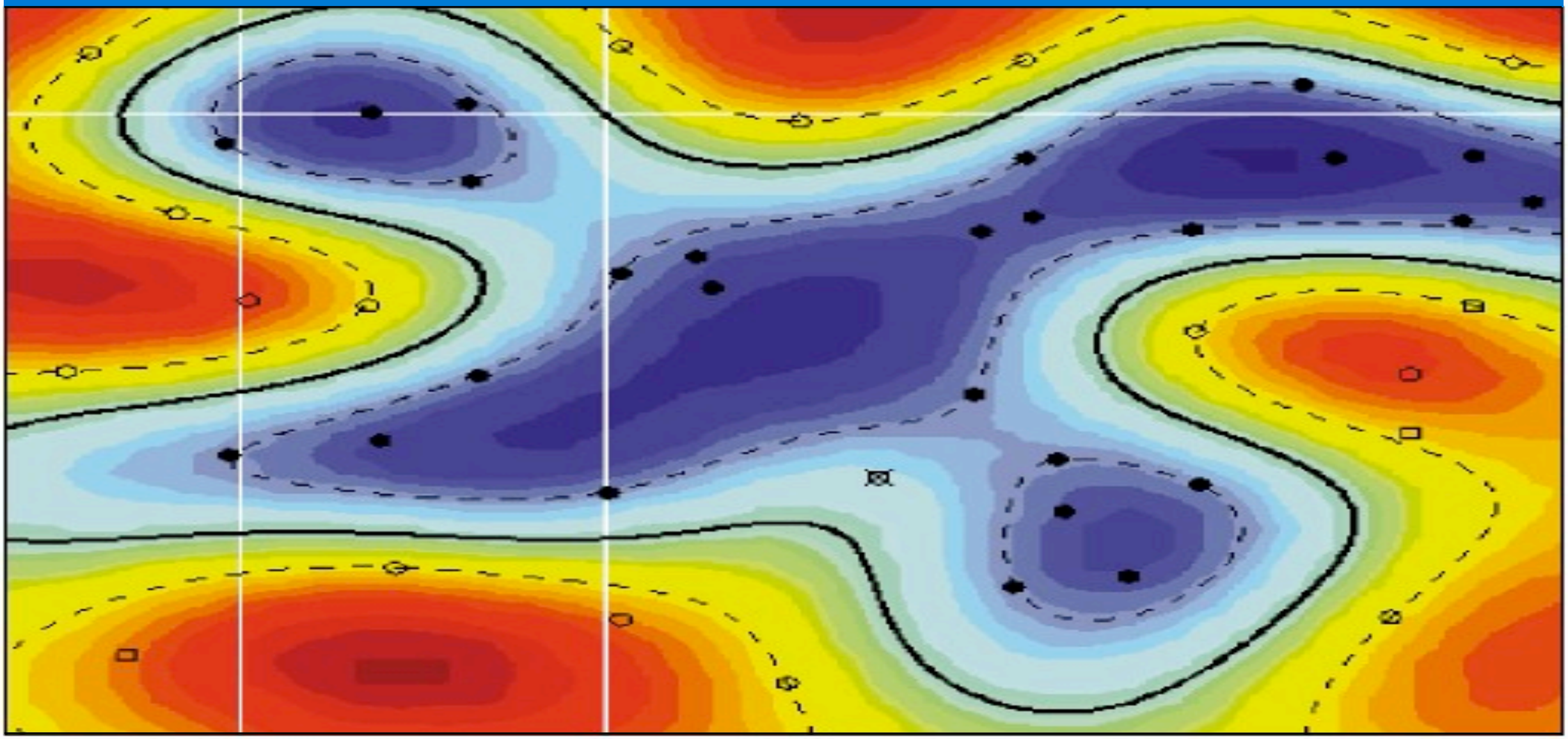
Learning with Kernels and Logical Representations

Paolo Frasconi and Andrea Passerini

TECHNISCHE UNIVERSITÄT DARMSTADT



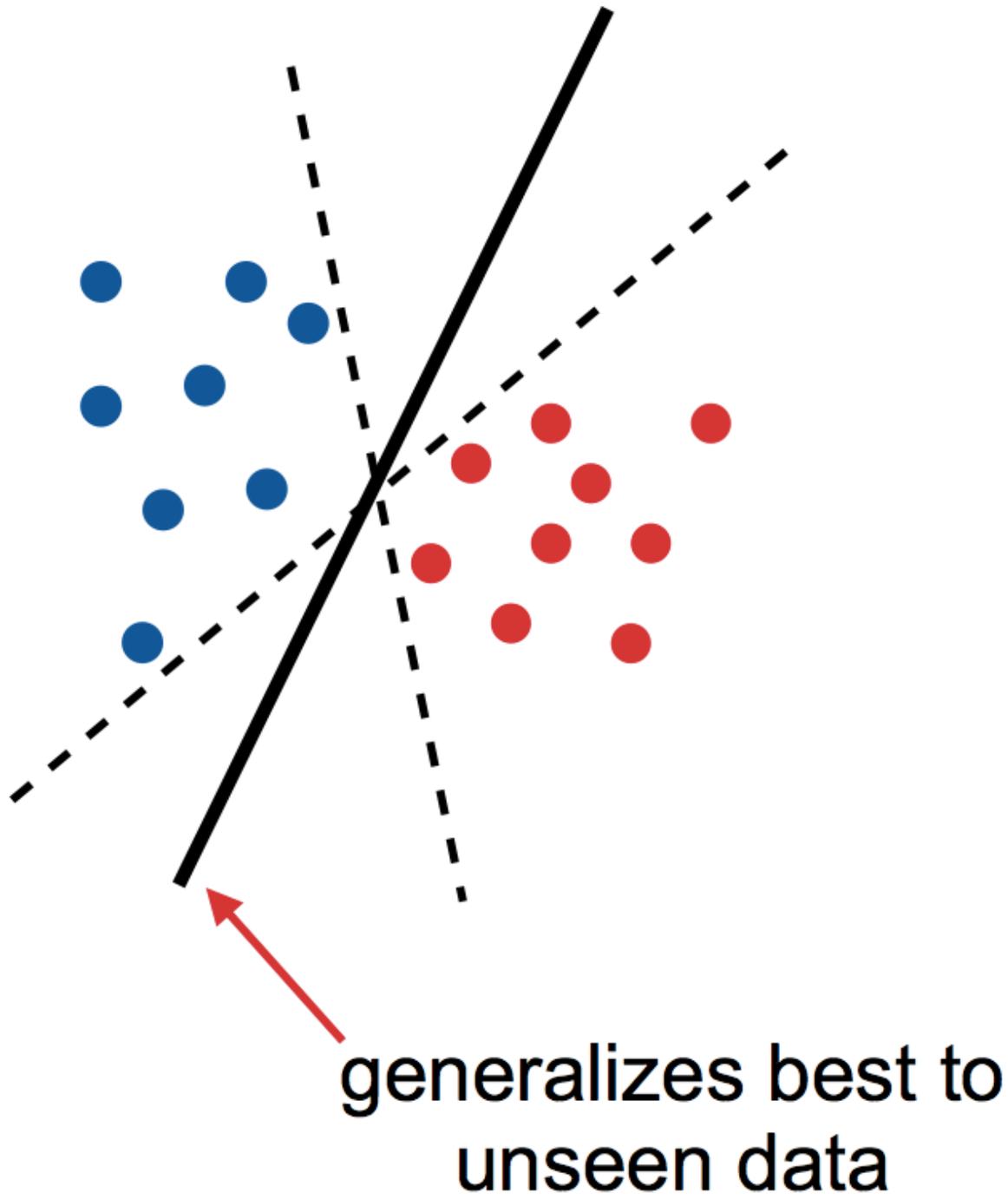
Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

- Basics
- Motivation
- Kernels on Prolog Ground Terms
- Kernels and Mereotopological Relations
- Kernels on Prolog Proof Trees
- kFOIL
- Conclusion

Kernel Methods (using the example of SVM)

TECHNISCHE UNIVERSITÄT DARMSTADT



The Kernel Trick

TECHNISCHE UNIVERSITÄT DARMSTADT

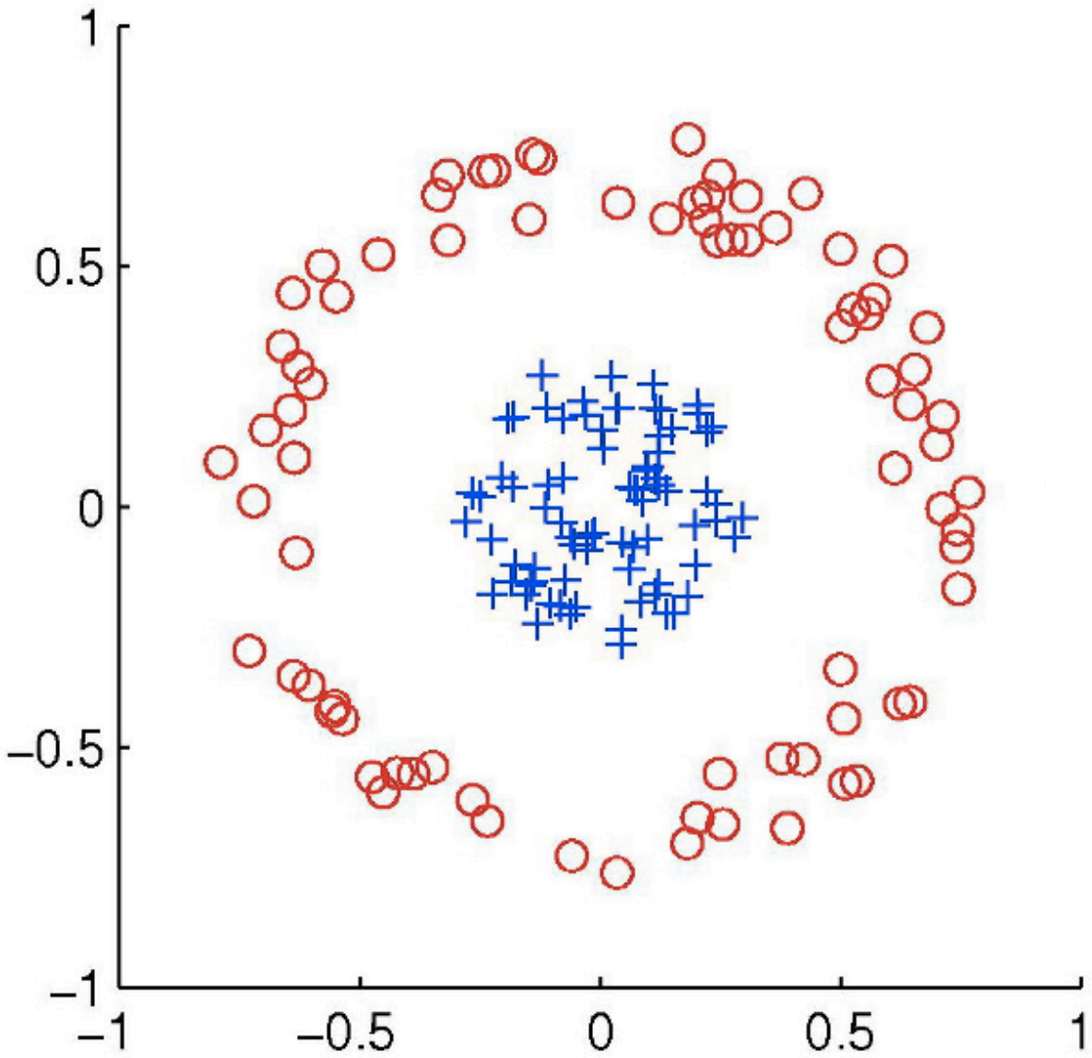


Image Source: http://omega.albany.edu:8008/machine-learning-dir/notes-dir/ker1/ker1.pdf

The Kernel Trick

TECHNISCHE UNIVERSITÄT DARMSTADT

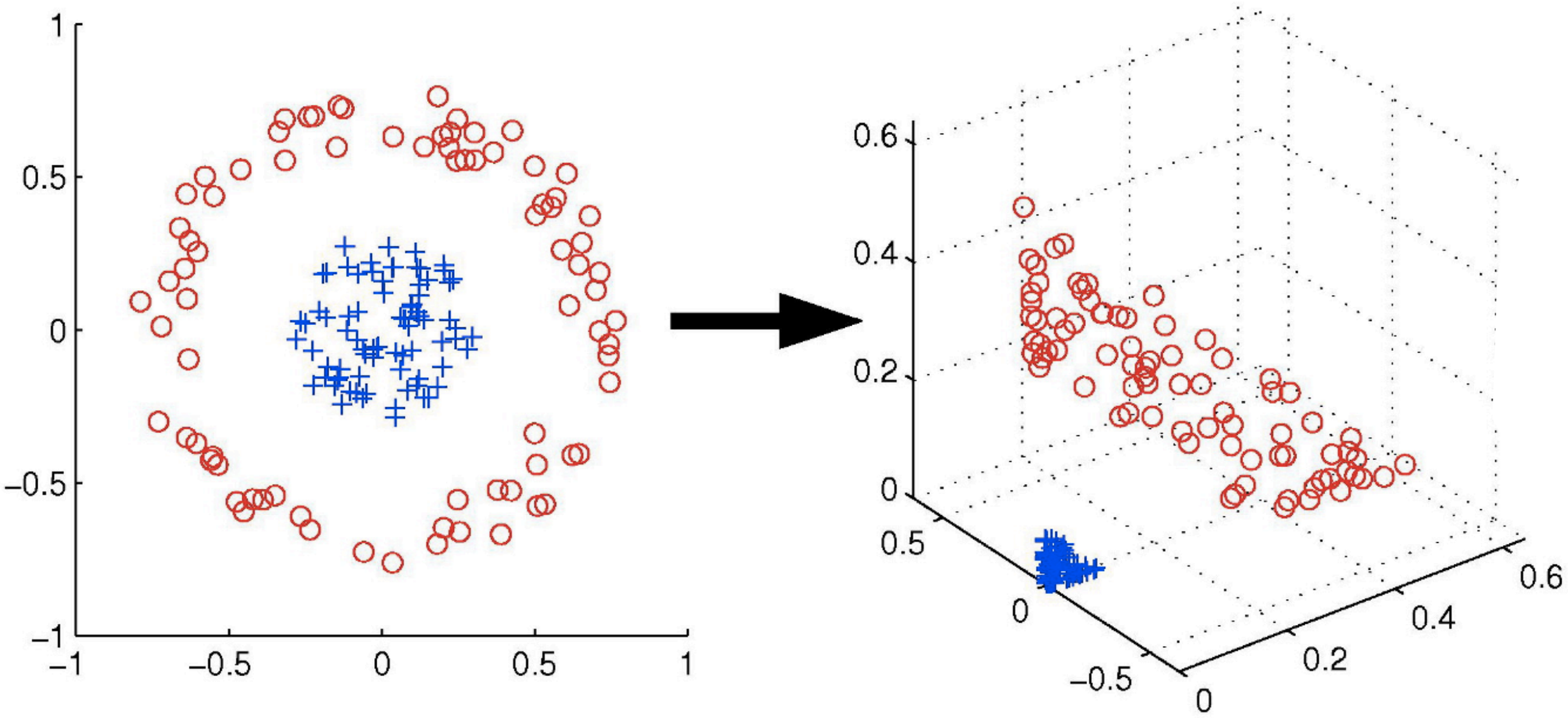
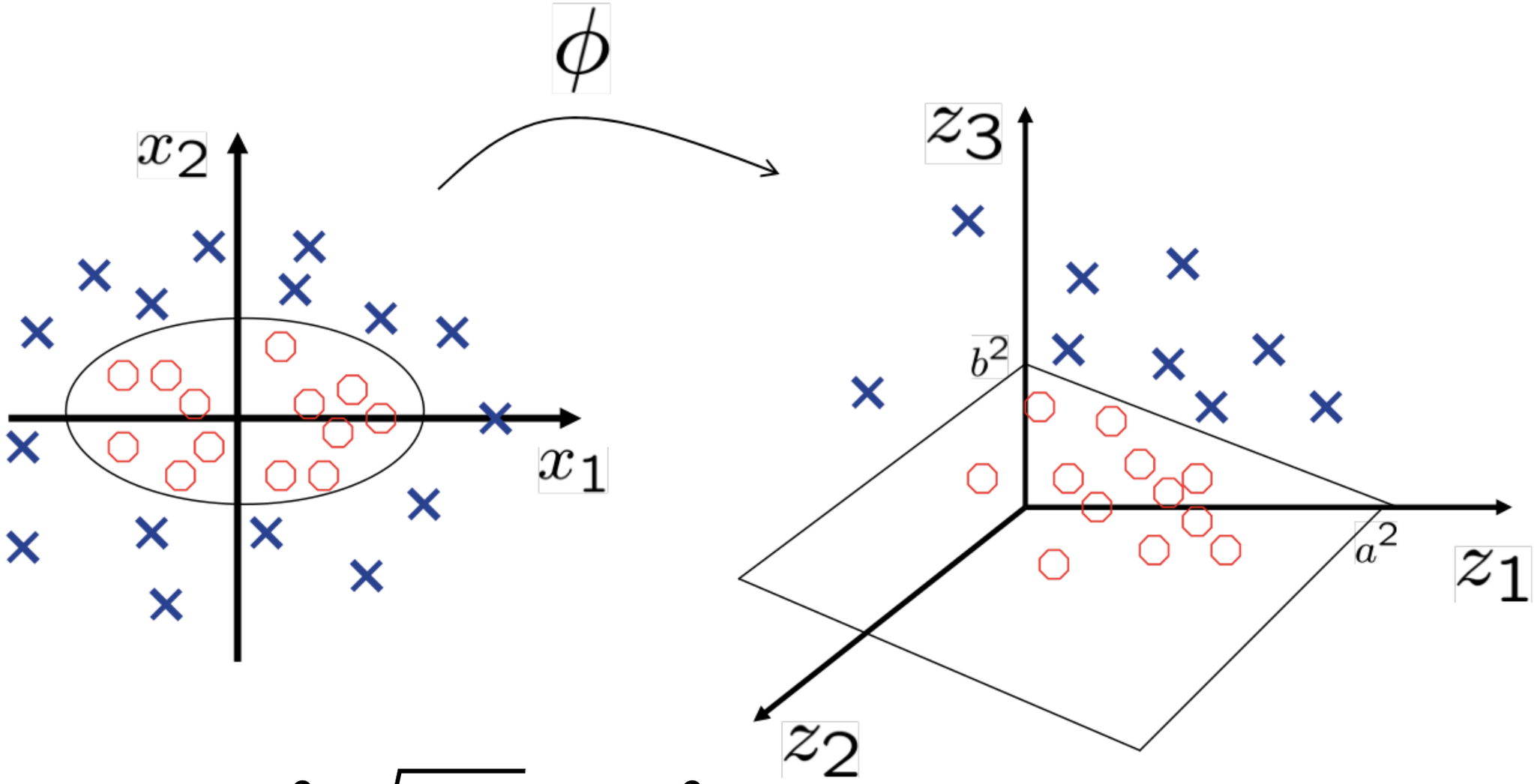


Image Source: http://omega.albany.edu:8008/machine-learning-dir/notes-dir/ker1/ker1.pdf

The Kernel Trick(2)

TECHNISCHE UNIVERSITÄT DARMSTADT



$$\phi : (x_1, x_2) \rightarrow (x_1^2, \sqrt{2x_1}x_2, x_2^2)$$

$$K(x, w) = \langle \phi(x), \phi(w) \rangle$$

Typed Terms in Prolog

TECHNISCHE UNIVERSITÄT DARMSTADT

- To introduce typed Terms in Prolog we need:
  - Set of constructors $\mathcal{T}$
    - consisting of at least the nullary constructor $\perp \in \mathcal{T}$
    - additional m-ary constructors $\tau\alpha_1,...,\alpha_m \in \mathcal{T}$ with type variables $\alpha_1,...,\alpha_m$
  - Set of Functions $\mathcal{F}$
    - with signatures $\tau_1 \times,...,\times \tau_n \mapsto \tau'$ where $n \leq 0$ is the number of arguments, $\tau_1,...,\tau_n \in \mathcal{T}$ their types and $\tau' \in \mathcal{T}$ the type of the result
    - functions of arity 0 have the signature $\perp \mapsto \tau'$ and can be therefore interpreted as constants
  - Predicates $\mathcal{P}$
    - the type signature of a predicate of arity n has the form $\tau_1 \times,...,\times \tau_n \mapsto \Omega$ where Ω is the type of booleans
    - we write $t : \tau$ to assert that t is a term of type τ

Typed Terms in Prolog(2): Example

TECHNISCHE UNIVERSITÄT DARMSTADT

- Terms $\mathcal{T}$={man, woman}
- Functions $\mathcal{F}$={john : man, mary : woman}
- Predicates $\mathcal{P}$={married : Pred(man × woman)}

- Exemplary programs:
  - legal:
    - married(john, mary)
  - illegal:
    - spouse(X,Y) ⟵ married(X,Y).
      spouse(X,Y) ⟵ married(Y,X).

Motivation

TECHNISCHE UNIVERSITÄT DARMSTADT

- already available background knowledge about a domain that is described in a logical representation can be used
- kernel methods are efficiently computable
- kernel methods can be used for classification, regression, ranking, novelty detection, clustering and principal component analysis
- SVM converges to the optimal function vs. other probabilistic ILP approaches that are based on generative modelling may converge to a sub-optimal error if the underlying assumptions are wrong

Kernels on Prolog Ground Terms: Untyped Terms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Let $\mathcal{C}$ be a set of constants and $\mathcal{F}$ a set of functors and denote by $\mathcal{U}$ the corresponding Herbrand universe
- Let $f/^n \in \mathcal{F}$ denote a functor having the name f and arity n
- The Kernel between two terms t and s is a function $\mathrm{K} : \mathcal{U} \times \mathcal{U} \mapsto \mathcal{R}$ inductively defined as follows:
  - if $s \in \mathcal{C}$ and $t \in \mathcal{C}$ then $\mathrm{K}(s,t)=\kappa(s,t)$ where $\kappa:\mathcal{C} \times \mathcal{C} \mapsto \mathcal{R}$ is a valid kernel on constants
  - else if s and t are compound terms and have different functors, i.e., $s=f(s_1,\ldots s_n)$ and $t=g(t_1,\ldots t_n)$, then $\mathrm{K}(s,t)=\iota(f/^n,g/^m)$ where $\iota:\mathcal{F} \times \mathcal{F} \mapsto \mathcal{R}$ is a valid kernel on functors
  - else if s and t are compound terms and have the same functor, i.e., $s=f(s_1,\ldots s_n)$ and $t=f(t_1,\ldots t_n)$, then $\mathrm{K}(s,t)=\iota(f/^n,f/^n)+\sum_{i=1}^{n}\mathrm{K}(s_i,t_i)$
  - in all other cases K(s,t)=0

Kernels on Prolog Ground Terms: Typed Terms

TECHNISCHE UNIVERSITÄT DARMSTADT

- The Kernel between two typed terms t and s is inductively defined as follows:
  - if $s \in \mathcal{C}$ and $t \in \mathcal{C}$, s : τ, t : τ then $\mathrm{K}(s,t)=\kappa_\tau(s,t)$ where $\kappa_\tau: \mathcal{C} \times \mathcal{C} \mapsto \mathcal{R}$ is a valid kernel on constants of type τ
  - else if s and t are compound terms that have the same type and different functors or signatures, i.e., $s=f(s_1,\ldots s_n)$ and $t=g(t_1,\ldots t_n)$, $s : \sigma_1 \times,\ldots,\times \sigma_n \mapsto \tau'$, $t : \tau_1 \times,\ldots,\times \tau_m \mapsto \tau'$, then $\mathrm{K}(s,t)=\iota_{\tau'}(f^{/n},g^{/m})$ where $\iota_{\tau'}: \mathcal{F} \times \mathcal{F} \mapsto \mathcal{R}$ is a valid kernel on functors that construct terms of type τ'
  - else if s and t are compound terms and have the same functor and type signature, i.e., $s=f(s_1,\ldots s_n)$ and $t=f(t_1,\ldots t_n)$ and $s,t : \tau_1 \times,\ldots,\times \tau_n \mapsto \tau'$, then

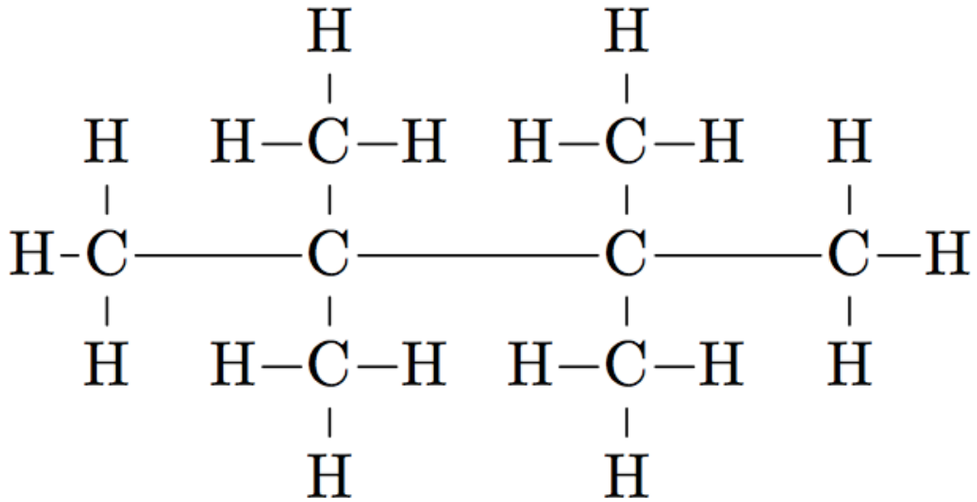
$$K(s,t) = \begin{cases} \kappa_{\tau_1 \times,\ldots,\times\tau_n \mapsto \tau'}(s,t) \\ \qquad \text{if } (\tau_1 \times,\ldots,\times\tau_n \mapsto \tau') \in \overline{\mathcal{T}} \\ \iota_{\tau'}(f^{/n}, f^{/n}) + \sum_{i=1}^{n} K(s_i,t_i) \quad \text{otherwise} \end{cases}$$

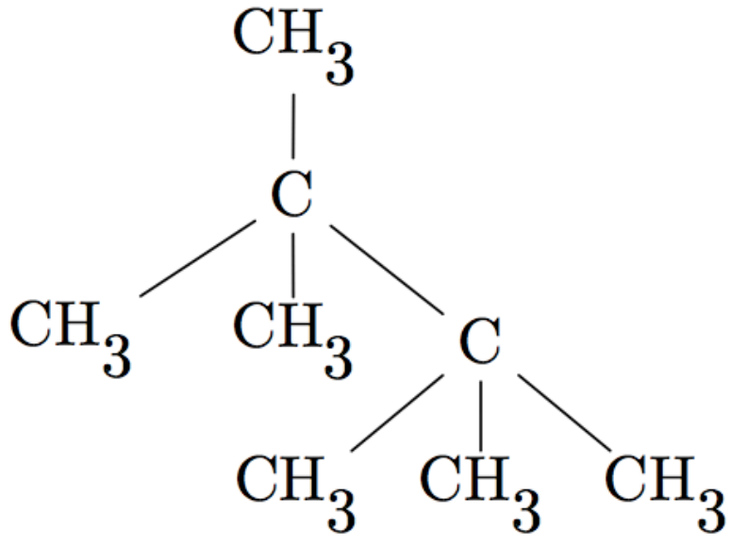
where $\kappa_{\tau_1 \times,\ldots,\times\tau_n \mapsto \tau'}: \mathcal{U} \times \mathcal{U} \mapsto \mathcal{R}$ is a valid kernel on terms having a distinguished type signature $\tau_1 \times,\ldots,\times\tau_n \mapsto \tau' \in \underline{\mathcal{T}}$

  - in all other cases K(s,t)=0

Kernels on Prolog Ground Terms: Example Boiling Point of Alkanes

TECHNISCHE UNIVERSITÄT DARMSTADT





```
c(h,h,h,c(c(h,h,h),c(h,h,h),c(c(h,h,h),c(h,h,h),c(h,h,h))))
```

- K: kernel on untyped terms by using comparing functors (carbon atoms) and the null function for comparing constants (hydrogen atoms).

  The kernel counts the number of carbon atoms in corresponding positions of two alkanes.
- avg. mean square error: 4.6°C

- K‘: As an additional source of information, we extracted the depths of the trees representing the molecules, and summed their product to the term kernel.
- avg. mean square error: 3.8°C

Mereotopology

TECHNISCHE UNIVERSITÄT DARMSTADT

- Two special predicates:
  - $\preceq_P$ „part of“
    - x $\preceq_P$ y declares x to be a part of y
    - $\preceq_P$ is partially ordered:

$$x \preceq_P x$$

$$x \preceq_P y \wedge y \preceq_P x \Rightarrow y =_P x$$

$$x \preceq_P y \wedge y \preceq_P z \Rightarrow x \preceq_P z$$

  - Connected
    - Connected(x,y) declares x to be connected to y

$$\mathrm{Connected}(x, x)$$

$$\mathrm{Connected}(x, y) \Rightarrow \mathrm{Connected}(y, x)$$

$$x \preceq_P y \Rightarrow \forall z.(\mathrm{Connected}(z, x) \Rightarrow \mathrm{Connected}(z, y))$$

Mereotopological Relations: Examples

TECHNISCHE UNIVERSITÄT DARMSTADT

- We denote by $\mathcal{M}$ the set of declared Meretopological Relations.
- Examples:

i) the proper parts of x: $\mathcal{R}_P(x) = \{y : y \prec_P x\}$;

ii) the connected proper parts of x: $\mathcal{R}_C(x) = \{(y, z) : y \prec_P x \wedge z \prec_P x \wedge \text{Connected}(y, z)\}$;

The Kernel on Parts K_P

TECHNISCHE UNIVERSITÄT DARMSTADT

The kernel on parts, denoted K_P, is naturally defined as the *set kernel* between the sets of proper parts:

$$K_P(x, x') = \sum_{y \in \mathcal{P}(x)} \sum_{y' \in \mathcal{P}(x')} k_P(y, y') \tag{20}$$

- Types can be used to fine-tune the definition of k_P:

$$k_P(y, y') = \begin{cases} \iota(y, y') & \text{if } y =_T y' \text{ and } y, y' \text{ are atomic objects;} \\ K_P(y, y') + \iota_(y, y') & \text{if } y =_T y' \text{ and } y, y' \text{ are non atomic objects;} \\ 0 & \text{otherwise (i.e. } y \neq_T y'\text{).} \end{cases}$$

The Kernel K_C, K_I and K_L

TECHNISCHE UNIVERSITÄT DARMSTADT

The kernel on connected parts compares the sets of objects $\mathcal{R}_C(x)$ and $\mathcal{R}_C(x')$ as follows:

$$K_C(x,x') = \sum_{(y,z)\in\mathcal{R}_C(x)} \sum_{(y',z')\in\mathcal{R}_C(x')} K_P(y,y') \cdot K_P(z,z').$$

The kernel on overlapping parts compares the sets of objects $\mathcal{R}_I(x)$ and $\mathcal{R}_I(x')$ as follows:

$$K_I(x,x') = \sum_{(y,z,w)\in\mathcal{R}_I(x)} \sum_{(y',z',w')\in\mathcal{R}_L(x')} K_P(w,w')\delta(y,y')\delta(z,z')$$

where $\delta(x,y) = 1$ if x and y have the same type and 0 otherwise. The kernel $K_L(x,x')$ on externally connected parts is defined in a similar way:

$$K_L(x,x') = \sum_{(y,z,u,v)\in\mathcal{R}_L(x)} \sum_{(y',z',u',v')\in\mathcal{R}_L(x')} K_P(u,u')K_P(v,v')\delta(y,y')\delta(z,z').$$

General Kernel for $\mathcal{M}$

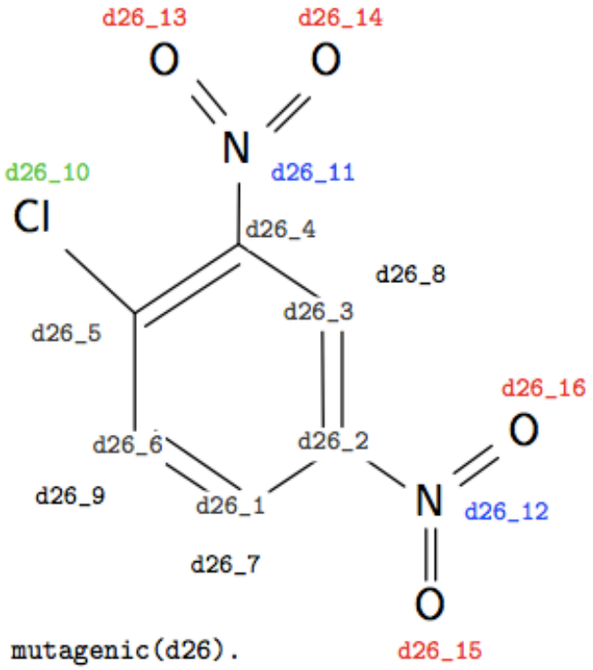
TECHNISCHE UNIVERSITÄT DARMSTADT

Given a set $\mathcal{M}$ of MRs (such as those defined above), the final form of the kernel is

$$K(x, x') = \sum_{M \in \mathcal{M}} K_M(x, x').$$

Kernels and Meretopological Relations: Example Mutagenesis

TECHNISCHE UNIVERSITÄT DARMSTADT



mutagenic(d26).

Extensional predicates

```
atm(d26,d26_1,c,22,-0.093).          bond(d26,d26_1,d26_2,7).
atm(d26,d26_2,c,22,-0.093).          bond(d26,d26_2,d26_3,7).
atm(d26,d26_3,c,22,-0.093).          bond(d26,d26_3,d26_4,7).
atm(d26,d26_4,c,22,-0.093).          bond(d26,d26_4,d26_5,7).
atm(d26,d26_5,c,22,-0.093).          bond(d26,d26_5,d26_6,7).
atm(d26,d26_6,c,22,-0.093).          bond(d26,d26_6,d26_1,7).
atm(d26,d26_7,h,3,0.167).            bond(d26,d26_1,d26_7,1).
atm(d26,d26_8,h,3,0.167).            bond(d26,d26_3,d26_8,1).
atm(d26,d26_9,h,3,0.167).            bond(d26,d26_6,d26_9,1).
atm(d26,d26_10,cl,93,-0.163).        bond(d26,d26_10,d26_5,1).
atm(d26,d26_11,n,38,0.836).          bond(d26,d26_4,d26_11,1).
atm(d26,d26_12,n,38,0.836).          bond(d26,d26_2,d26_12,1).
atm(d26,d26_13,o,40,-0.363).         bond(d26,d26_13,d26_11,2).
atm(d26,d26_14,o,40,-0.363).         bond(d26,d26_11,d26_14,2).
atm(d26,d26_15,o,40,-0.363).         bond(d26,d26_15,d26_12,2).
atm(d26,d26_16,o,40,-0.363).         bond(d26,d26_12,d26_16,2).
```

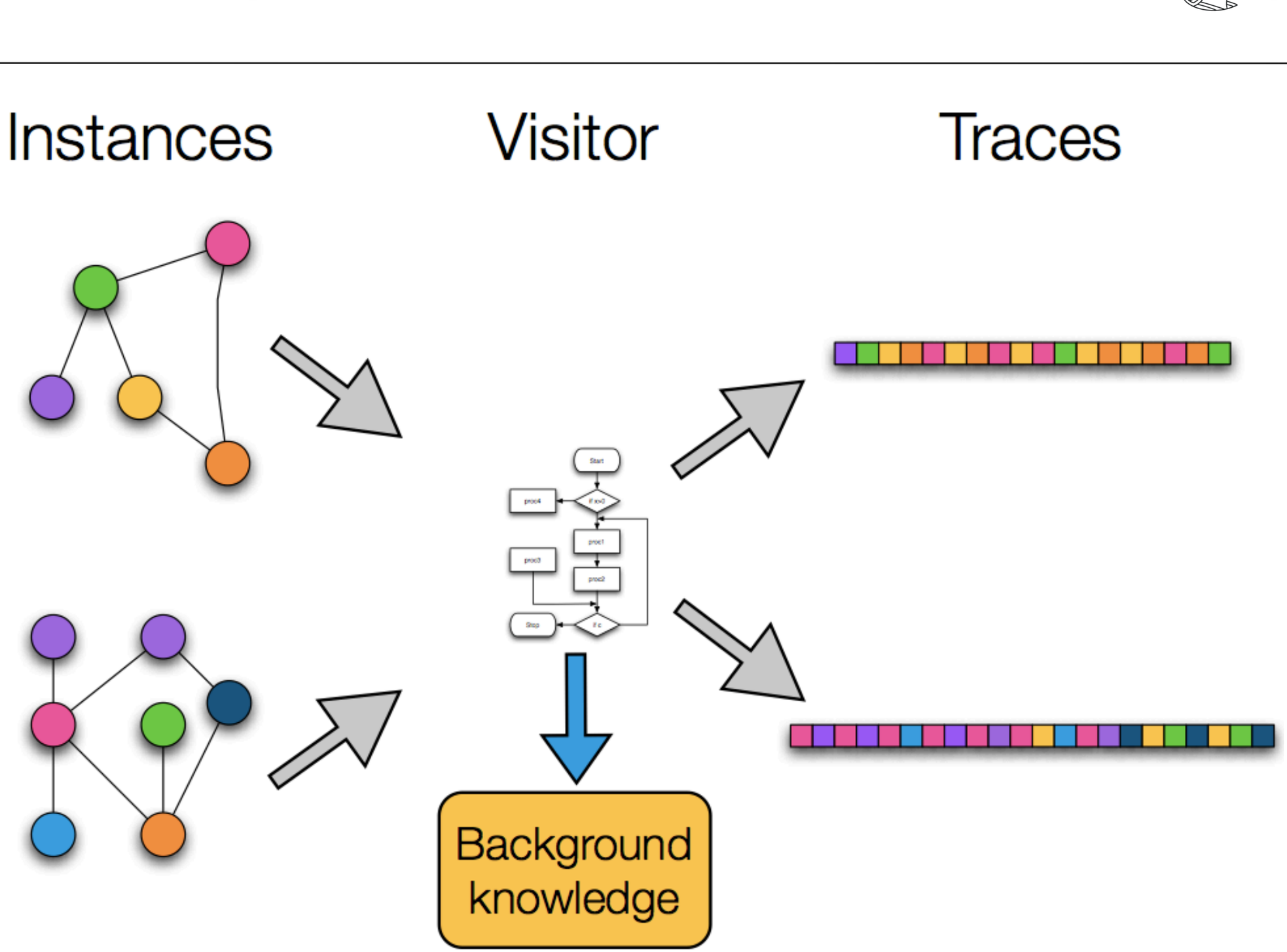
Intensional predicates

```
nitro(Drug,[Atom0,Atom1,Atom2,Atom3]) :-
   atm(Drug,Atom1,n,38,_),
   bondd(Drug,Atom0,Atom1,1),
   bondd(Drug,Atom1,Atom2,2),
   atm(Drug,Atom2,o,40,_),
   bondd(Drug,Atom1,Atom3,2),
   Atom3 @> Atom2,
   atm(Drug,Atom3,o,40,_).


bondd(Drug,Atom1,Atom2,Type) :-
   bond(Drug,Atom1,Atom2,Type).
bondd(Drug,Atom1,Atom2,Type) :-
   bond(Drug,Atom2,Atom1,Type).
```

```
benzene(Drug,Ring_list) :-
   atoms(Drug,6,Atom_list,[c,c,c,c,c,c]),
   ring6(Drug,Atom_list,Ring_list,[7,7,7,7,7,7]).

ring6(Drug,[Atom1|List],[Atom1,Atom2,Atom4,Atom6,Atom5,Atom3],
   [Type1,Type2,Type3,Type4,Type5,Type6]) :-
   bondd(Drug,Atom1,Atom2,Type1),   memberchk(Atom2,[Atom1|List]),
   bondd(Drug,Atom1,Atom3,Type2),   memberchk(Atom3,[Atom1|List]),
   Atom3 @> Atom2,
   bondd(Drug,Atom2,Atom4,Type3),   Atom4 \== Atom1,
   memberchk(Atom4,[Atom1|List]),
   bondd(Drug,Atom3,Atom5,Type4),   Atom5 \== Atom1,
   memberchk(Atom5,[Atom1|List]),
   bondd(Drug,Atom4,Atom6,Type5),   Atom6 \== Atom2,
   memberchk(Atom6,[Atom1|List]),
   bondd(Drug,Atom5,Atom6,Type6),   Atom6 \== Atom3.
```

Kernels on Prolog Proof Trees

TECHNISCHE UNIVERSITÄT DARMSTADT



Kernels on Prolog Proof Trees (2)

TECHNISCHE UNIVERSITÄT DARMSTADT

- The learning setting can be briefly sketched as follows:
  - the learner is given:
    - a data set $\mathcal{D}=\{(x_1,y_1),\ldots (x_m,y_m)\}$
    - background knowledge $\mathcal{B}$
    - a visitor program $\mathcal{V}$
  - for each instance x_i, a trace Tx_i is obtained by running the visitor program
  - a kernel machine(e.g., an SVM) is then trained to form the function $f : \mathcal{X} \mapsto \mathcal{Y}$ defined as

$$f(x) = \sum_{i=1}^{m} c_i K(T_{x_i}, T_x)$$

$$T_x = [T_{1,x}, \ldots, T_{N,x}]$$

Trace x

$$T_{l,x} = \{T_{l1,x}, \ldots, T_{ls_{l,x},x}\}$$

Proof trees of visitor l in trace x

Proof Tree Pruning

TECHNISCHE UNIVERSITÄT DARMSTADT

```
atoms(X,[]).
atoms(X,[H|T]):-
    atm(X,H,_,_,_),
    atoms(X,T).
```

```
visit_benzene(X):-
    benzene(X,Atoms),
    atoms(X,Atoms).
```

- the proof tree could be pruned at benzene(X,Atoms) by including the following fact in the visitor program

```
leaf(benzene(_,_)).
```

- this will reduce the complexity of the feature space associated with the kernel

Kernels on Traces

TECHNISCHE UNIVERSITÄT DARMSTADT

- kernel over program traces

$$K(T_x, T_z) = \sum_{l=1}^{N} K_l(T_{l,x}, T_{l,z}).$$

- kernel over visitors

$$K_l(T_{l,x}, T_{l,z}) = \sum_{p=1}^{s_{l,x}} \sum_{q=1}^{s_{l,z}} K_{tree}(T_{lp,x}, T_{lq,z}).$$

- kernel over proof trees
  - K_{tree} can be implemented by representing proof trees as typed Prolog ground terms (see Kernels on Prolog ground terms)

Kernels on Prolog Proof Trees: Example Bongard Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Background knowledge

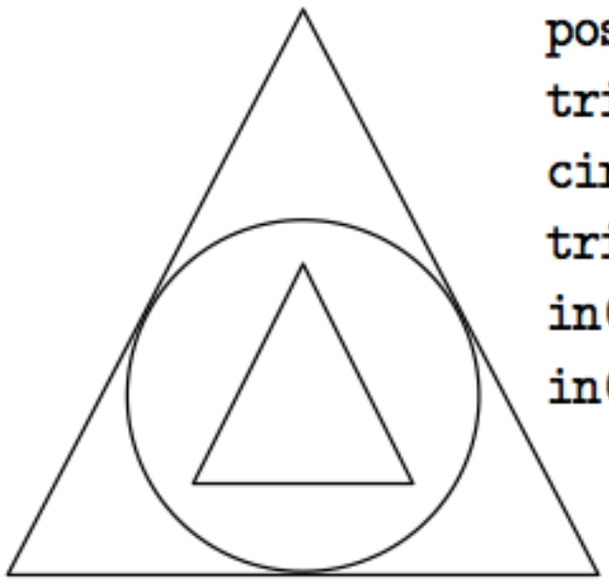
```
inside(X,A,B):- in(X,A,B).                    % clause nr 1
inside(X,A,B):-                               % clause nr 2
    in(X,A,C),
    inside(X,C,B).
polygon(X,A) :-  triangle(X,A).               % clause nr 3
polygon(X,A) :-  rectangle(X,A).              % clause nr 4
polygon(X,A) :-  circle(X,A).                 % clause nr 5
```

- Visitor program

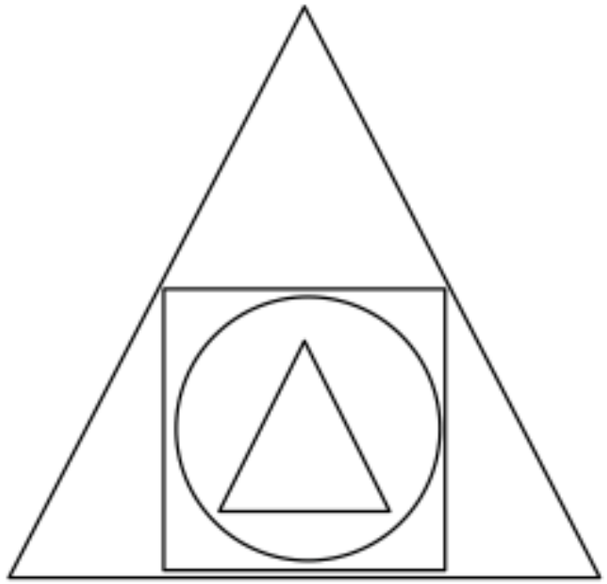
```
visit(X):-                                    % clause nr 6
    inside(X,A,B),polygon(X,A),polygon(X,B).
```

Kernels on Prolog Proof Trees: Example Bongard Problem(2)

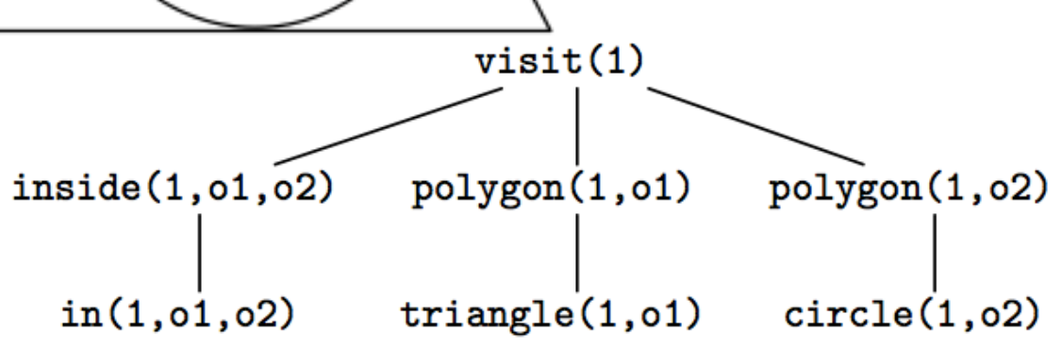
TECHNISCHE UNIVERSITÄT DARMSTADT

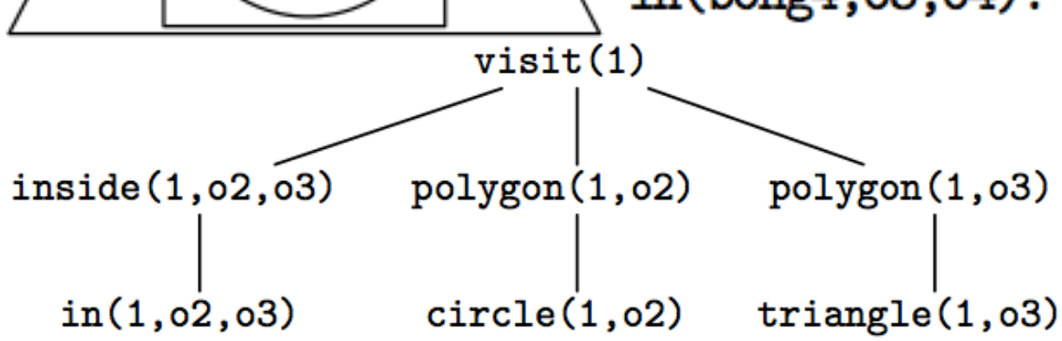


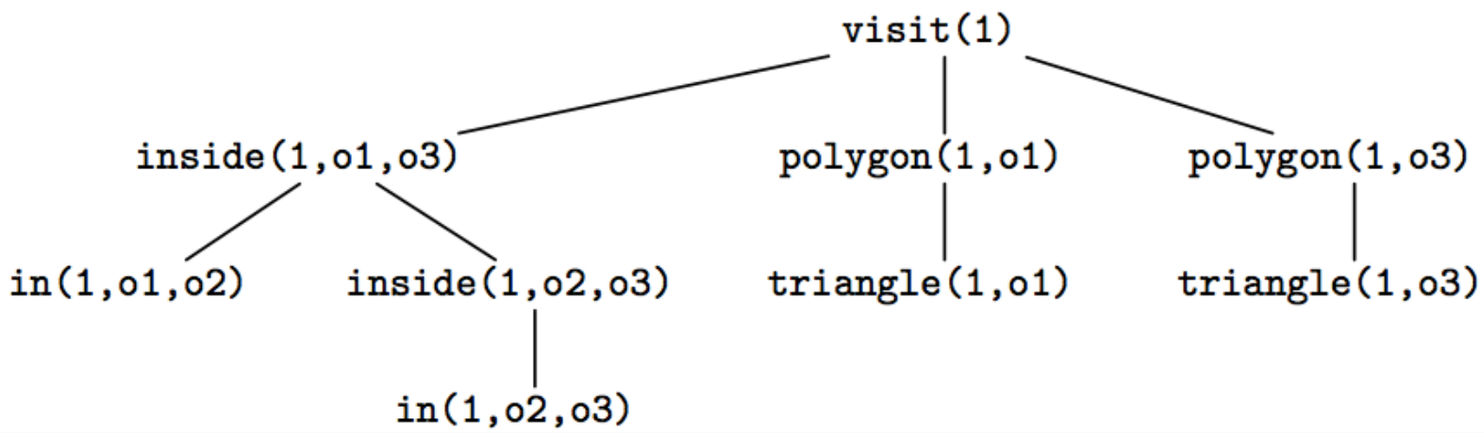
```
positive(bong1).
triangle(bong1,o1).
circle(bong1,o2).
triangle(bong1,o3).
in(bong1,o1,o2).
in(bong1,o2,o3).
```



```
negative(bong4).
triangle(bong4,o1).
rectangle(bong4,o2).
circle(bong4,o3).
triangle(bong4,o4).
in(bong4,o1,o2).
in(bong4,o2,o3).
in(bong4,o3,o4).
```







Kernels on Prolog Proof Trees: Example Bongard Problem(3)

TECHNISCHE UNIVERSITÄT DARMSTADT

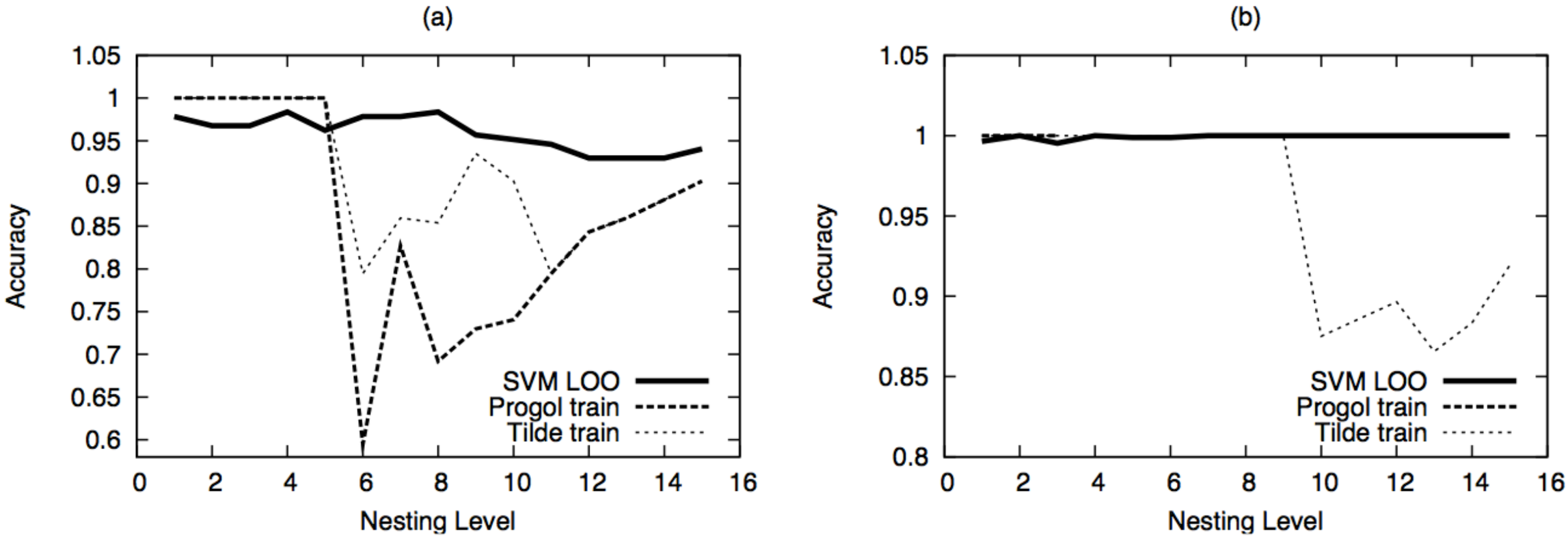
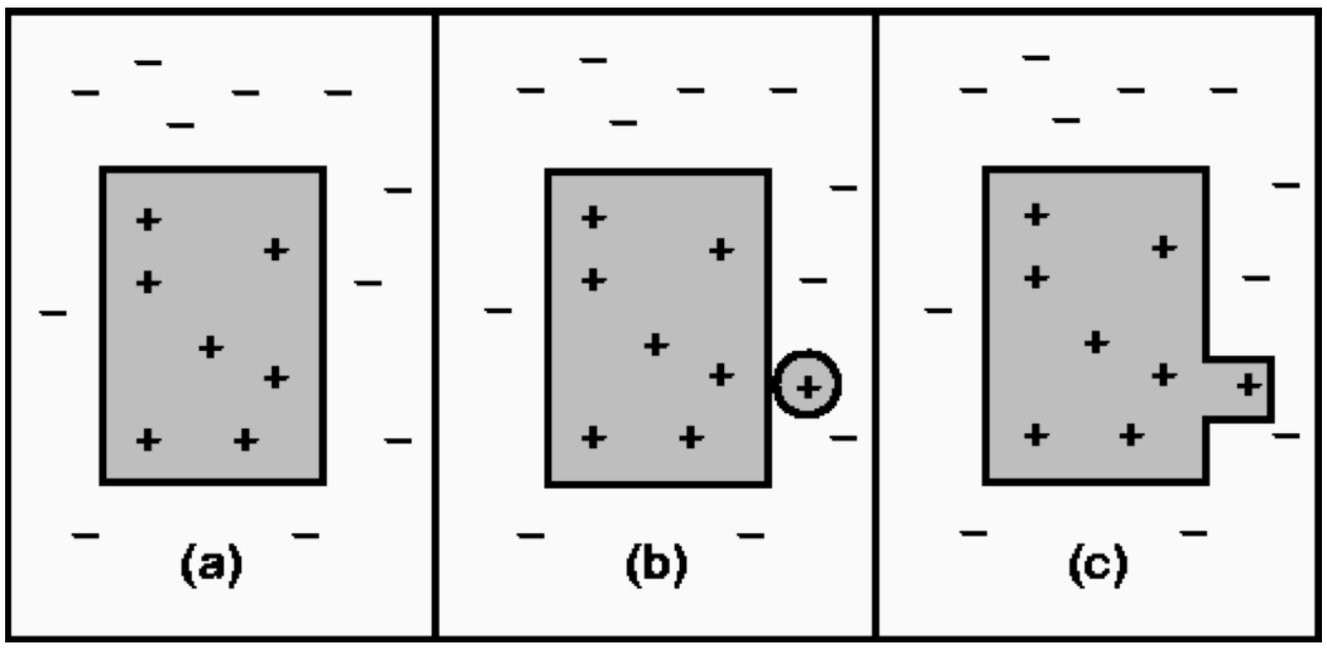


Fig. 7. Comparison between SVM leave-one-out error, Progol and Tilde empirical error in learning the *triangle-*X^n*-triangle* for different values of n, for data sets corresponding to $m = 10$ (a) and $m = 50$ (b).

FOIL

TECHNISCHE UNIVERSITÄT DARMSTADT

- Separate and Conquer rule learner
  - Generalize
    - add a rule that covers at least one positive but no negative samples
  - Separate
    - remove all samples from the training data that are covered by this rule
  - Conquer
    - if there are positive samples remaining repeat until there are no uncovered positive samples left



http://www.cl.uni-heidelberg.de/courses/archiv/ws03/einfki/KI-2004-01-13.pdf

kFOIL

TECHNISCHE UNIVERSITÄT DARMSTADT

kFOIL($\mathcal{D}, \mathcal{B}, \epsilon$)

1 $\mathcal{H} := \emptyset$
2 **repeat**
3 $c :=$ "$pos(x) \leftarrow$"
4 **repeat**
5 $c := \arg\max_{c' \in \rho(c)}$ Score$(\mathcal{D}, \mathcal{H} \cup \{c'\}, \mathcal{B})$
6 **until** stopping criterion
7 $\mathcal{H} := \mathcal{H} \cup \{c\}$
8 **until** score improvement is smaller than ϵ
9 **return** $\mathcal{H}$

- max. score is computed using a SVM

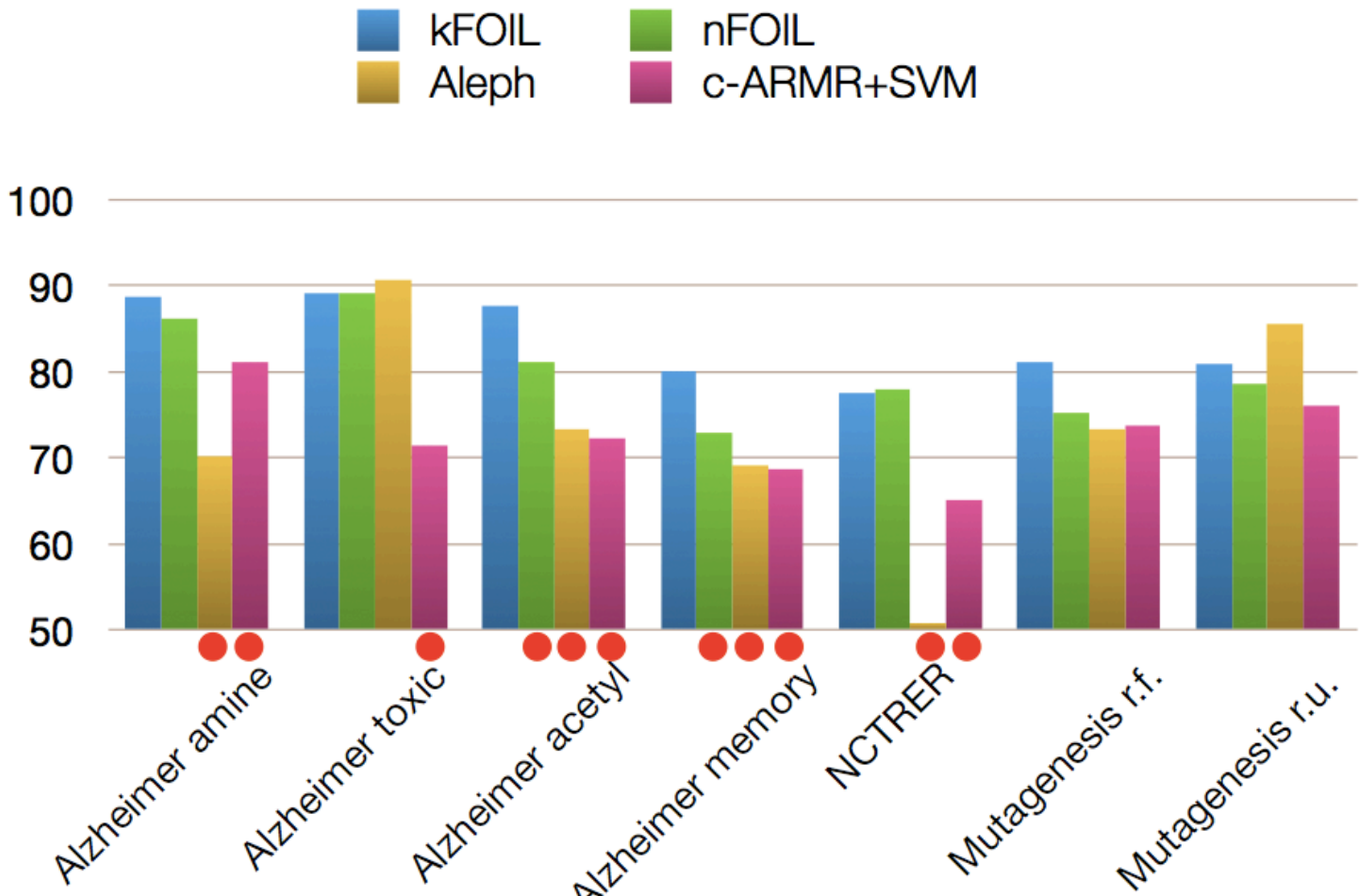
$$\text{Score}(\mathcal{D}, \mathcal{H}, \mathcal{B}) = \sum_{(x_i, y_i) \in \mathcal{D}} V(y_i, f(x_i))$$

- covered samples are not removed as the SVM would have to be retrained

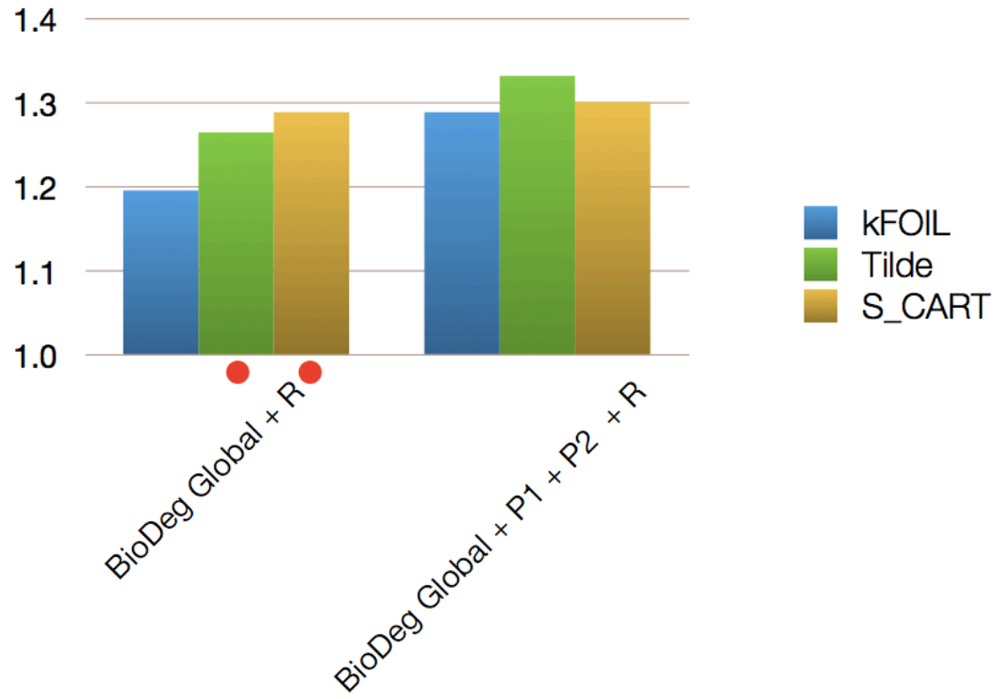
kFOIL Performance

TECHNISCHE UNIVERSITÄT DARMSTADT

Binary Classification



Regression



● kFOIL significantly better (95% confidence)

Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

- using kernels in the ILP setting has been shown to perform well, especially when the dimension of the feature space is unknown in advance
  - f.e. the Bongard problem which can only be solved with ILP if a predicate that counts the number of nested polygons is added to the background knowledge.

    Kernels on program traces can effectively discover concept without the additional hint.

- efficiently computable, but lacking interpretability(except for kFOIL)

- kFOIL is the less developed and tested method, but very promising and deserves more investigation

- the recent growth in knowledge representation and ontologies suggests that logic-based representations may be much more widespread in the future → interest increases in methods that can use this knowledge without programmer intervention

Sources

TECHNISCHE UNIVERSITÄT DARMSTADT

- http://www.disi.unitn.it/~passerini/papers/aprilchapter.pdf
- http://www.cs.berkeley.edu/~jordan/courses/281B-spring04/lectures/lec3.pdf
- http://omega.albany.edu:8008/machine-learning-dir/notes-dir/ker1/ker1.pdf
- http://www.lix.polytechnique.fr/~catuscia/teaching/cg520/98Fall/lecture_notes_98/LN_Dale/eval.ps