Machine Learning with Templates

Dr. Michael Fiske
Aemea Institute
mike@aemea.org

ICAI July 19, 2012
Las Vegas

Machine Learning Motivation

Some tasks are better suited for bottom-up methods.

- Pattern recognition

  Chris Bishop. *Pattern Recognition & Machine Learning* (2006).

  Hertz, Krogh & Palmer.

  Introduction to the Theory of Neural Computation (1991).

- Perception

  Gerald Edelman. *Neural Darwinism* (1987).

- Vision

  David Marr. *Vision* (1982).

Machine Learning Motivation

Some tasks are better suited for top-down methods.

- IBM's Deep Blue playing chess

  Hsu, Anantharaman, Campbell & Nowatzyk.
  A Grandmaster Chess Machine (1990).

- Chef program creates new cooking recipes

  Hammond. *Inside Case-Based Reasoning* ed. Schank
  & Riesebeck. (1989).

Useful Properties 1, 2

1. Categorization is probabilistic.

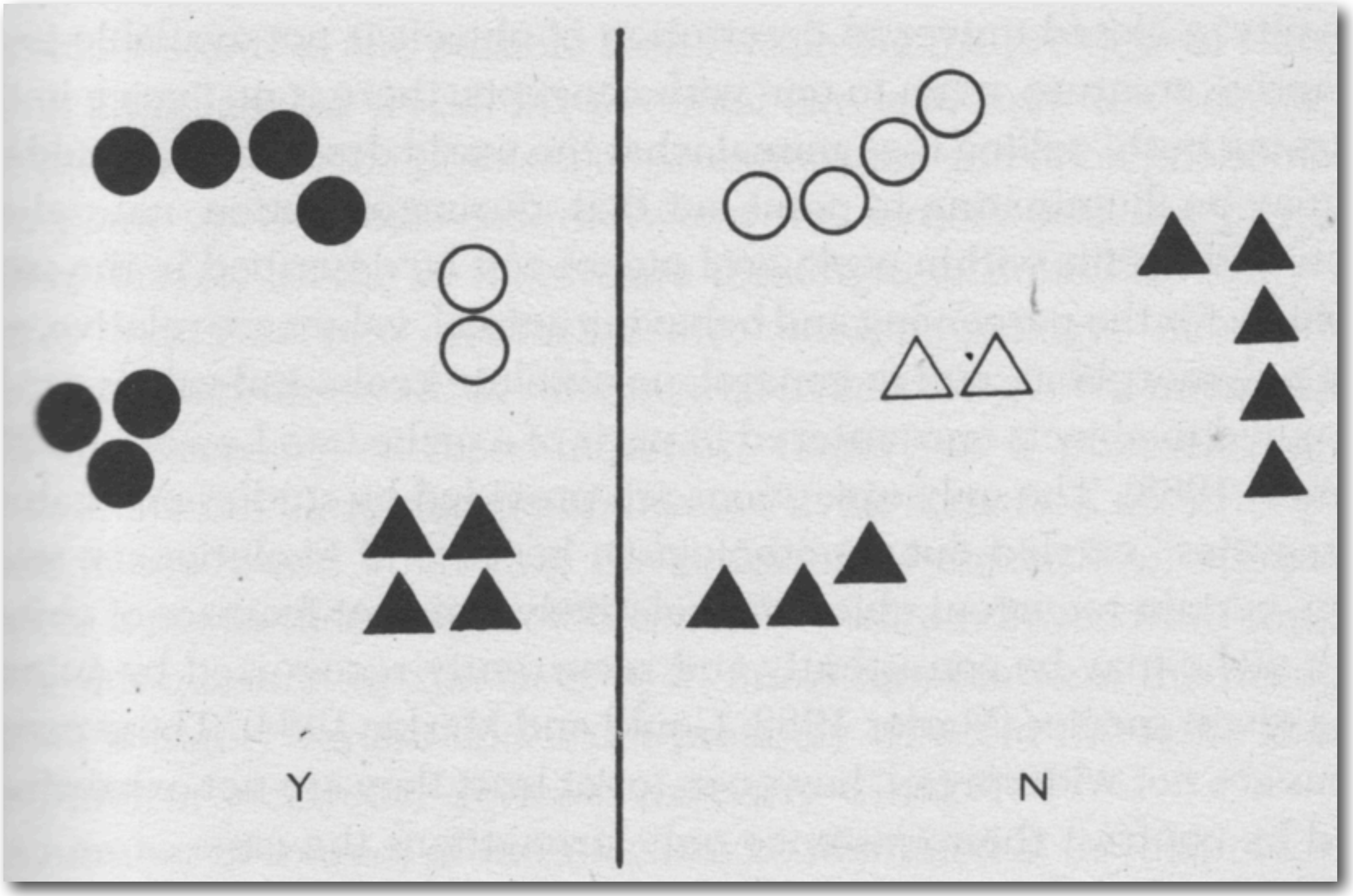
   For each category c, recognition algorithm outputs $\mu(e, c)$ in [0, 1], which measures how much example e lies in category c.

2. Categorization is polymorphous.

   During recognition, a template only eliminates a category if has not “seen” a nearby value during learning.

Polymorphous Rules

The polymorphous rule for category **Y** is that each element has at least two of the properties: **roundness**, **solid color**, or **bilateral symmetry**.



Useful Properties 3, 4

3. Learning Algorithm is fast — no local minima.

   No greedy optimization algorithm such as gradient descent.

4. Recognition Algorithm is extremely fast.

   Uses exponential elimination.

   Example: 180 million Chinese characters. If templates on average eliminate a third (1/3) of the remaining Chinese character categories, it takes at most 46 steps to finish one iteration of the outer loop.

Useful Properties 5, 6

5. Templates can be designed by people, evolution algorithm 6.1, other machine learning methods, or a combination.

6. Applicable across many domains.
   - ✓ Bioinformatics
   - ✓ Financial Forecasting
   - ✓ Goal-based Planning
   - ✓ Information Retrieval
   - ✓ Machine Vision
   - ✓ Natural Language Processing / Understanding
   - ✓ Pattern Recognition (e.g., writing, speech, voice)

Template Recognition Algorithm

C is the category space. e is the example to categorize.

```
Read learned templates $\{T_1, T_2, . . . T_n\}$ from memory.
Initialize each category score $s_c$ to zero.
Outer loop: m trials
{
   Initialize set $R := C$.
   Inner loop: choose $\rho$ templates randomly.
   {
      Choose template $T_k$ with probability $p_k$
      For each category $c \in R$ if $M(T_k(e), T_k(c))$ is close to 0
      then set $R := R - \{c\}$. (Remove category c from R)
   }
   For each category c remaining in R, $s_c := s_c + 1$.
}
Example e lies in category c with measure $\mu(e,c) = s_c/m$.
```

Sharp categorization boundary: Example e lies in categories c satisfying $s_c/m > \theta$ where θ is the category threshold

Template Definitions

$\boldsymbol{C}$ is the category space.

$\boldsymbol{V}$ is the template value space. $\boldsymbol{P}(\boldsymbol{V})$ is the power set of $\boldsymbol{V}$.

$\boldsymbol{E}$ is the example space.

$\boldsymbol{S}$ is the similarity space.

Template value function $\boldsymbol{T_k}: \boldsymbol{E} \to \boldsymbol{V}$

Template prototype function $\mathfrak{T}_k: \boldsymbol{C} \to \boldsymbol{P(V)}$

Matching function $\boldsymbol{M}: \boldsymbol{V} \times \boldsymbol{P}(\boldsymbol{V}) \to \boldsymbol{S}$

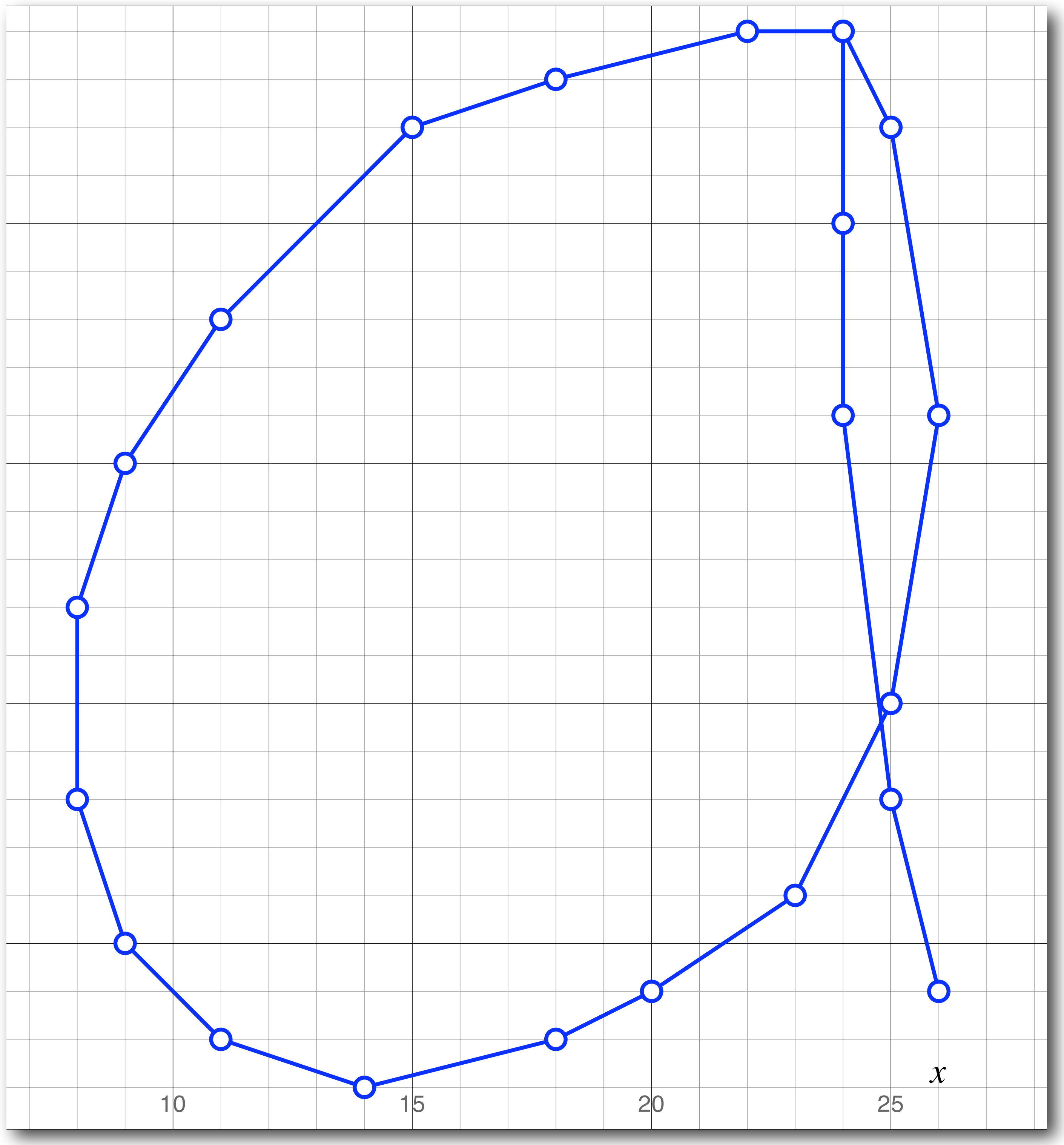
$\boldsymbol{M}(\boldsymbol{T_k}(\boldsymbol{e}), \mathfrak{T}_k(\boldsymbol{c}))$ is a similarity value that measures how close template value $\boldsymbol{T_k}(\boldsymbol{e})$ is to the set of prototypical values $\mathfrak{T}_k(\boldsymbol{c})$.

Handwriting Example

$C = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$

$V = \{0, 1, 2, 3, \ldots\}$

Example ***e*** of letter ***a*** ➔



Intersection Template T_0

Template T_0 computes the number of intersection points with a horizontal line at 3/4 of the letter height measured from the top.

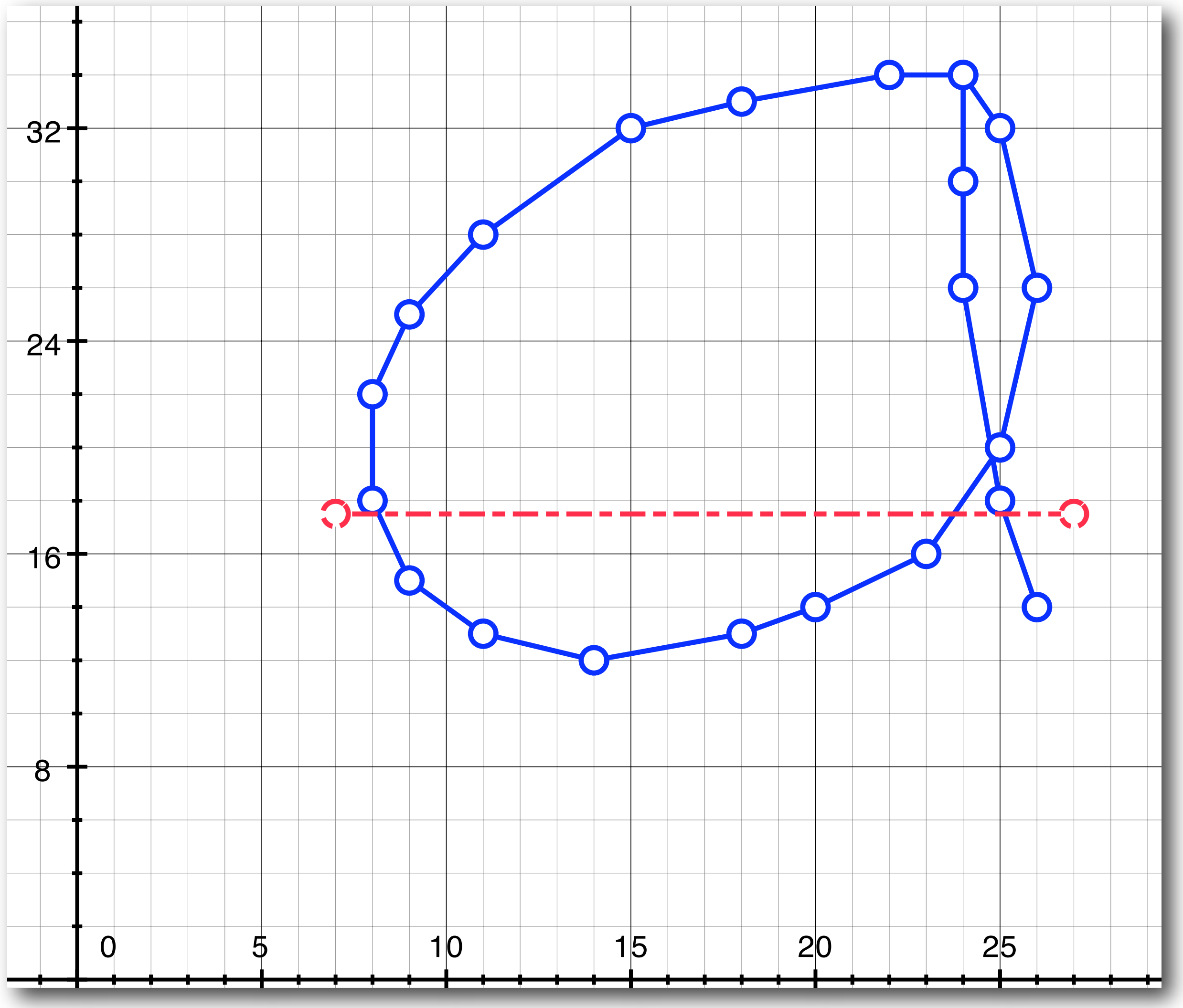
3 intersection points:

(8.2, 17.5)

(23.8, 17.5)

(25.1, 17.5)

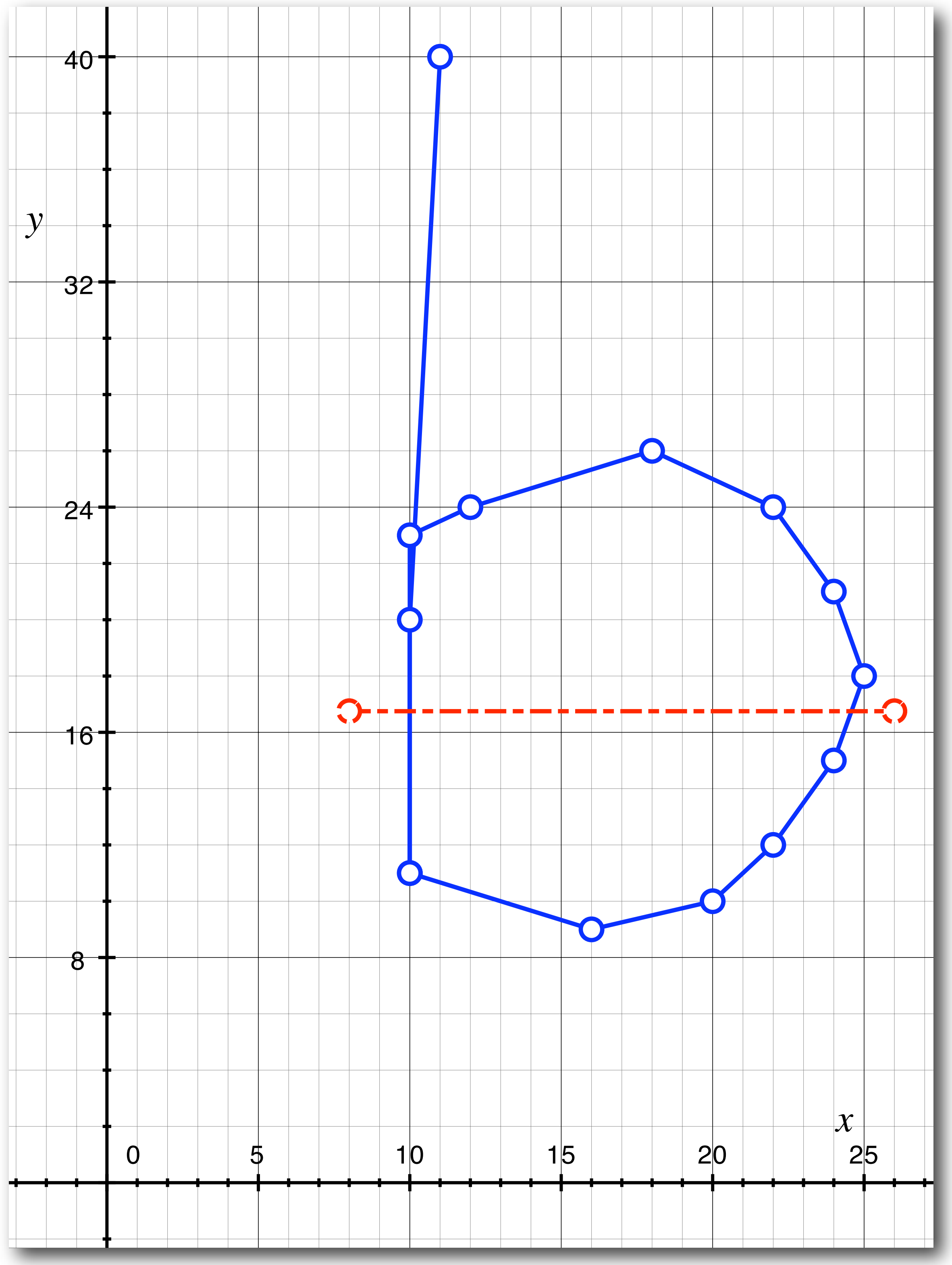
$$T_0(\boldsymbol{a}) = 3$$



Intersection with letter ***b***

Two intersection points: (10, 16.75)   (24.6, 16.75)

$T_0(\boldsymbol{b}) = 2$



Template Learning Algorithm

```
Read user designed or evolved templates {$T_1$, $T_2$, . . . $T_n$}.
Outer loop: iterate thru each template $T_k$
{
   Initialize X := $\mathfrak{T}_k(c_1)$.
   Initialize A := X.
   Inner loop: iterate thru each category $c_i$.
   {
      Set $E(c_i)$ := all learning examples in $c_i$.
      Build prototype function as follows:
      Set $\mathfrak{T}_k(c_i)$ := $\bigcup\{v\}$ for each $v = T_k(e)$ and $e \in E(c_i)$.
      A := A $\cup$ $\mathfrak{T}_k(c_i)$.
      Build part of matching function $M(\_\_, \mathfrak{T}_k(c_i))$.
   }
   If (A == X), then remove $T_k$ from the template set.
}
Probability $p_k$ := 1/m where there are m remaining templates.
Store remaining templates.
```

Building Templates with Evolution

Evolution Summary Part 1

For each category pair (c_i, c_j) , where $i < j$, the building blocks $\{f_1, f_2, \ldots, f_r\}$ are used to build a population of m templates.

Construct $\{l_1, \ldots, l_m\}$ where each bit sequence l_k encodes template $T_k^{(i,j)}$ built from $\{f_1, f_2, \ldots, f_r\}$.

Superscript (i, j) indicates that these templates are evolved to distinguish examples chosen from $E(c_i)$ and $E(c_j)$.

Building Templates with Evolution

Evolution Summary Part 2

Fitness of template $\boldsymbol{T_k^{(i,j)}}$ is measured by how well it distinguishes examples from $\boldsymbol{E(c_i)}$ and $\boldsymbol{E(c_j)}$, amount of memory used, and computational speed.

The population of bit-sequences is evolved using crossover and mutation until there are at least **Q** templates each with an acceptable fitness.

Current Status and Next Steps

1. Testing against the UCI Database.

2. What templates work best for what tasks?

3. Human designed versus evolved templates.

4. If commercial interest, please contact:
   Michael Fiske. michael@fiskesoftware.com