Complexity-Preserving Functions
Symmetry

Jan Lemeire

- Experience in programming and machine learning
  - construction of pragmatic algorithms
  - need for theory
- share my insights, but without formal justification
- different than statistical viewpoint

Symmetries

- Kolmogorov complexity K(x)
- Two-part code:

  "*separate regularities from random part*"

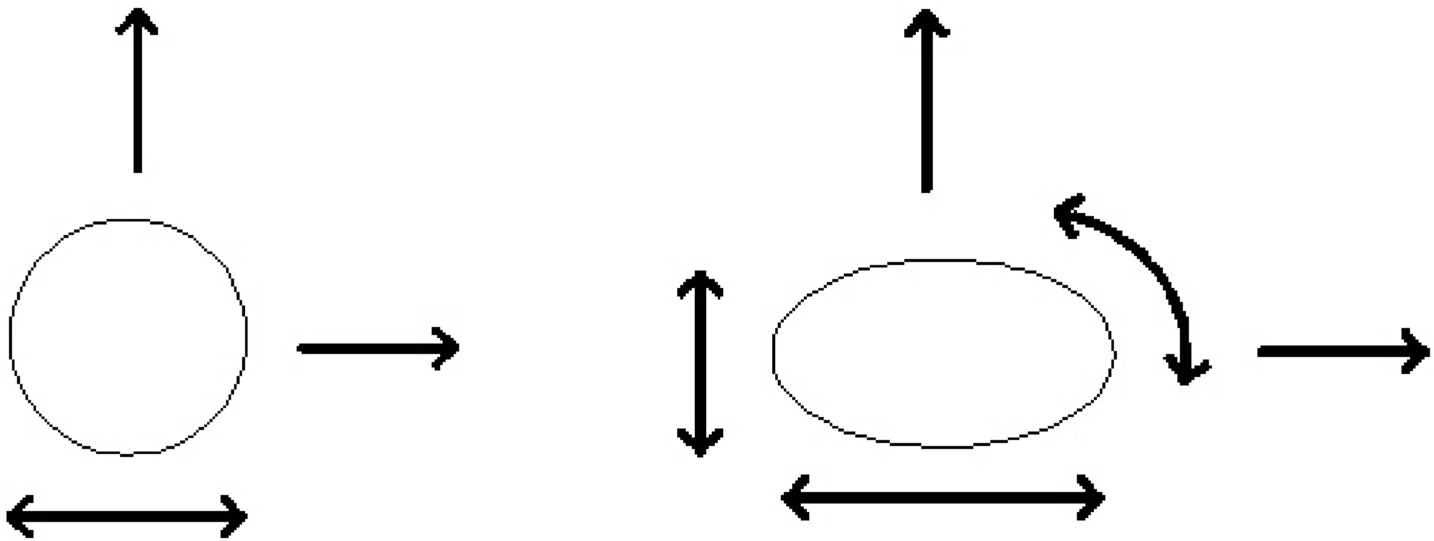
- Symmetry of an object:

  "*transformation that leaves essential features unchanged*"

  – symmetries: group theory

  – all symmetries of object: automorphism group

Transformations of x that preserve K



Circle *Ellipse*

- Degrees of freedom of object
- But: there are ‘reductions’ ⇒ smaller K
  - ellipse ⇒ circle

Two-part code

Shortest program for circle:

$$\alpha \in [\ 0\ ,2\Pi\ [\ : \text{plot}(x_0\text{+r.cons}\alpha,\ y_0\text{+r.sin}\alpha)$$

- Are parameters explicit in program?
- Separation of program and parameters
- Other interpretation of two-part code (model & error)
  - parameters ≠ random part?

Typical set of x

- All K-preserving transformations of x
  - form an automorphism group
  - $\Rightarrow$ set C_x of objects with same average K
  - “orbits of x under the automorphism group”
- *minimal sufficient statistic*
  - $p(d)=x$
  - for all d: p generates S, the ‘most likely’ set of which x is a typical element

Concept Learning (1)

- useful in unsupervised learning:

  one example $\Rightarrow$ set

  – of similar objects

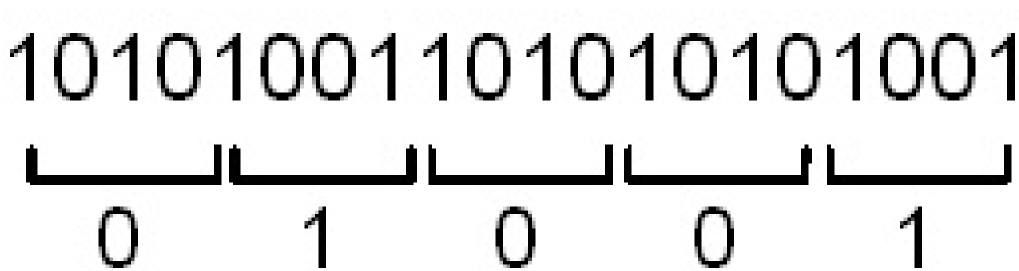
  – with minimal model complexity

  – example is typical element

- "Compress while learning, to learn by compression" (again)

Concept Learning (2)

Learn concept of binary strings with boolean formulae



1010=0

1001=1

⇒ easier to learn (VC dimension is reduced)

Concept Learning (3)

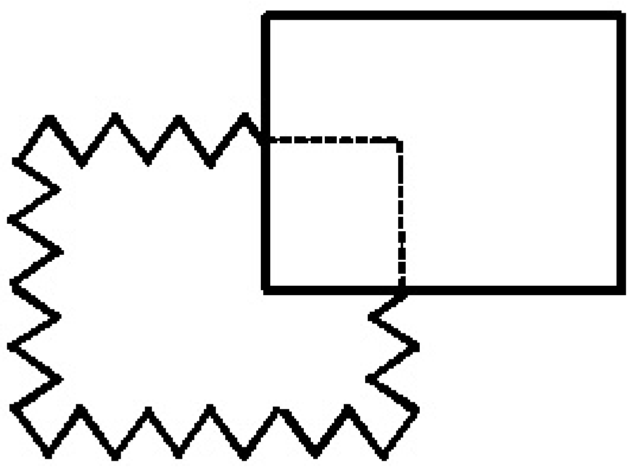
- Concept with symmetry g: $\forall\, x \in c \Rightarrow g(x) \in c$
- However, if symmetry g is not ‘present’ in concept class C, ‘induction following g cannot take place’
- eg.: x is a binary string, g = inverse, learn with kDNF formulae
  - split c in 2 subsets e and e^{-1}
  - learn e with f_1
  - for learning e^{-1}, an independent f_2 should be learned
  - c: f_1 or f_2
    - $f_2 = f_1^{-1}$
  - however, adding g would increase learnability

    c: f_1 or $g(f_1)$

Conclusions

- Symmetries look promising…
  - problems do exhibit symmetries
- Choice of model class is important!
  - the model itself reflects the object
  - the model determines the induction capacities

- Each symmetry (regularity) breaking increases the complexity
- don’t misinterpret Occam’s Razor and simplicity



Concept Learning (4)

- $x \in c \Rightarrow g(x) \in c$
- x & g(x) not close, according to their euclidean distance
- eg: 2 circles in $n \times n$ euclidean space
- => one cannot be learned from the other in distance-based algorithms (nearest-neighbour, case-based reasoning)