Sequences classification by least general generalisations

Fabien TORRE
joint work with F. TANTINI and A. TERLUTTE

INRIA LNE and CNRS LIFL (Mostrare) – LORIA (Parole)

ICGI 2010, Valencia

Outline of the talk

Supervised classification with VOLATA

A supervised classification problem

needs to define:

1. an input space: $\mathcal{X}$;
2. a finite set of discrete classes: $\mathcal{Y}$;
3. an hypothesis language: $\mathcal{H} \supseteq \mathcal{X}$ and $h \in \mathcal{H} : \mathcal{X} \rightarrow \mathcal{Y}$;
4. a subsumption relation $\succeq$ between hypotheses.

Supervised classification with VOLATA

A supervised classification problem

needs to define:

1. an input space: $\mathcal{X}$;
2. a finite set of discrete classes: $\mathcal{Y}$;
3. an hypothesis language: $\mathcal{H} \supseteq \mathcal{X}$ and $h \in \mathcal{H} : \mathcal{X} \to \mathcal{Y}$;
4. a subsumption relation $\succeq$ between hypotheses.

VOLATA

- requires a generalisation operator called *least general generalisation* (LGG);
- then provides several learning methods, especially ensemble methods.

Least General Generalisations

Definition: *lgg*

Given a set of examples $E \subseteq \mathcal{X}$, an hypothesis $h \in \mathcal{H}$ is a *least general generalisation* of E iff:

- $\forall e \in E : h \succeq e$;
- there exists no h' such that $\forall e \in E : h' \succeq e$ and $h \succ h'$.

Least General Generalisations

Definition: *lgg*

Given a set of examples $E \subseteq \mathcal{X}$, an hypothesis $h \in \mathcal{H}$ is a *least general generalisation* of E iff:

- $\forall e \in E : h \succeq e$;
- there exists no h' such that $\forall e \in E : h' \succeq e$ and $h \succ h'$.

... and if the lgg is unique

Two possible definitions of the LGG operator:

- $\mathrm{LGG}(e_1, e_2, \ldots, e_n \in \mathcal{X})$ returns $h \in \mathcal{H}$;
- $\mathrm{LGG}(h_{n-1}, e_n)$ returns $h \in \mathcal{H}$.

We prefer the second one, more efficient for learning.

VOLATA: a three-levels architecture

- (level 1) LGG operator computes the least general generalisation of a set of examples. Follows from $(\mathcal{H}, \succeq)$.

VOLATA: a three-levels architecture

- (level 1) LGG operator computes the least general generalisation of a set of examples. Follows from $(\mathcal{H}, \succeq)$.
- (level 2) examples generalisation using LGG and classes. For a given class, CG (*correct generalisation*) generalises examples one by one and checks correction of each generalisation wrt other classes. Depends on the presentation order of examples.

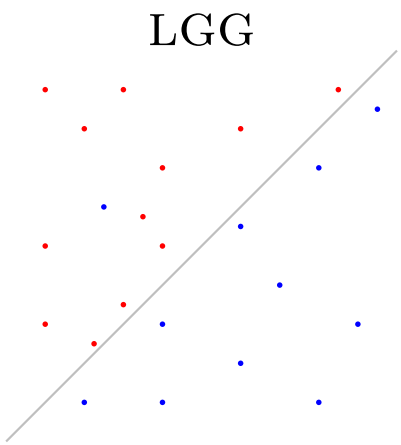
VOLATA: a three-levels architecture

- (level 1) LGG operator computes the least general generalisation of a set of examples. Follows from $(\mathcal{H}, \succeq)$.
- (level 2) examples generalisation using LGG and classes. For a given class, CG (*correct generalisation*) generalises examples one by one and checks correction of each generalisation wrt other classes. Depends on the presentation order of examples.
- (level 3) full classifiers learning. GLOBOOST is an ensemble method that uses the order dependency of CG to obtain random correct hypotheses and combine them with a *one hypothesis, one vote* principle. Importance of diversity.

Only the first level depends on hypothesis language $\mathcal{H}$, the two others are generics.

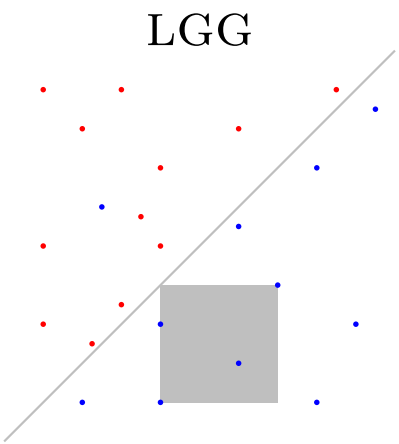
LGG, CG and GLOBOOST in action

In the plane with examples/points and hypotheses/rectangles:



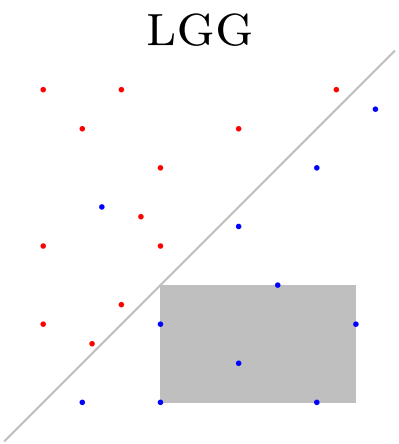
LGG, CG and GLOBOOST in action

In the plane with examples/points and hypotheses/rectangles:



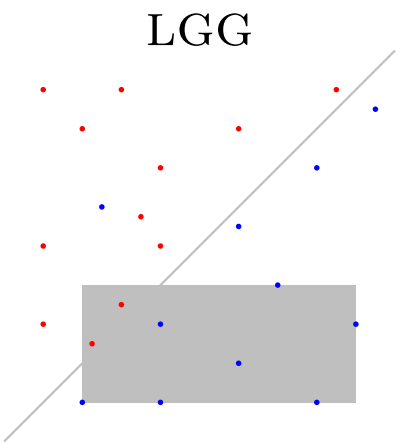
LGG, CG and GLOBOOST in action

In the plane with examples/points and hypotheses/rectangles:



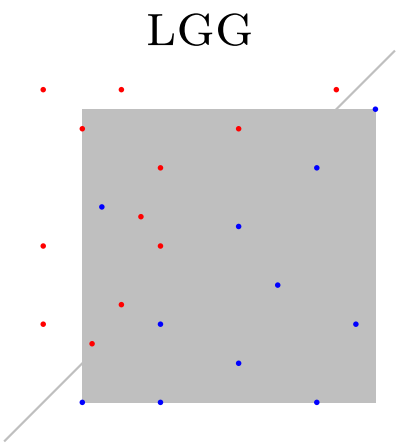
LGG, CG and GLOBOOST in action

In the plane with examples/points and hypotheses/rectangles:



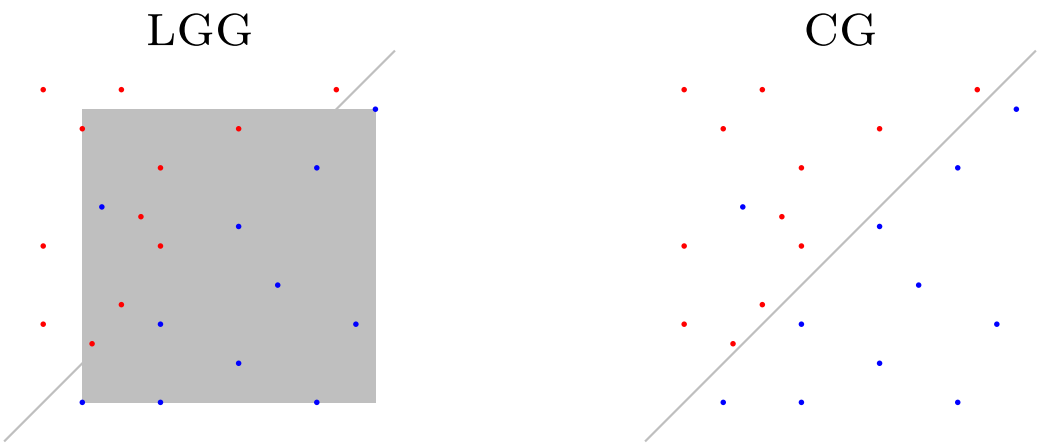
LGG, CG and GLOBOOST in action

In the plane with examples/points and hypotheses/rectangles:



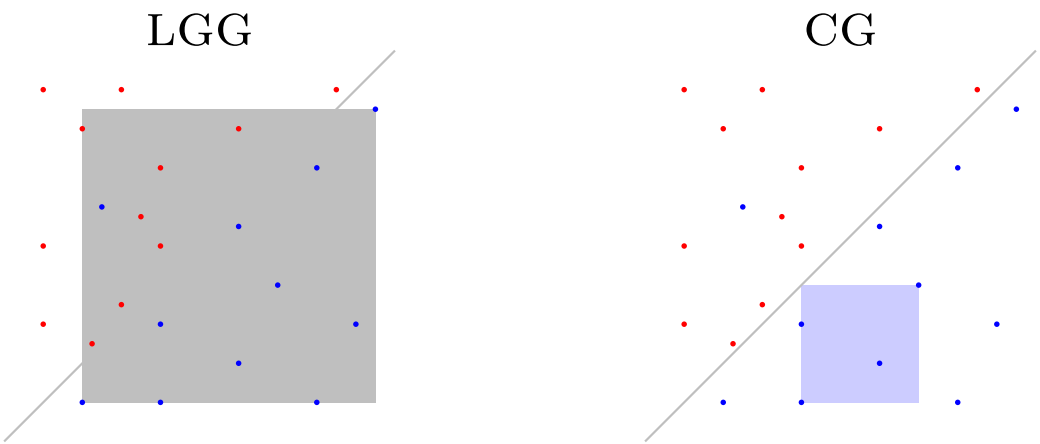
LGG, CG and GloBoost in action

In the plane with examples/points and hypotheses/rectangles:



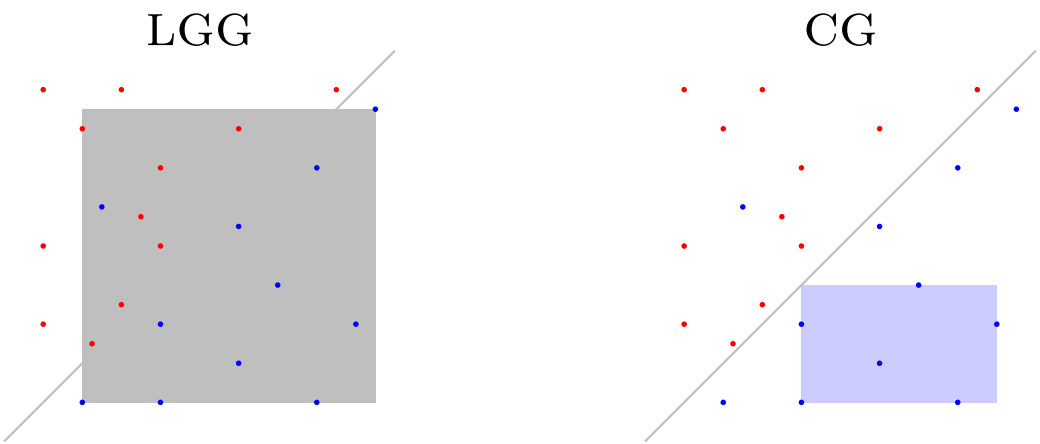
LGG, CG and GloBoost in action

In the plane with examples/points and hypotheses/rectangles:



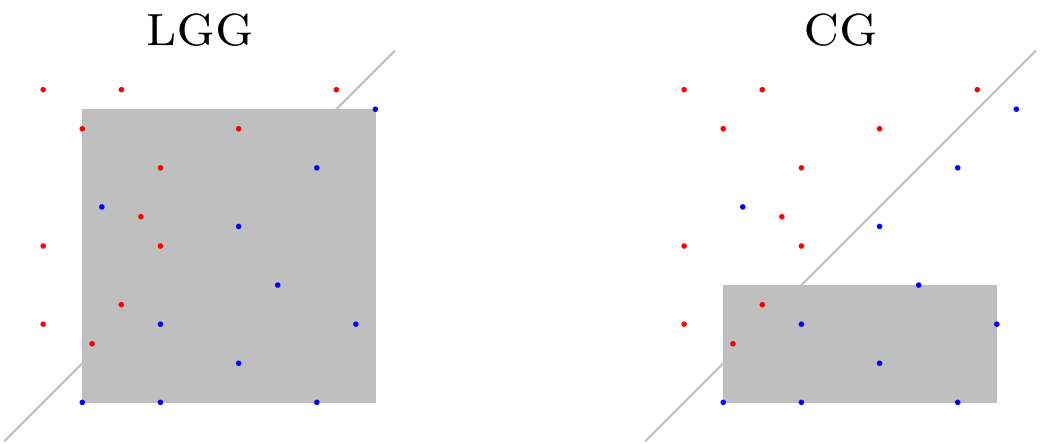
LGG, CG and GloBoost in action

In the plane with examples/points and hypotheses/rectangles:



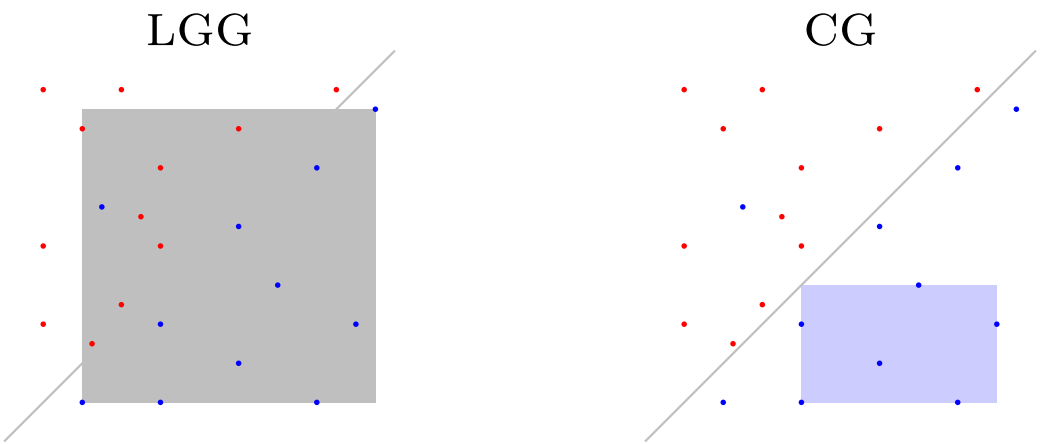
LGG, CG and GloBoost in action

In the plane with examples/points and hypotheses/rectangles:



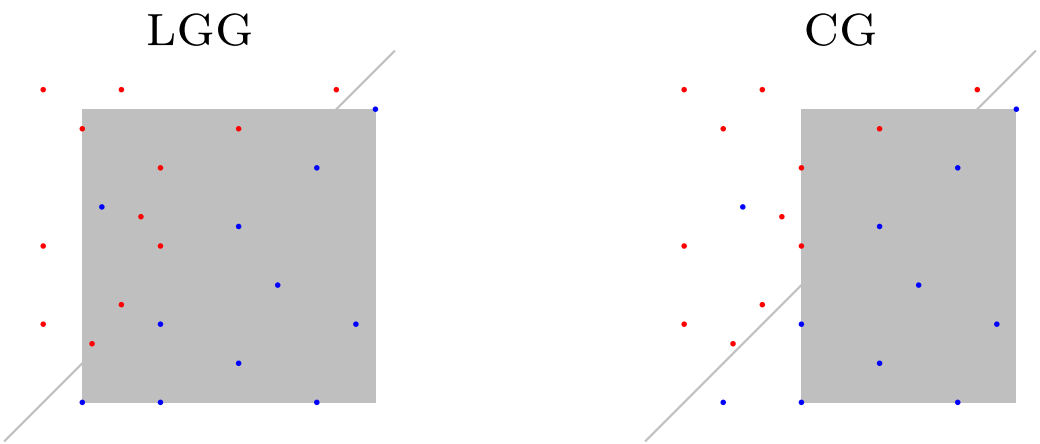
LGG, CG and GloBoost in action

In the plane with examples/points and hypotheses/rectangles:



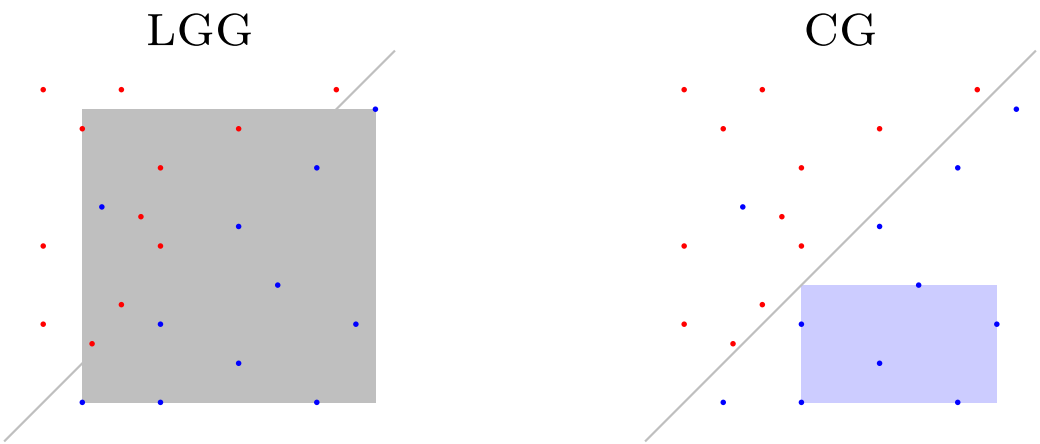
LGG, CG and GloBoost in action

In the plane with examples/points and hypotheses/rectangles:



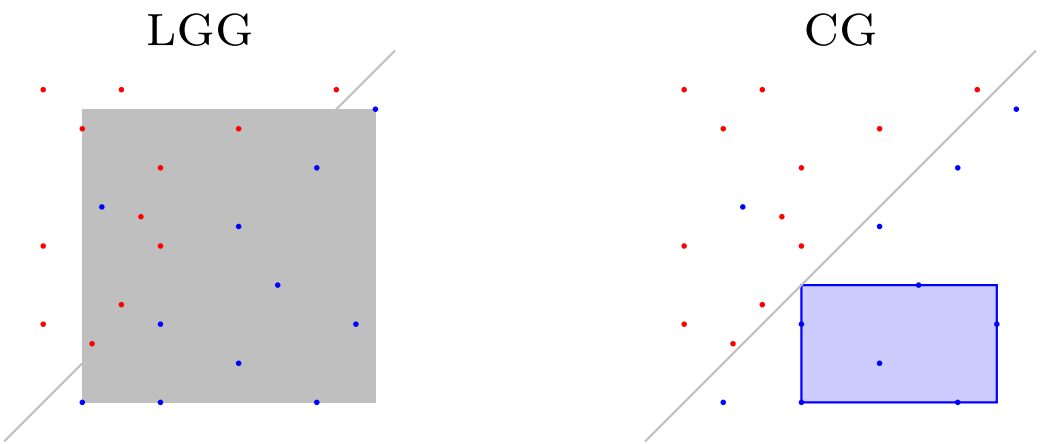
LGG, CG and GloBoost in action

In the plane with examples/points and hypotheses/rectangles:



LGG, CG and GloBoost in action

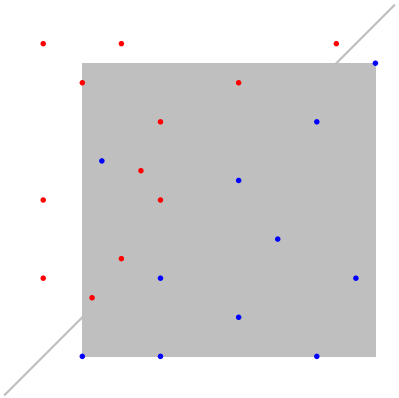
In the plane with examples/points and hypotheses/rectangles:



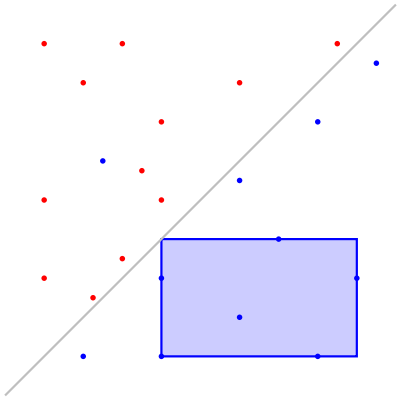
LGG, CG and GLOBOOST in action

In the plane with examples/points and hypotheses/rectangles:

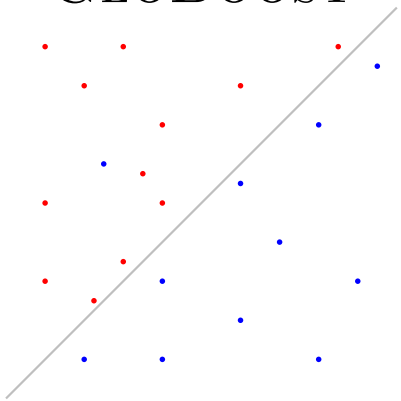
LGG



CG

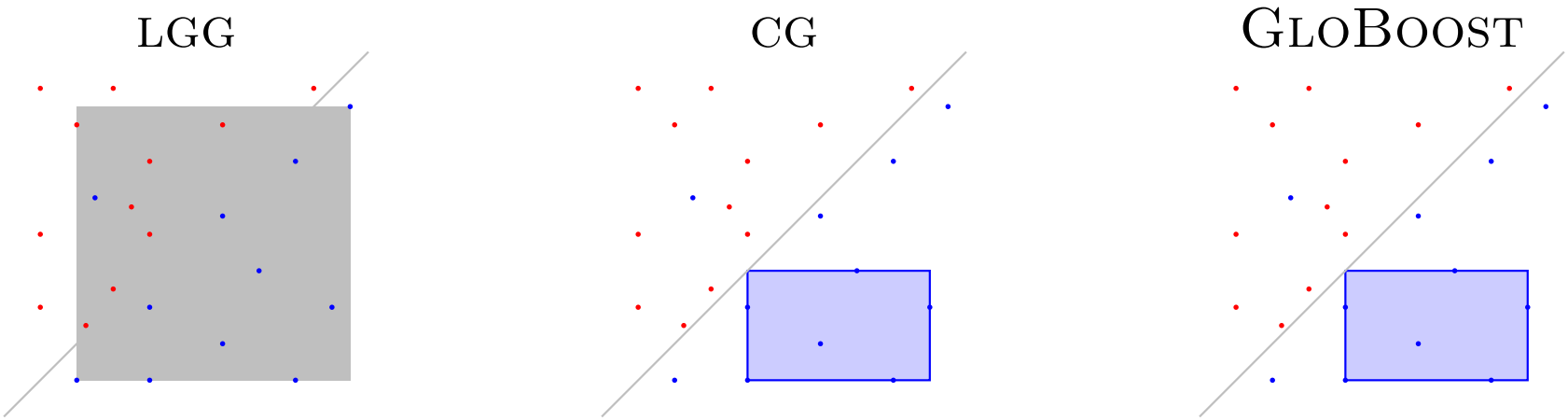


GLOBOOST



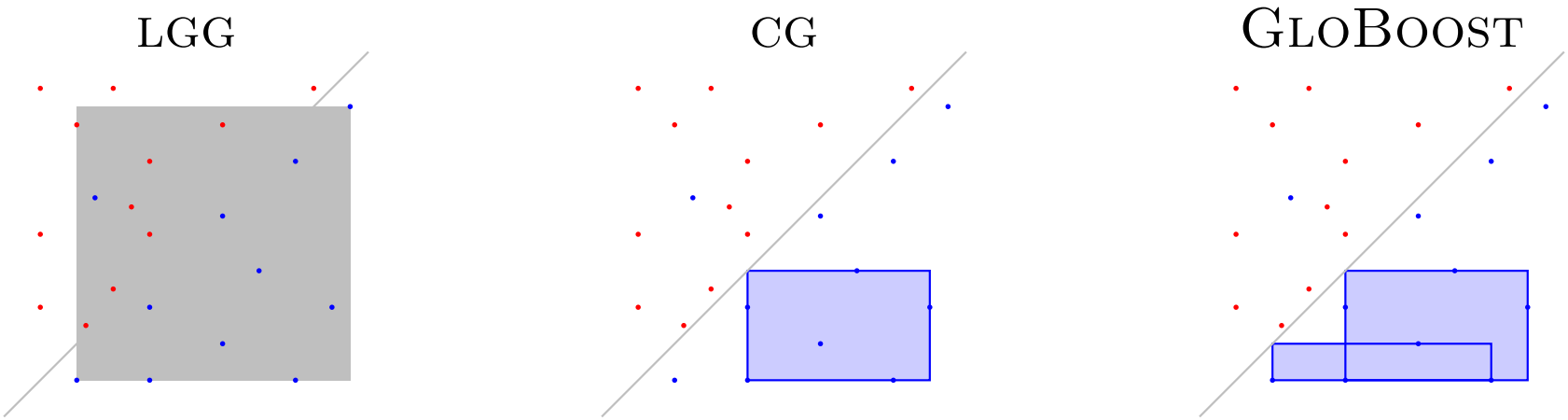
LGG, CG and GloBoost in action

In the plane with examples/points and hypotheses/rectangles:



LGG, CG and GloBoost in action

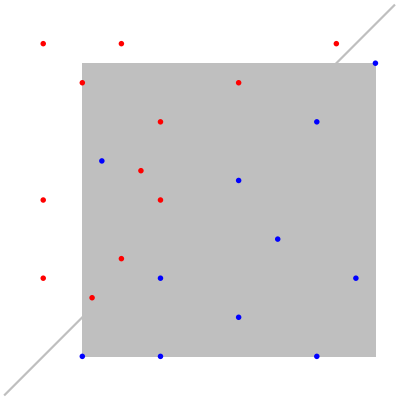
In the plane with examples/points and hypotheses/rectangles:



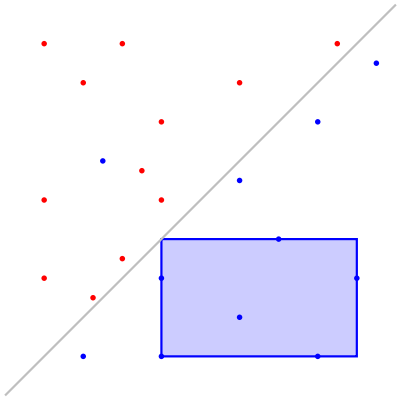
LGG, CG and GLOBOOST in action

In the plane with examples/points and hypotheses/rectangles:

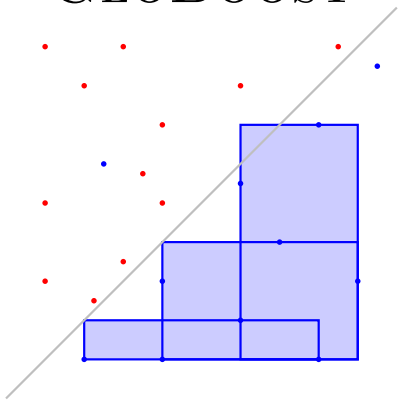
LGG



CG



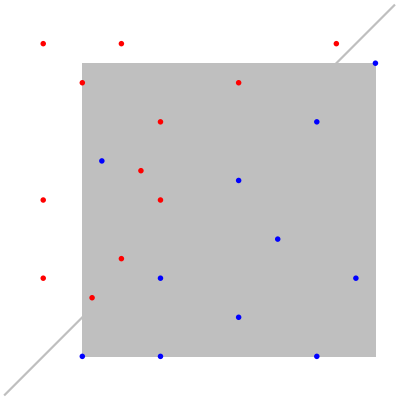
GLOBOOST



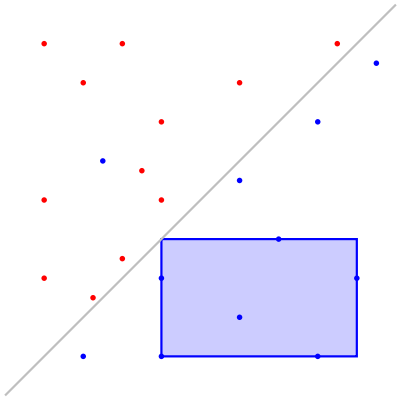
LGG, CG and GLOBOOST in action

In the plane with examples/points and hypotheses/rectangles:

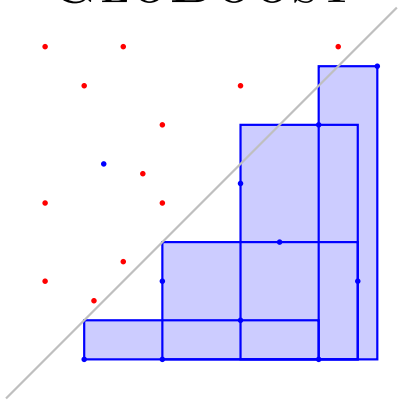
LGG



CG



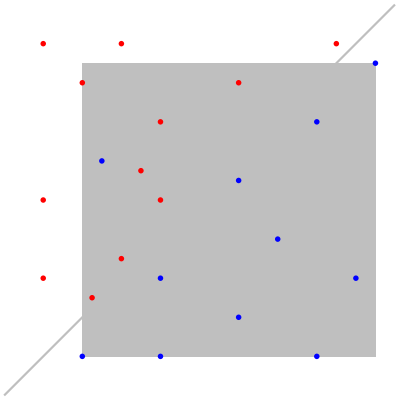
GLOBOOST



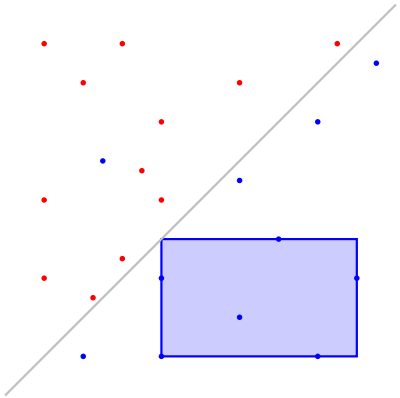
LGG, CG and GLOBOOST in action

In the plane with examples/points and hypotheses/rectangles:

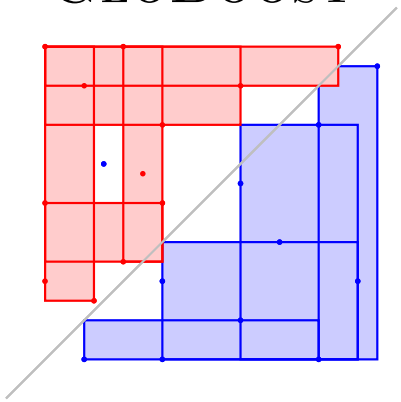
LGG



CG

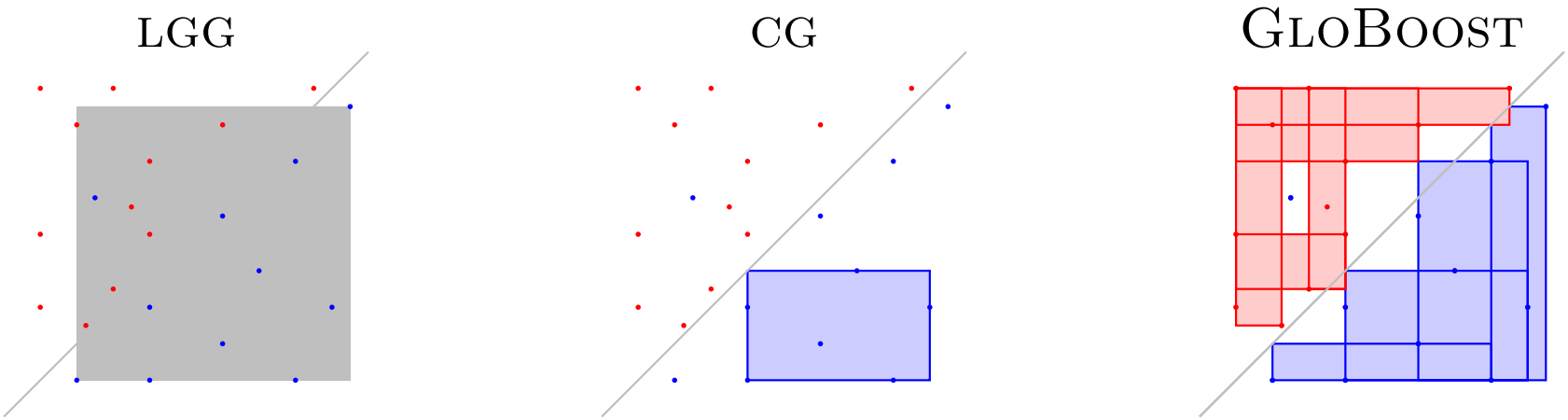


GLOBOOST



LGG, CG and GloBoost in action

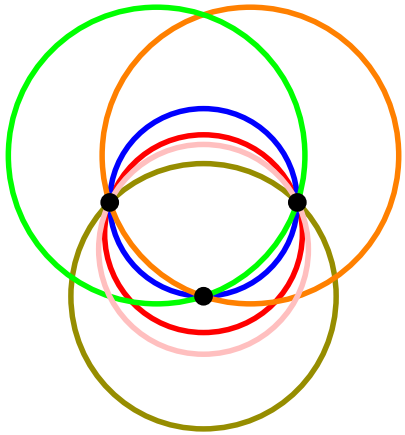
In the plane with examples/points and hypotheses/rectangles:



works only because *least general rectangles* are unique...

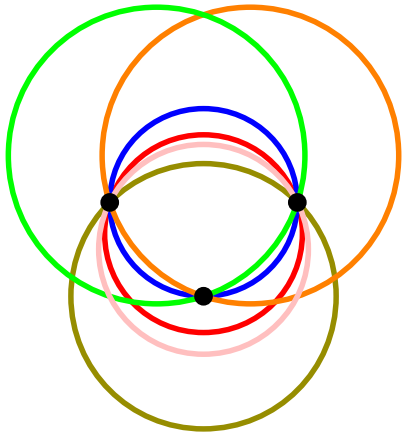
The case of disks in the plane

In the plane,
with examples/points and hypotheses/disks,
there is an infinity of least general disks
that include three points.



The case of disks in the plane

In the plane,
with examples/points and hypotheses/disks,
there is an infinity of least general disks
that include three points.



Requirements to classify with VOLATA

1. choose appropriate $(\mathcal{H}, \succeq)$;
2. guarantee that least general hypotheses are unique;
3. define the corresponding LGG operator.

Application to grammatical inference and sequence classification?

Outline of the talk

Least general generalisations and grammatical inference

Comparison

Learning in the limit. When a positive example arrives, the learner must propose a language that contains seen examples, and finally the target language.

LGG *operator in GI context.* Given a language L and a word w, it provides the smallest language that contains L and w.

LGG operators learn in the limit!

Least general generalisations and grammatical inference

Comparison

Learning in the limit. When a positive example arrives, the learner must propose a language that contains seen examples, and finally the target language.

LGG *operator in GI context.* Given a language L and a word w, it provides the smallest language that contains L and w.

LGG operators learn in the limit!

LGG operators in learnability proofs

Available for:

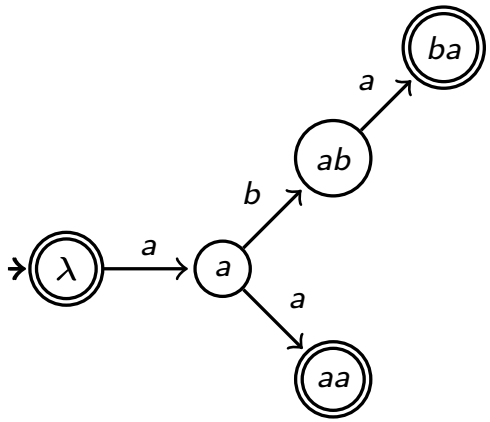
- k-TSS automata [García and Vidal, 1990]
- 0-reversible automata [Angluin, 1982]

but not for balls of words [de la Higuera et al., 2008].

LGG operators for k-TSS and 0-reversible automata

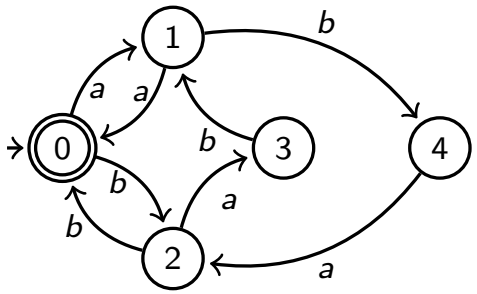
LGG-TSSI:

1. $S = \{\lambda, aa, aba\}$ and $k = 3$;
2. learned automaton:



LGG-ZR:

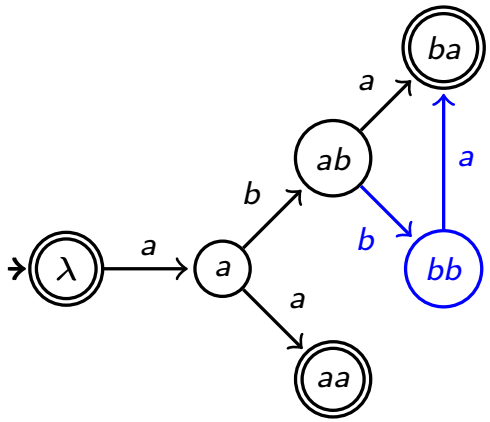
1. $S = \{\lambda, aa, bb, abab, baba\}$
2. learned automaton:



LGG operators for k-TSS and 0-reversible automata

LGG-TSSI:

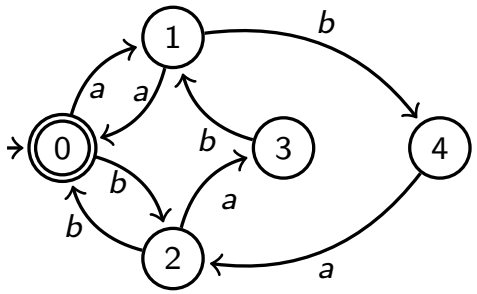
1. $S = \{\lambda, aa, aba\}$ and $k = 3$;
2. learned automaton:



3. adding *abba*

LGG-ZR:

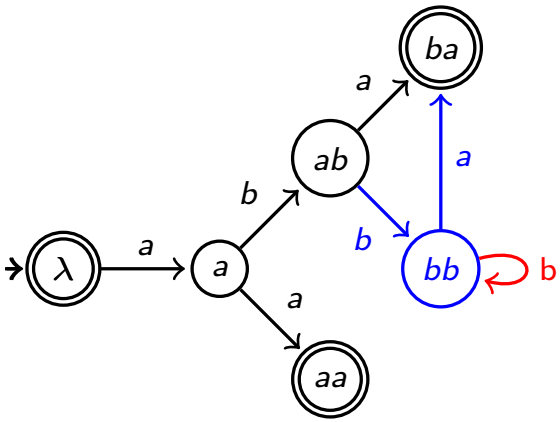
1. $S = \{\lambda, aa, bb, abab, baba\}$
2. learned automaton:



LGG operators for k-TSS and 0-reversible automata

LGG-TSSI:

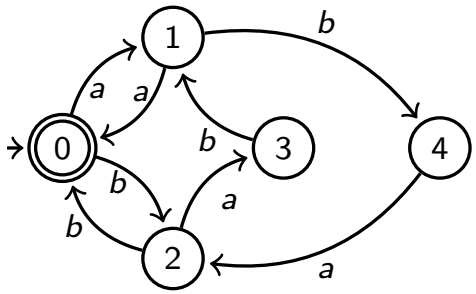
1. $S = \{\lambda, aa, aba\}$ and $k = 3$;
2. learned automaton:



3. adding *abba* and *abbba*.

LGG-ZR:

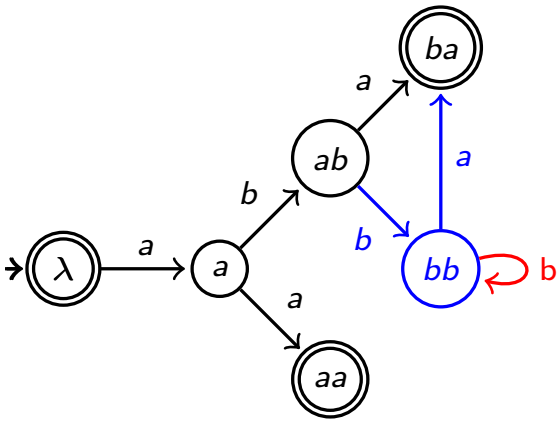
1. $S = \{\lambda, aa, bb, abab, baba\}$
2. learned automaton:



LGG operators for k-TSS and 0-reversible automata

LGG-TSSI:

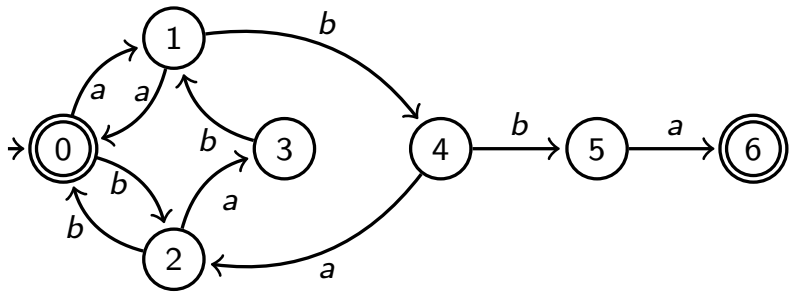
1. $S = \{\lambda, aa, aba\}$ and $k = 3$;
2. learned automaton:



3. adding *abba* and *abbba*.

LGG-ZR:

1. $S = \{\lambda, aa, bb, abab, baba\}$
2. learned automaton:

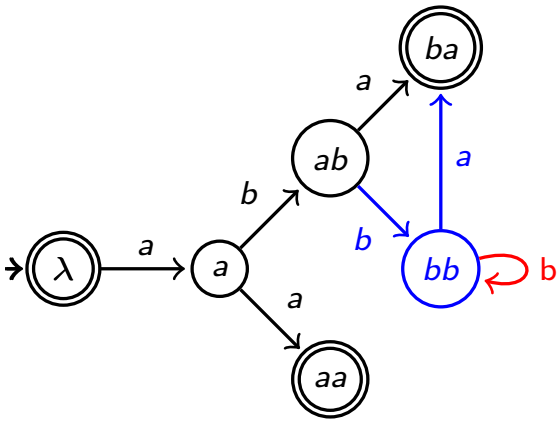


3. adding *abba*.

LGG operators for k-TSS and 0-reversible automata

LGG-TSSI:

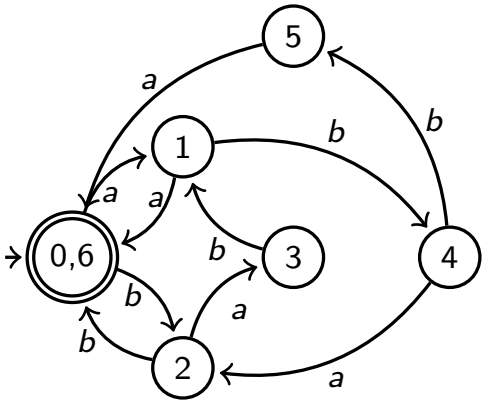
1. $S = \{\lambda, aa, aba\}$ and $k = 3$;
2. learned automaton:



3. adding *abba* and *abbba*.

LGG-ZR:

1. $S = \{\lambda, aa, bb, abab, baba\}$
2. learned automaton:

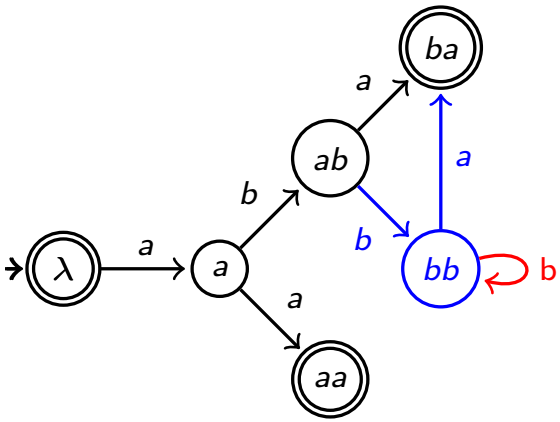


3. adding *abba*.

LGG operators for k-TSS and 0-reversible automata

LGG-TSSI:

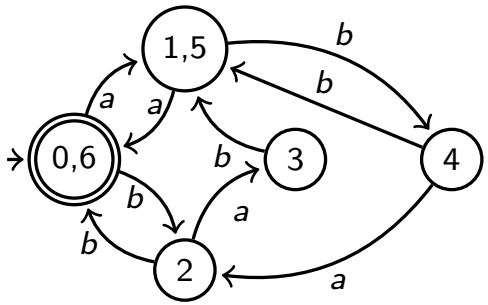
1. $S = \{\lambda, aa, aba\}$ and $k = 3$;
2. learned automaton:



3. adding *abba* and *abbba*.

LGG-ZR:

1. $S = \{\lambda, aa, bb, abab, baba\}$
2. learned automaton:

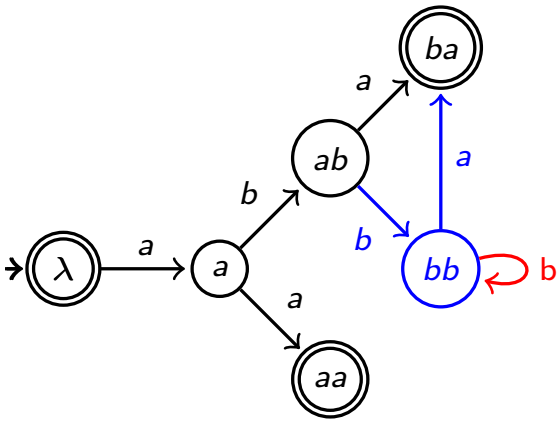


3. adding *abba*.

LGG operators for k-TSS and 0-reversible automata

LGG-TSSI:

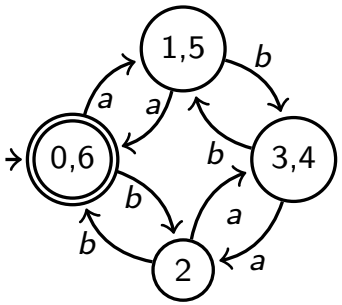
1. $S = \{\lambda, aa, aba\}$ and $k = 3$;
2. learned automaton:



3. adding *abba* and *abbba*.

LGG-ZR:

1. $S = \{\lambda, aa, bb, abab, baba\}$
2. learned automaton:

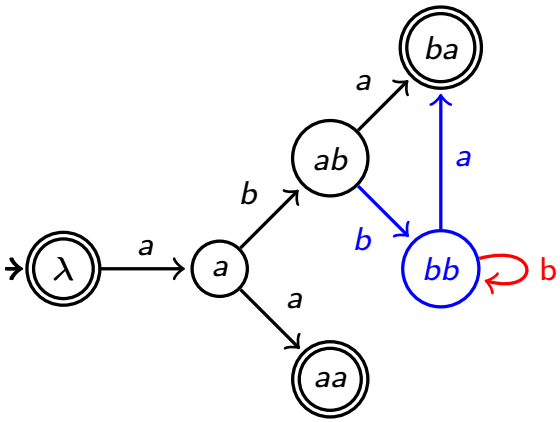


3. adding *abba*.

LGG operators for k-TSS and 0-reversible automata

LGG-TSSI:

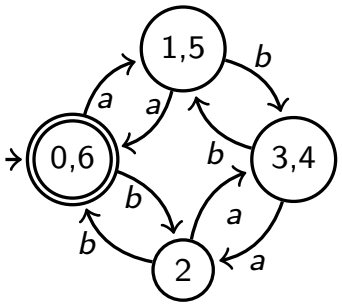
1. $S = \{\lambda, aa, aba\}$ and $k = 3$;
2. learned automaton:



3. adding *abba* and *abbba*.

LGG-ZR:

1. $S = \{\lambda, aa, bb, abab, baba\}$
2. learned automaton:



3. adding *abba*.

Now able to combine such automata and predict class sequences!

Balls of words: another representation for languages

Balls of words [de la Higuera et al., 2008]

- three unit cost edit operations:
  - insertion : $aab \rightarrow aa\underline{b}b$
  - deletion : $a\underline{a}b \rightarrow ab$
  - substitution: $a\underline{a}b \rightarrow a\underline{b}b$
- edit distance: $d(w_1, w_2)$ is the minimum number of operations needed to transform w_1 into w_2;
- a langage is defined by a centre (a word) and a radius: $e \in B_2(bab)$ (or $B_2(bab) \succeq e$) iff $d(bab, e) \leq 2$;
- a ball of words represents a finite language;
- learnability results, by positive examples, with noisy data.

Balls of words: another representation for languages

Balls of words [de la Higuera et al., 2008]

- three unit cost edit operations:
  - insertion : $aab \rightarrow aa\underline{b}b$
  - deletion : $a\underline{a}b \rightarrow ab$
  - substitution: $a\underline{a}b \rightarrow a\underline{b}b$
- edit distance: $d(w_1, w_2)$ is the minimum number of operations needed to transform w_1 into w_2;
- a langage is defined by a centre (a word) and a radius: $e \in B_2(bab)$ (or $B_2(bab) \succeq e$) iff $d(bab, e) \leq 2$;
- a ball of words represents a finite language;
- learnability results, by positive examples, with noisy data.

LGG operator for balls of words ?

Multiple least general generalisations for balls of words

A sample with multiple *lgg*

- Let $E = [a, b, ab]$;
- $h = B_1(a)$ subsumes all examples in E;
- $h' = B_1(b)$ subsumes all examples in E;
- h contains aa, h' contains bb: h and h' are not comparable;
- both h and h' are *lgg* of E because all balls included in both h and h' do not subsume E (like $B_1(\lambda)$ that does not subsume $ab \in E$).

The *unique lgg assumption* is not true for balls of words.
We have to define a new generalisation operator for balls!

An algorithm for generalising balls of words

Algorithm GBALL

Require: $e \in \mathcal{X}$ an example, $h = B_r(o) \in \mathcal{H}$ an hypothesis.
Ensure: $g \in \mathcal{H}$ a generalisation of e and h ($g \succeq h$ and $g \succeq e$).

1: $c = o \xrightarrow{*} e$ {a shortest path}
2: let x, y integers and o' a word such that $o \xrightarrow{x} o' \xrightarrow{y} e$
3: $x = d(o, o')$, $y = d(o', e)$, $x + y = d(o, e)$
4: $r' = \max(r + x, y)$
5: **return** $B_{r'}(o')$

About GBALL

- no experimental difference between strategies to chose o';
- properties of this algorithm?

Order dependency

Order dependency

Order dependency

Order dependency

Order dependency

Dependency on the presentation order of examples, interesting property for ensemble methods!

Monotonic generalisation...

If $g = \text{GBALL}(e, h)$ then $g \succeq e$ and $g \succeq h$.

Proof.

Recall of the algorithm: $x + y = d(o, e)$ and $r' = \max(r + x, y)$.

- $g \succeq e$ because $d(o', e) = y \leq r'$ and therefore $e \in B_{r'}(o')$;
- $g \succeq h$ because for each word $w \in B_r(o)$ and by triangular inequality:

$$d(o', w) \leq d(o', o) + d(o, w)$$

therefore $d(o', w) \leq x + r \leq r'$ and $w \in B_{r'}(o')$.

□

... but not least general

Counterexample

- Let $E = [a, b]$;
- first hypothesis = first example = $h = B_0(a)$;
- two possibilities to choose o' on the path $a \xrightarrow{1} b$;
- either $\text{GBALL}(h, b) = B_1(a)$, or $\text{GBALL}(h, b) = B_1(b)$;
- but the ball $B_1(\lambda)$ contains E;
- and is more specific than the computed hypotheses: $B_1(a) \succeq B_1(\lambda)$ and $B_1(b) \succeq B_1(\lambda)$.

our method generalises a little too much...

Non monotonic correct generalisation

Example

Recall: CG generalises using GBALL and checks correction.

- Let $E^+ = [\lambda, b, a]$ and $E^- = [bb]$
- CG with GBALL and $x = 1$ provide successively:
  - $B_0(\lambda)$ (initial hypothesis);
  - $B_1(b)$ (rejected because $B_1(b)$ accepts bb);
  - $B_1(a)$ (validated).
- $B_1(a) \succeq b$ while the adding of b has been previously rejected.

Implies less efficient AdaBoost implementation...

Non monotonic correct generalisation

Example

Recall: CG generalises using GBALL and checks correction.

- Let $E^+ = [\lambda, b, a]$ and $E^- = [bb]$
- CG with GBALL and $x = 1$ provide successively:
  - $B_0(\lambda)$ (initial hypothesis);
  - $B_1(b)$ (rejected because $B_1(b)$ accepts bb);
  - $B_1(a)$ (validated).
- $B_1(a) \succeq b$ while the adding of b has been previously rejected.

Implies less efficient AdaBoost implementation...

Summary

Balls have multiple *lgg*. GBALL is order dependant and monotonic but does not give a *lgg* ball and leads to a non monotonic CG.

Outline of the talk

Experiments (1): UCI sequential datasets [Asuncion and Newman, 2007]

Classical GI methods against *ensemble methods*.

Datasets and protocol

- tic-tac-toe, badges, us-first-names, promoters, splice;
- 10-fold cross validation, 90% of data for learning.

Competitors

- Majority, RPNI, TraxBar, Red-Blue;
- GloBoost + lgg-tssi and GloBoost + lgg-zr
- GloBoost + gball and random strategy.

Results on UCI datasets

	tic-tac-toe	badges	promoters	first-name	splice
Majority	65.34 %	**71.43 %**	50.00 %	81.62 %	50.26 %
RPNI	91.13 %	62.24 %	-	81.42 %	-
TraxBar	90.81 %	57.48 %	56.60 %	81.37 %	**58.33 %**
Red-Blue	**93.89 %**	61.09 %	**63.02 %**	**82.83 %**	54.65 %
Glgg-tssi $_{1\,000}$	91.47 %	**72.69 %**	**61.13 %**	**89.50 %**	**78.07 %**
Glgg-zr $_{1\,000}$	**98.36 %**	71.43 %	50.00 %	83.07 %	-
Ggball $_{1\,000}$	92.95 %	80.41 %	87.63 %	87.10 %	93.76 %
Ggball $_{10\,000}$	94.69 %	**81.39 %**	88.43 %	88.80 %	**95.63 %**
Ggball $_{100\,000}$	**94.96 %**	81.36 %	**89.08 %**	**89.06 %**	95.62 %

Observations: combinations are better than classical methods

- reversible: too specific;
- automata: too slow.
- balls: fastness+diversity;
- balls: good in genomic.

Experiments (2): handwritten digit classification

VOLATA and balls on a real problem.

Datasets and protocol

- *Nist special database 3*;
- 10 classes, 10 568 examples;
- 10-fold cross validation, 10% of data for learning;

Competitors

- competitor: SeDiL [Boyer et al., 2008].
- GloBoost + gball and random strategy.

Experiments (2): handwritten digit classification

VOLATA and balls on a real problem.

Datasets and protocol

- *Nist special database 3*;
- 10 classes, 10 568 examples;
- 10-fold cross validation, 10% of data for learning;

Competitors

- competitor: SeDiL [Boyer et al., 2008].
- GloBoost + gball and random strategy.

Results

SeDiL	**95.86 %**
GGBALL $_{1\,000}$	93.81 %
GGBALL $_{10\,000}$	95.93 %
GGBALL $_{100\,000}$	**96.32 %**

Good results for VOLATA and balls!

Summary and perspectives

Automata

- unique *lgg* for two classes;
- corresponding LGG available and embedded in VOLATA;
- not fast, very specific;
- other language classes with or without unique least general generalisations?
- classes that reach regular languages by union.

Balls

- GBALL usable by VOLATA;
- fast generalisations and fast classifications;
- great diversity, many balls;
- good experimental results;
- real applications;
- study hollow balls.

Thank you for your attention. Any questions?

A ball of first-names

back to conclusion

A learned ball: $B_7(LRLRTSVKCA)$
contains 346 female first-names (all at distance 7) and 0 male:

- ALBERTHA
- BERTA
- DRUSILLA
- ELSA
- FRANCESCA
- HORTENSIA
- JESSIKA
- KRYSTINA
- LORENZA
- MIRTA
- NERISSA
- OCTAVIA
- PARTICIA
- REBBECA
- SYLVIA
- TERESSA
- URSULA
- VERONICA
- etc.

A ball of first-names

back to conclusion

A learned ball: $B_7(LRLRTSVKCA)$
contains 346 female first-names (all at distance 7) and 0 male:

- ALBERTHA
- BERTA
- DRUSILLA
- ELSA
- FRANCESCA
- HORTENSIA
- JESSIKA
- KRYSTINA
- LORENZA
- MIRTA
- NERISSA
- OCTAVIA
- PARTICIA
- REBBECA
- SYLVIA
- TERESSA
- URSULA
- VERONICA
- etc.

- average distance between examples : 4.9
- average min distance between examples : 1.2
- maximal min distance between examples : 3.0

Bibliographie I


Angluin, D. (1982).
Inference of reversible languages.
Journal of the ACM, 29(3):741–765.


Asuncion, A. and Newman, D. (2007).
UCI machine learning repository.


Boyer, L., Esposito, Y., Habrard, A., Oncina, J., and Sebban, J. (2008).
Sedil: Software for edit distance learning.
In Daelemans, W., Goethals, B., and Morik, K., editors, *Proceedings of the 19th European Conference on Machine Learning*, pages 672–677. Springer.

Bibliographie II


de la Higuera, C., Janodet, J.-C., and Tantini, F. (2008).
Learning languages from bounded resources: The case of the dfa and the balls of strings.
In Clark, A., Coste, F., and Miclet, L., editors, *Proceedings of the 9th International Conference in Grammatical Inference*, volume 5278 of *Lecture Notes in Artificial Intelligence*, pages 43–56. Springer.


García, P. and Vidal, E. (1990).
Inference of k-testable languages in the strict sense and application to syntactic pattern recognition.
IEEE Trans. Pattern Anal. Mach. Intell., 12(9):920–925.