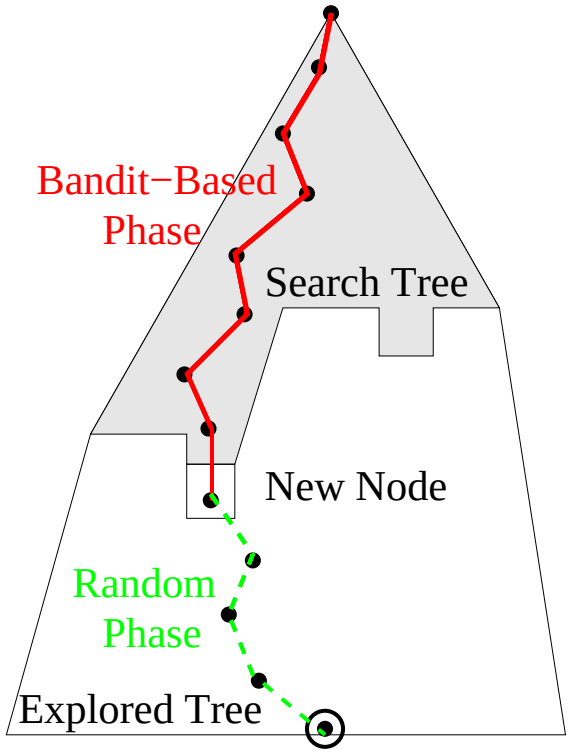
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - ★ Select next action (bandit-based phase)
    - ★ Add a node (leaf of the search tree)
    - ★ Select next action bis (random phase)
    - ★ Compute instant reward
    - ★ Update information in visited nodes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



[1] L. Kocsis, and C. Szepesvári *Bandit based Monte-Carlo planning*. ECML'06

Upper Confidence Tree

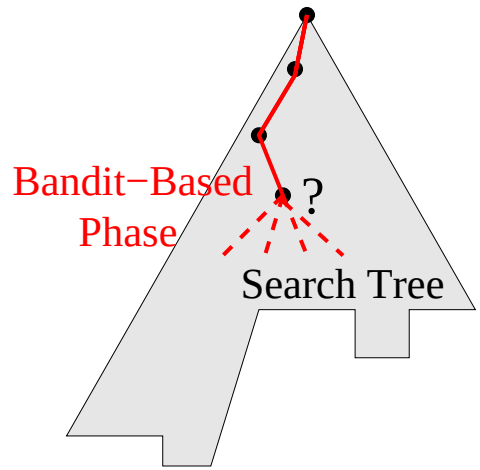
- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - ★ Select next action (bandit-based phase)
    - ★ Add a node (leaf of the search tree)
    - ★ Select next action bis (random phase)
    - ★ Compute instant reward
    - ★ Update information in visited nodes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



[1] L. Kocsis, and C. Szepesvári *Bandit based Monte-Carlo planning*. ECML'06

Multi-Arm Bandit-based phase

- Upper Confidence Bound (UCB1-tuned) [1]
  - Select $\underset{a \in A}{\text{argmax}}\ \hat{\mu}_a + \sqrt{\frac{c_e \log(T)}{t_a} min\left(\frac{1}{4}, \hat{\sigma}_a^2 + \sqrt{\frac{c_e \log(T)}{t_a}}\right)}$



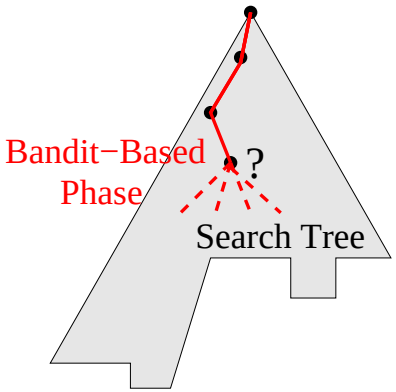
- ★ T: Total number of trials in current node
- ★ t_a: Number of trials for action a
- ★ $\hat{\mu}_a$: Empirical average reward for action a
- ★ $\hat{\sigma}_a^2$: Empirical variance of reward for action a

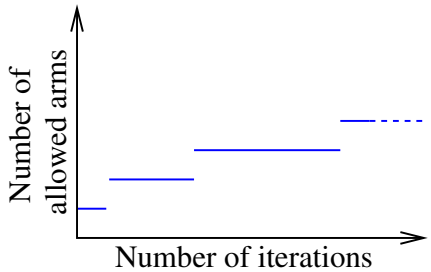
[1] P. Auer, N. Cesa-Bianchi, and P. Fischer *Finite-time analysis of the Multiarmed Bandit Problem*. ML'02

FUSE: bandit-based phase

Dealing with many arms

- Bottleneck
  - A many-armed problem (hundreds of features)
    - $\Rightarrow$ need to guide UCT
- Ingredient 1: controlling 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





[1] S. Gelly, and D. Silver *Combining online and offline knowledge in UCT*. ICML'07

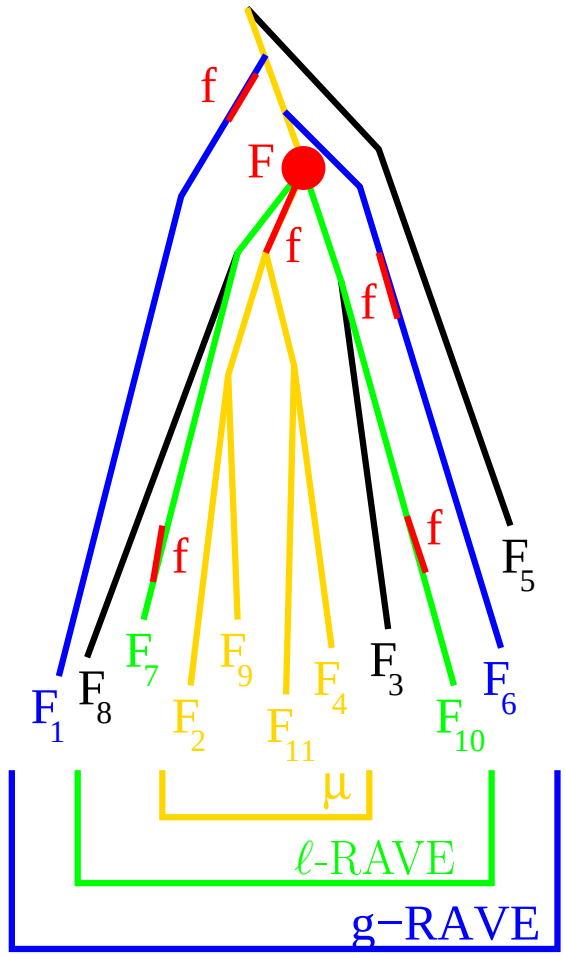
[2] R. Coulom *Efficient selectivity and backup operators in Monte-Carlo tree search*. Computer and Games 2006

[3] P. Rolet, M. Sebag, and O. Teytaud *Boosting Active Learning to optimality: a tractable Monte-Carlo, Billiard-based algorithm*. ECML'09

FUSE: bandit-based phase

Sharing information among nodes

- Ingredient 2: sharing information among nodes
  - Rapid Action Value Estimation (RAVE) [1]
    - ★ RAVE(f) = average reward when $f \in F$

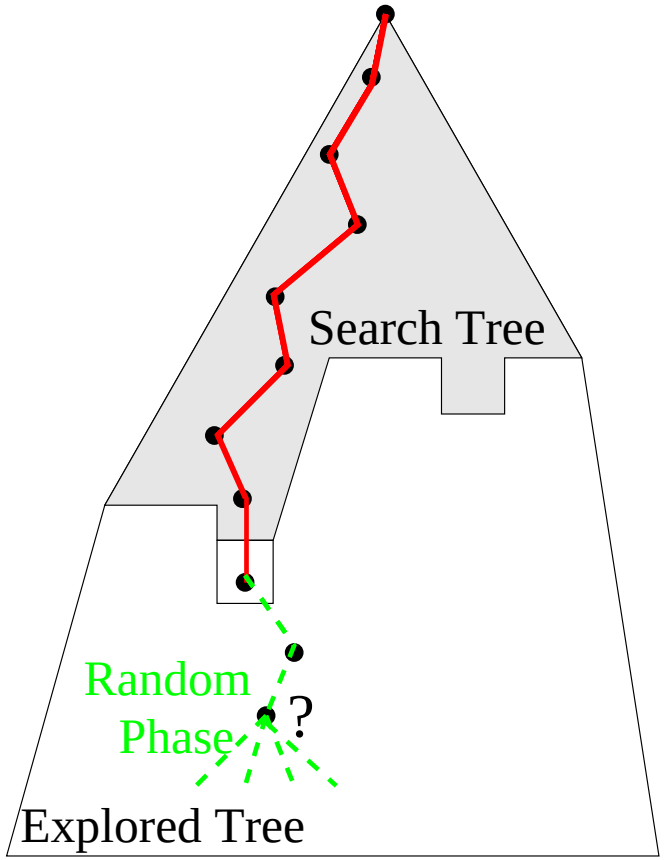


[1] S. Gelly, and D. Silver *Combining online and offline knowledge in UCT*. ICML'07

FUSE: random phase

Dealing with an unknown horizon

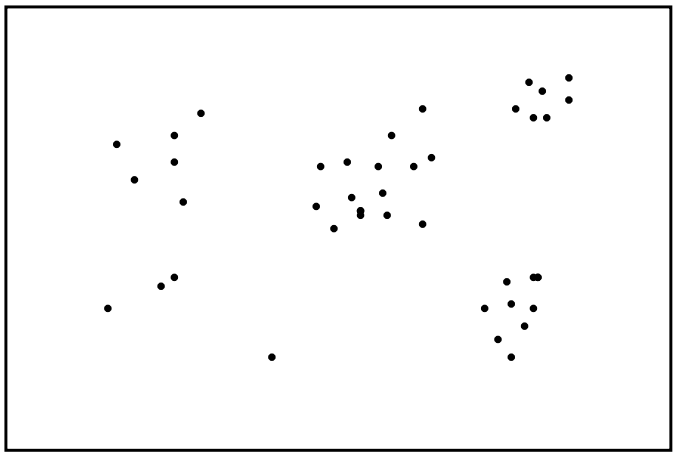
- Unknown best size of the feature subset
- Random phase policy
  - With probability $1 - q^{|F|}$ stop
  - Else • add a uniformly selected feature
    - $|F| = |F| + 1$
  - Iterate



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 ($m \ll n$)

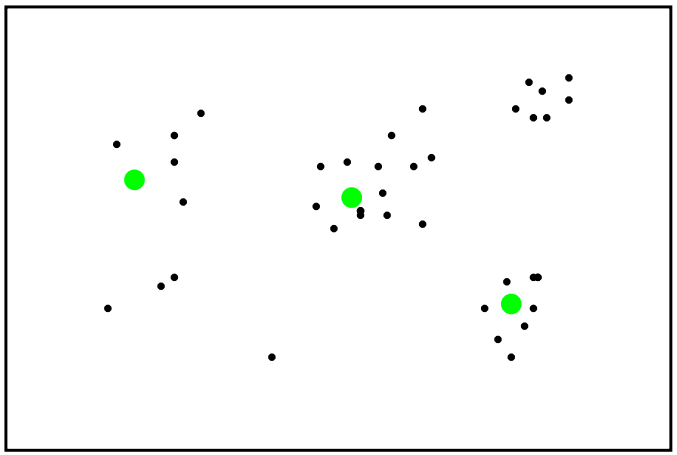


* Mann Whitney Wilcoxon test: $V(F) = \frac{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ \mathcal{N}_{F,k}(x) < \mathcal{N}_{F,k}(x'),\ y < y'\}|}{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ y < y'\}|}$

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 ($m \ll n$)

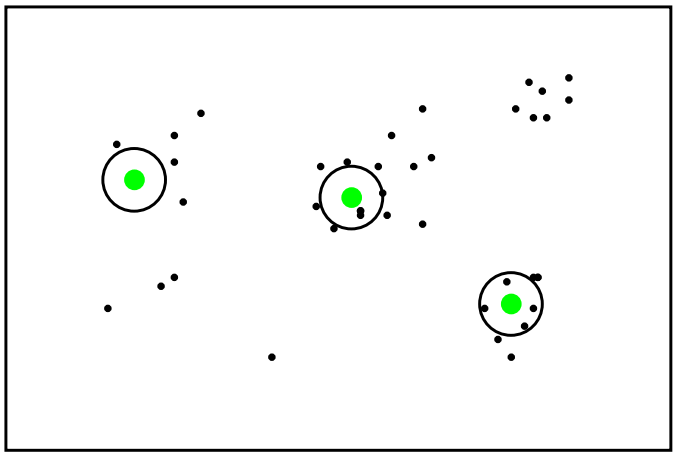


* Mann Whitney Wilcoxon test: $V(F) = \frac{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ \mathcal{N}_{F,k}(x) < \mathcal{N}_{F,k}(x'),\ y < y'\}|}{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ y < y'\}|}$

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 ($m \ll n$)

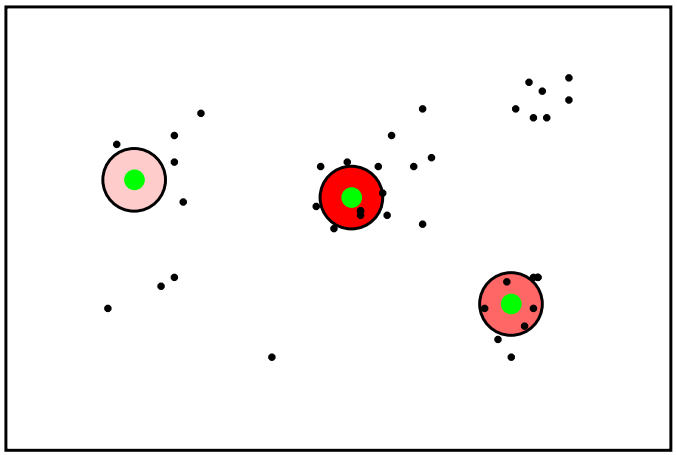


* Mann Whitney Wilcoxon test: $V(F) = \frac{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ \mathcal{N}_{F,k}(x) < \mathcal{N}_{F,k}(x'),\ y < y'\}|}{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ y < y'\}|}$

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 ($m \ll n$)

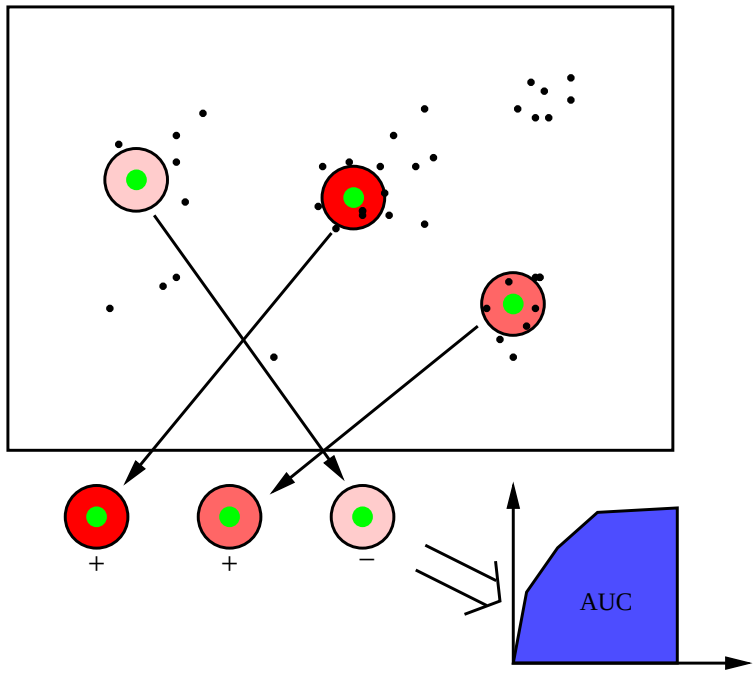


* Mann Whitney Wilcoxon test: $V(F) = \frac{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ \mathcal{N}_{F,k}(x) < \mathcal{N}_{F,k}(x'),\ y < y'\}|}{|\{((x,y),(x',y')) \in \mathcal{V}^2,\ y < y'\}|}$

FUSE: reward(F)

Generalization error estimate

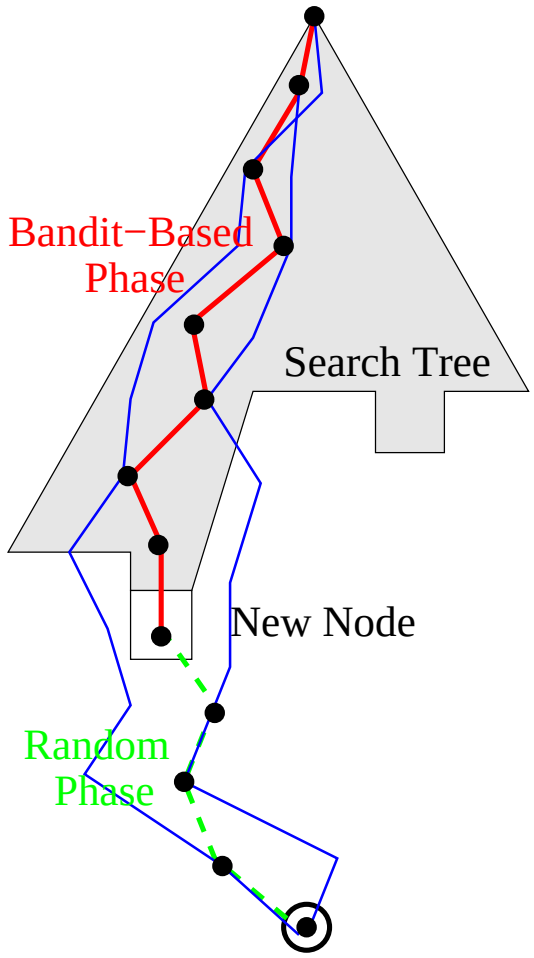
- Requisite
  - fast (to be computed 10^4 times)
  - unbiased
- Proposed reward
  - k-NN like
  - + AUC criterion *
- Complexity: $\tilde{O}(mnd)$
  - d Number of selected features
  - n Size of the training set
  - m Size of sub-sample ($m \ll n$)



* 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
  - $\Rightarrow$ Several paths to the same node
- Update only current path



From UCT to Feature Selection

- The algorithm
  - N iterations: each iteration $\nearrow$ Follow a path; $\searrow$ Evaluate a final node
- $\Rightarrow$ Build

Search tree	$\longleftrightarrow$	RAVE score
$\Downarrow$		$\Downarrow$
FUSE		**FUSE^R**
Wrapper approach		Filter approach
Most visited path		Based on RAVE

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

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"

[1] *Feature Selection Challenge*. NIPS'03

Experimental setting

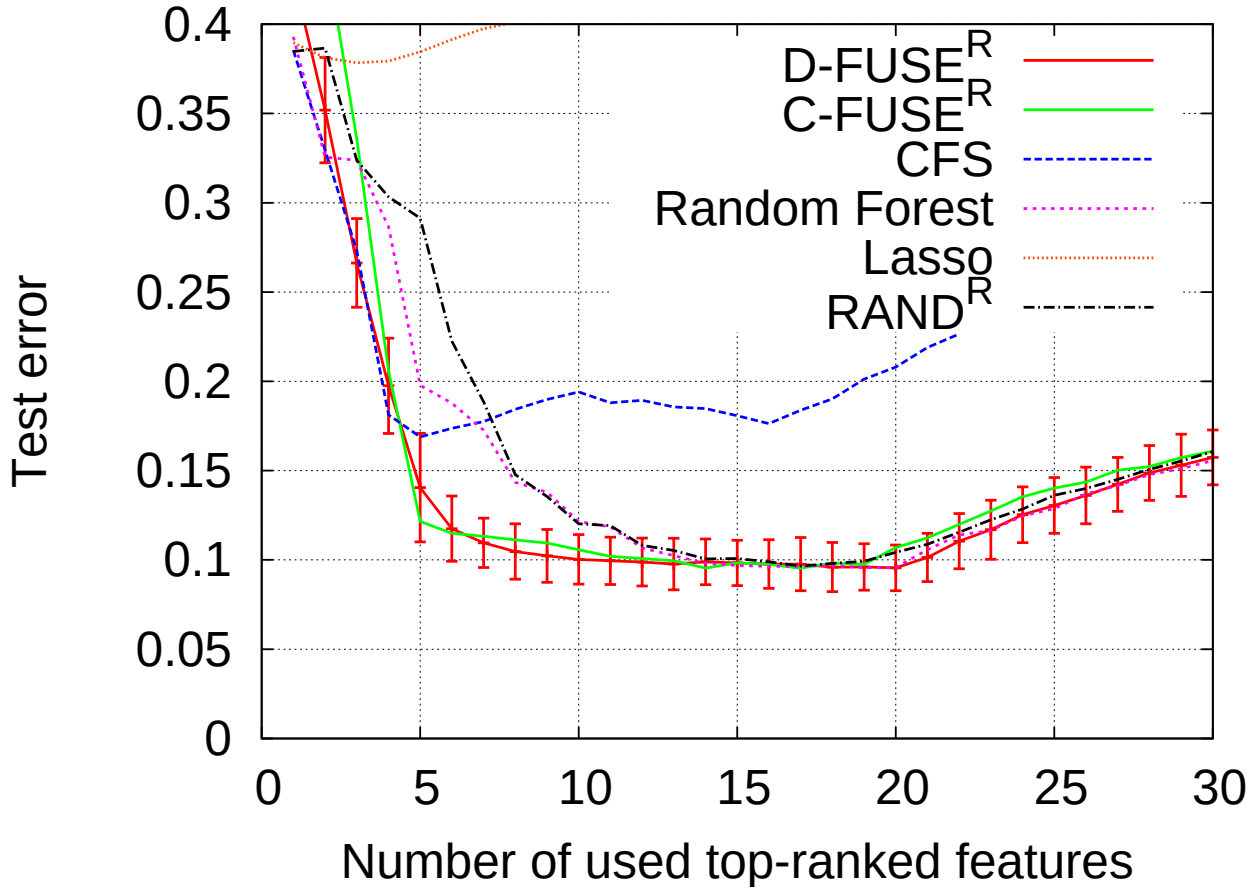
- Baselines
  - CFS (Constraint-based Feature Selection) [1]
  - Random Forest [2]
  - Lasso [3]
  - RAND^R: RAVE obtained by selecting 20 random features at each iteration
- Results averaged on 50 splits (10 $\times$ 5 fold cross-validation)
- End learner
  - Hyper-parameters optimized by 5 fold cross-validation

[1] M. A. Hall *Correlation-based Feature Selection for discrete and numeric class Machine Learning*. ICML'00

[2] J. Rogers, and S. R. Gunn *Identifying feature relevance using a Random Forest*. SLSFS'05

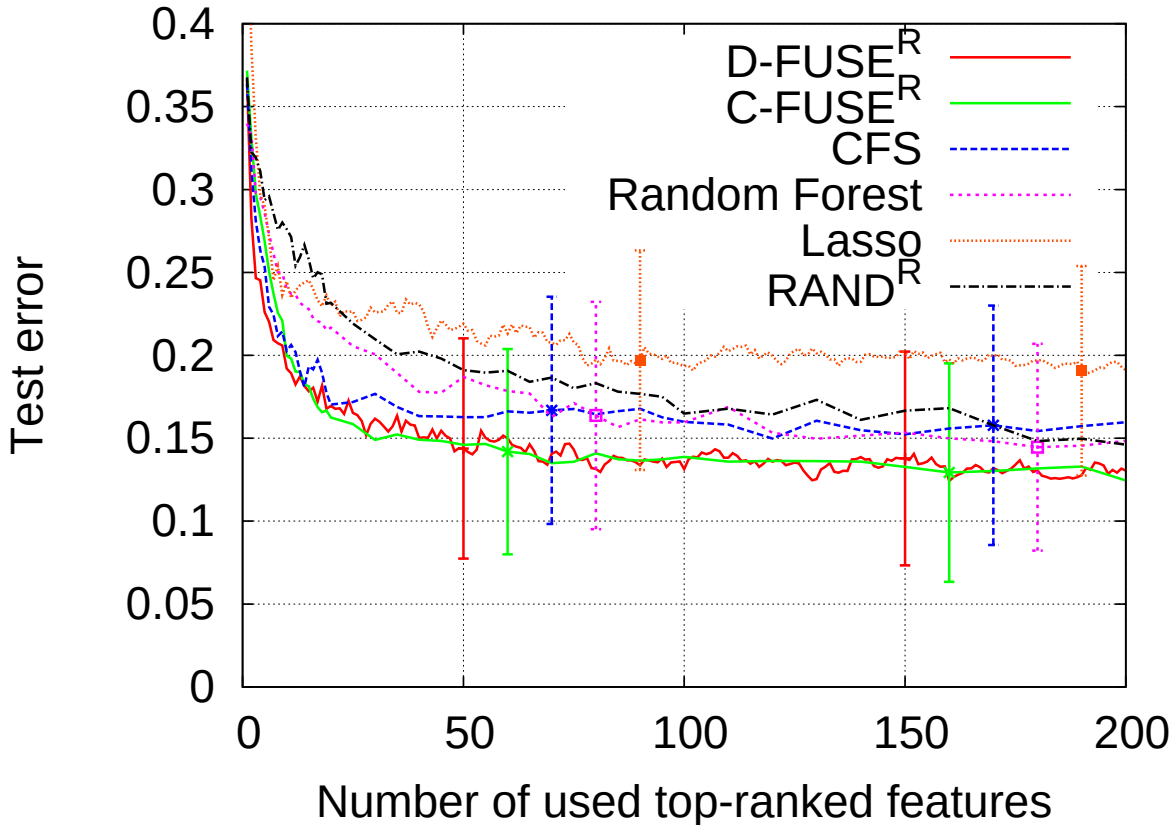
[3] R. Tibshirani *Regression shrinkage and selection via the Lasso*. Journal of the Royal Statistical Society 94

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)

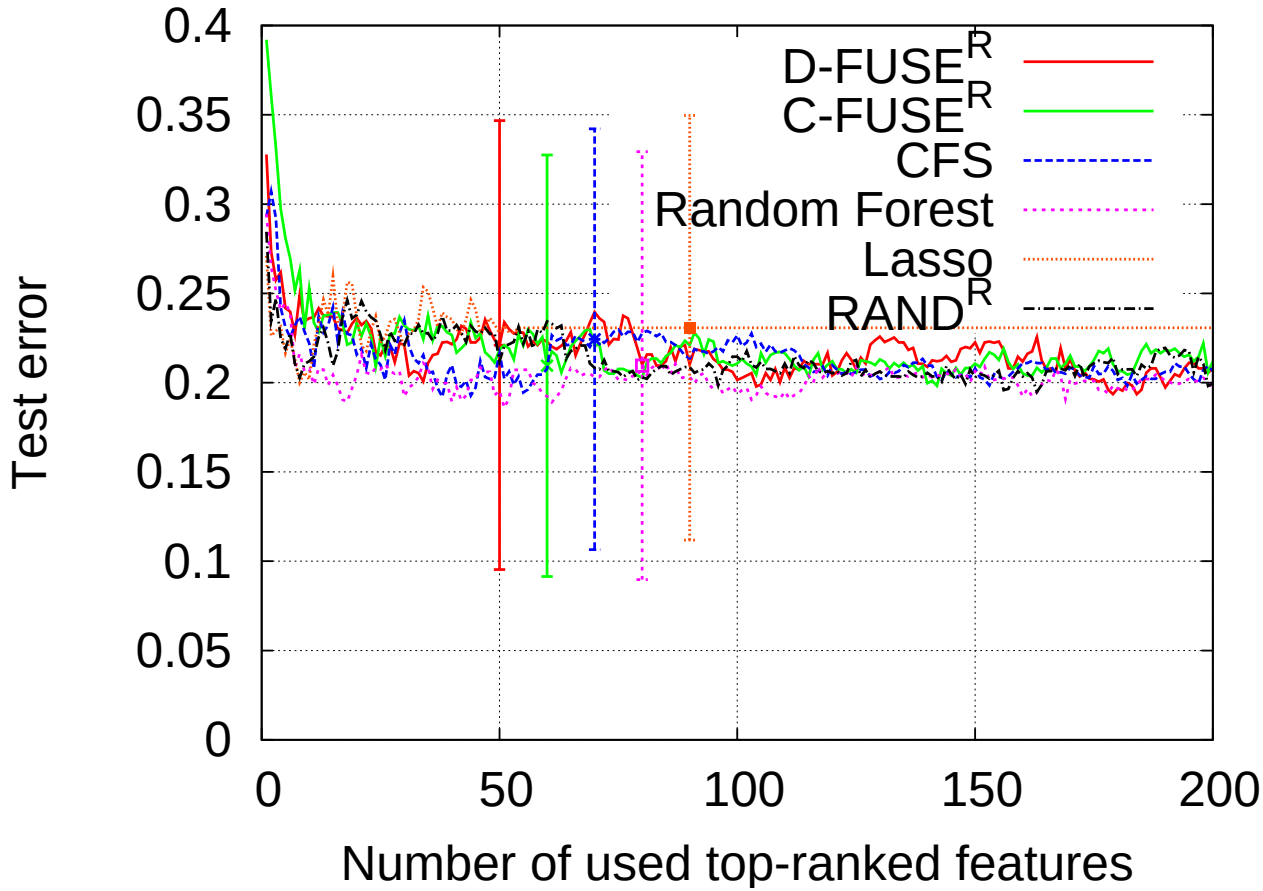
Results on Arcene after 200,000 iterations



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

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

Results on Colon after 200,000 iterations



- Remark
  - 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

- Remarks
  - Selected features: accurate
  - Promising results

---

[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

Conclusion and Perspectives

- 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
  - Other end learners
  - Extend to Feature construction
    - ★ Inspired by [1]

[1] F. de Mesmay, A. Rimmel, Y. Voronenko, and M. Püschel *Bandit-based optimization on graphs with application to library performance tuning*. ICML'09

Feature Selection as a one-player game

Romaric Gaudel[1,2,3] and **Michèle Sebag**[1,2,3]

[1] Univ. Paris-Sud, LRI, UMR8623
[2] CNRS [3] INRIA-Saclay

ICML, June 2010

Filter approaches for Feature Selection

- Score features
- Select the best ones

Pro

- Cheap

Cons

- Cannot tackle all inter-dependencies between features

- Filter approaches
  - ANOVA (Analysis of Variance)
  - RELIEFF [1]

[1] K. Kira, and L. A. Rendell *A practical approach to feature selection*. ML'92

Wrapper approaches for Feature Selection

- Test feature subsets
  - Really tackle the combinatorial problem

Pro

- Look for the best solution

Cons

- Computationally expensive

- Wrapper approaches
  - Look ahead [1]
  - Mix forward/backward search [2]
  - Mix global/local search [3]

---

[1] D. Margaritis *Toward provably correct Feature Selection in arbitrary domains*. NIPS'09

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

[3] M. Boullé *Compression-based averaging of selective Naive Bayes classifiers*. J. Mach. Learn. Res. 07

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 [1] or a linear combination of kernels [2]
- Possibly misled by feature interdependencies

- Embedded approaches
  - Lasso [1]
  - Multiple Kernel Learning [2]
  - Gini score on Random Forest [3]

---

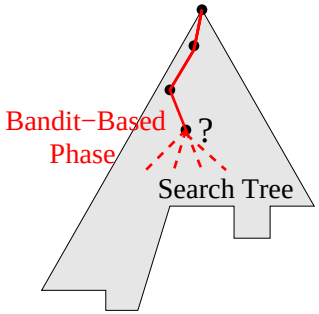
[1] R. Tibshirani *Regression shrinkage and selection via the Lasso*. Journal of the Royal Statistical Society 94

[2] F. Bach *Exploring large feature spaces with hierarchical Multiple Kernel Learning*. NIPS'08

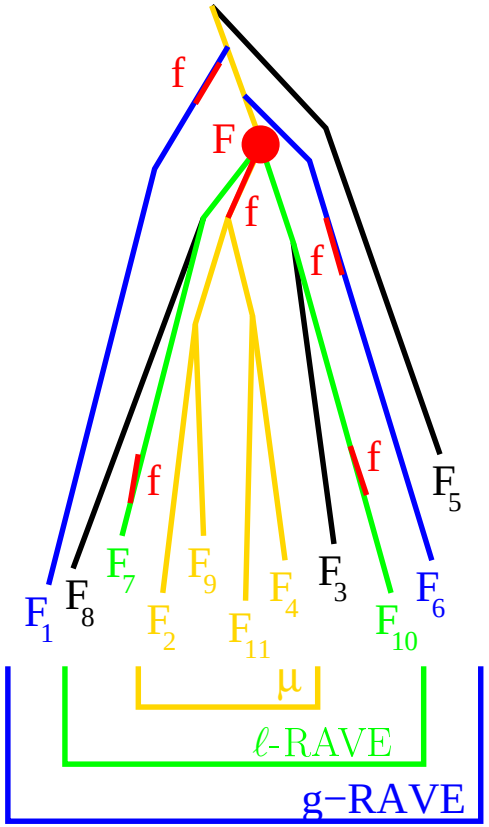
[3] J. Rogers, and S. R. Gunn *Identifying feature relevance using a Random Forest*. SLSFS'05

FUSE: bandit-based phase

Dealing with many arms



- Bottleneck
  - A many-armed problem (hundreds of features)
    - $\Rightarrow$ need to guide UCT
- Ingredient 1: controlling the number of arms
  - Discrete heuristics [1,2]: Progressive Widening
    - ★ Consider only $\lfloor T^b \rfloor$ actions
  - Continuous heuristics [3]
    - ★ Use a small exploration constant c_e
- Ingredient 2: sharing information among nodes
  - Rapid Action Value Estimation (RAVE) [3]



[1] P. Rolet, M. Sebag, and O. Teytaud *Boosting Active Learning to optimality: a tractable Monte-Carlo, Billiard-based algorithm*. ECML'09

[2] R. Coulom *Efficient selectivity and backup operators in Monte-Carlo tree search*. Computer and Games 2006

[3] S. Gelly, and D. Silver *Combining online and offline knowledge in UCT*. ICML'07

FUSE: bandit-based phase

Sharing information among nodes

- Use of RAVE
  - Discrete heuristics [1]
    - ★ When a new action allowed, add $\operatorname{argmax}_{f \in \mathcal{F}} \text{RAVE}(f)$
  - Continuous heuristics [2]
    - ★ Tradeoff UCB-RAVE

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

- ★ $\alpha \searrow$ when $t_{F,f} \nearrow$
- ★ $\beta \searrow$ when $\#\{f \in F_t, F \rightsquigarrow F_t\} \nearrow$

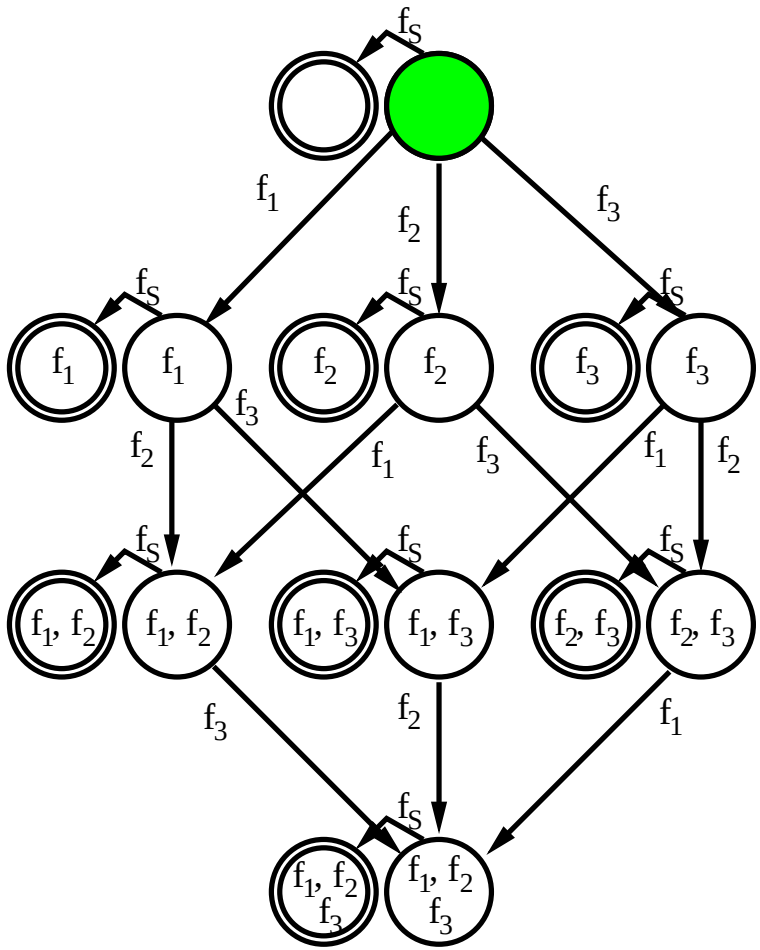
[1] P. Rolet, M. Sebag, and O. Teytaud *Boosting Active Learning to optimality: a tractable Monte-Carlo, Billiard-based algorithm*. ECML'09

[2] S. Gelly, and D. Silver *Combining online and offline knowledge in UCT*. ICML'07

Feature stop

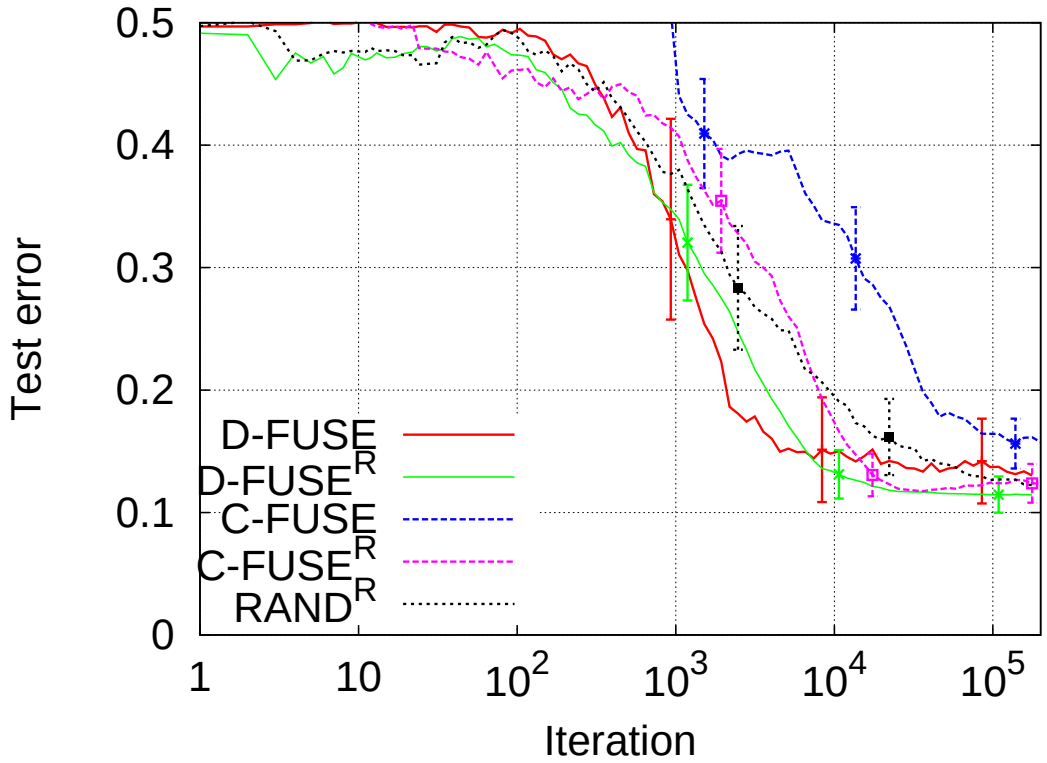
Dealing with an unknown horizon

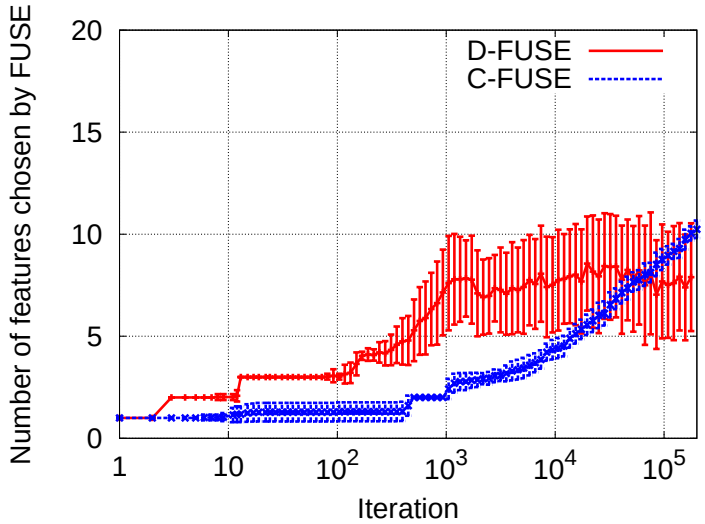
- Any state can be final or not
  - Final(F) = "$f_s \in F$"
  - f_s: A virtual stopping feature
- RAVE(f_s)
  - g-RAVE($f_s^{(d)}$) $= average\,\{V(F_t), |F_t| = d+1\}$
    - ★ $V(F_t)$: Reward of Feature Subset F_t selected at iteration t
    - ★ d: When RAVE(f_s) is used, d is set to the number of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



Sensitivity of FUSE to the Computational Effort

Madelon





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

Experimental setting

- FUSE Hyperparameters

			How to restrict exploration		
			Discrete heuristics	Continuous heuristics	
Parameter	k-NN	q	b	c_e	c, c_l
Value i	5-NN	$1 - 10^i$ $\{-1, -3, -5\}$	$1/2$	10^i $\{-4, -2, 0, 2\}$	10^i $\{-\infty, 2, 4\}$

- Plot best results for FUSE^R
- Preliminary results
  - FUSE is limited to deal with deep search tree
  - FUSE coincides with the beginning of the FUSE^R curve

Best hyperparameters

HEURISTICS	ANY	DISC.	CONTINUOUS	
PARAMETER	q	b	c_e	c, c_l
TESTED VALUE i	$1-10^i$ $\{-1,-3,-5\}$	$1/2$	10^i $\{-4,-2,0,2\}$	10^i $\{-\infty,2,4\}$
ARCENE	$1-10^{-1}$	$1/2$		
	$1-10^{-1}$		10^{-2}	ANY
	$1-10^{-3}$		10^{-4}	ALMOST ANY
MADELON	$1-10^{-3}$	$1/2$		
	$1-10^{-1}$		10^{-2}	$\{(10^2,0),(10^4,0)\}$
COLON	$1-10^{-5}$	$1/2$		
	$1-10^{-5}$		ANY	ALMOST ANY