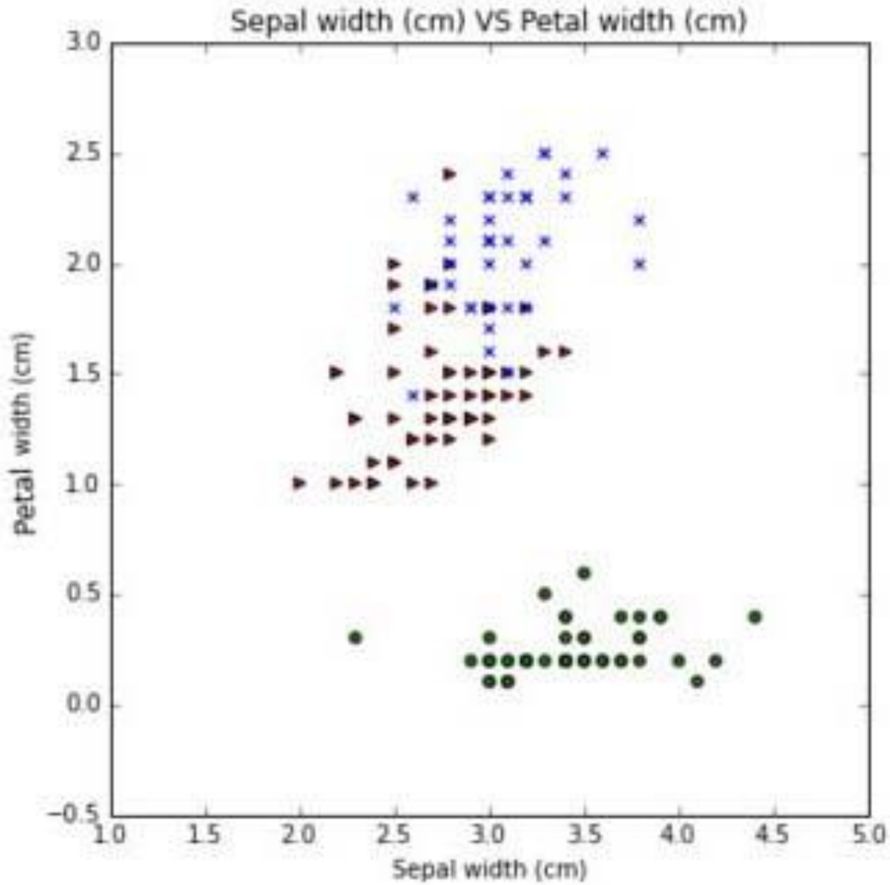
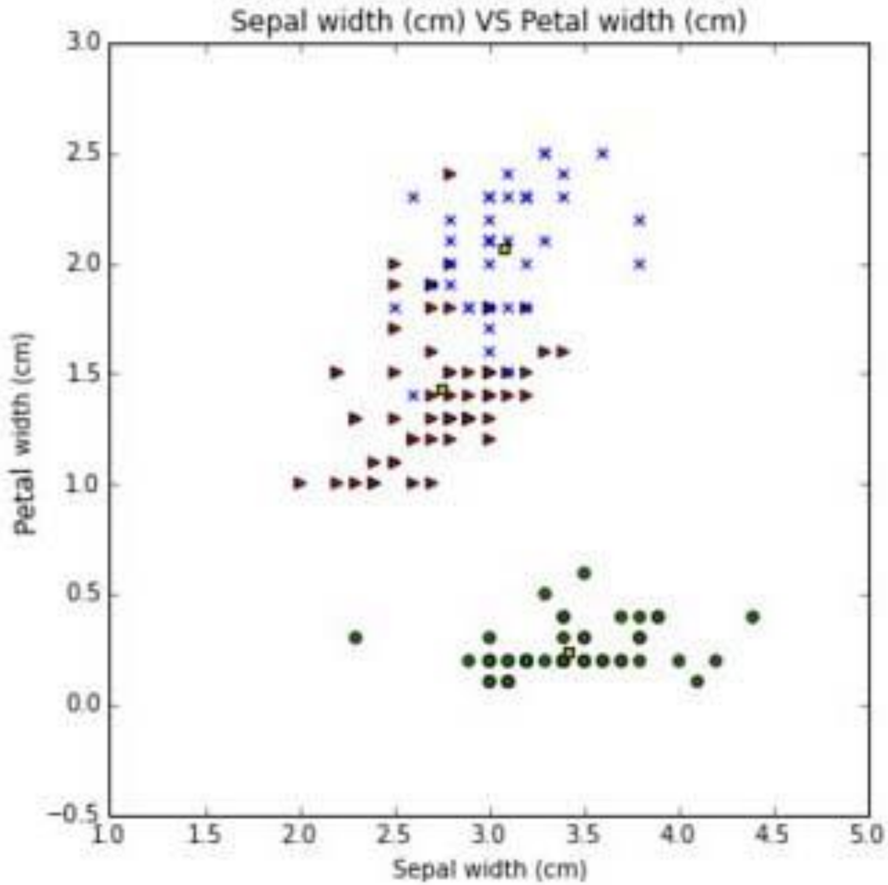
Classification Using Genetic Programming

Patrick Kellogg
General Assembly
Data Science Course (8/23/15 - 11/12/15)

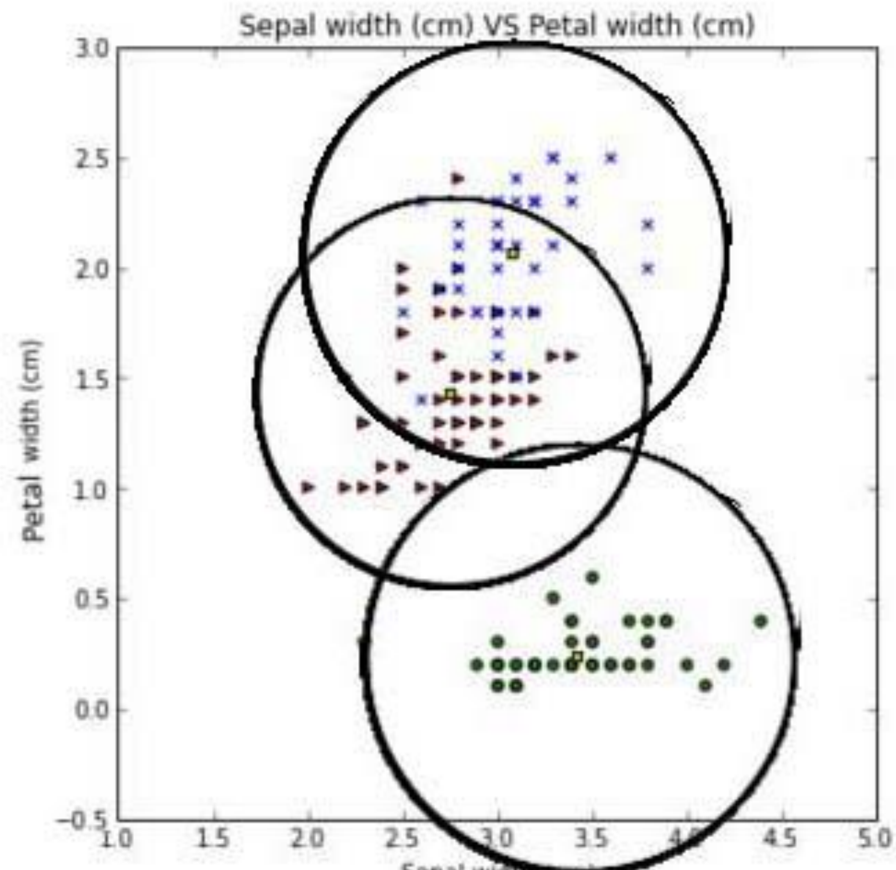
Iris Data Set



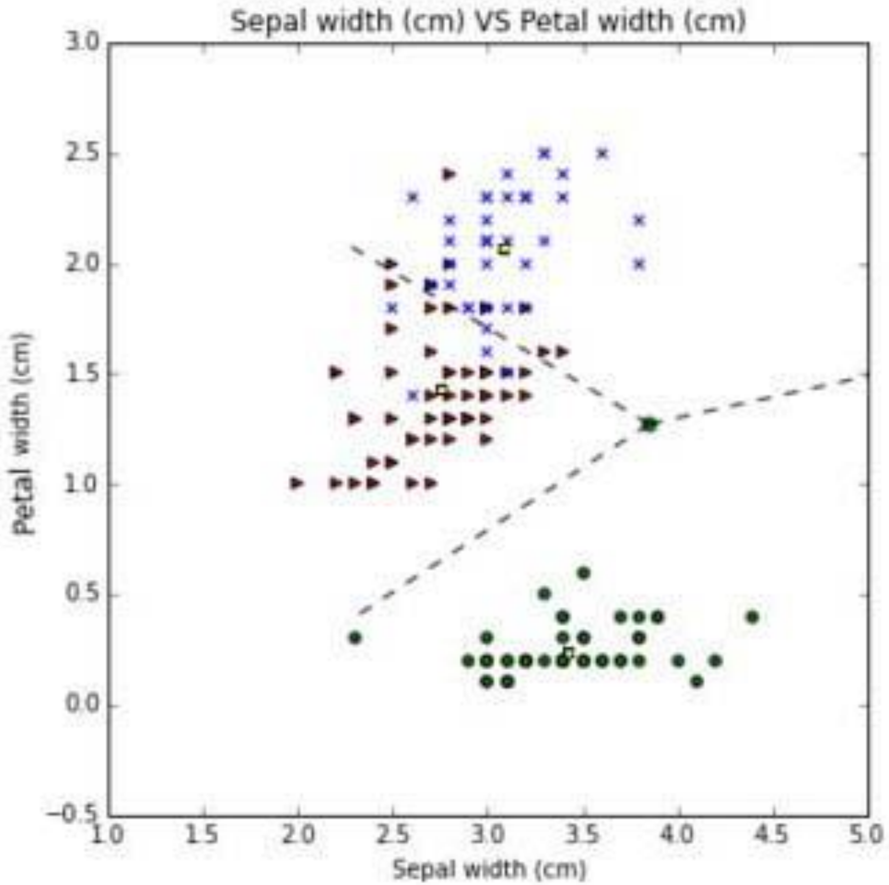
Iris Data Set



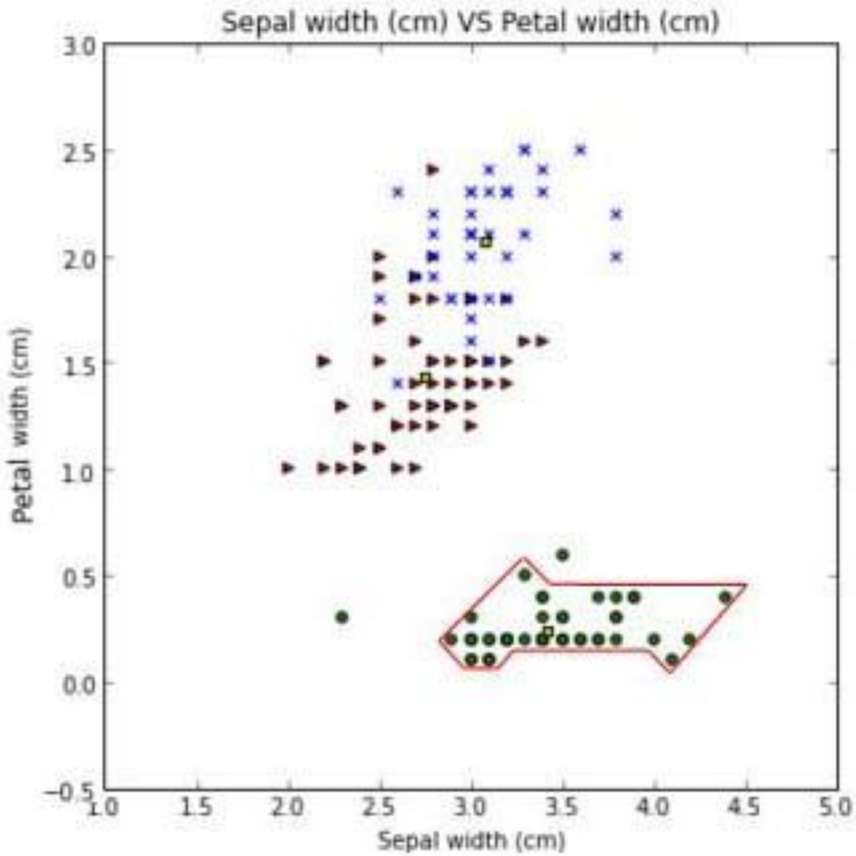
Iris Data Set



Iris Data Set



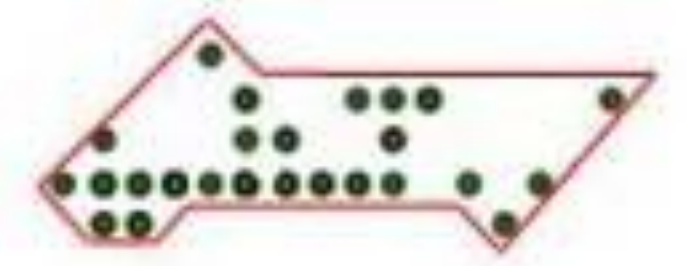
Iris Data Set



- Create a geometrical boundary for the class "Setosa"

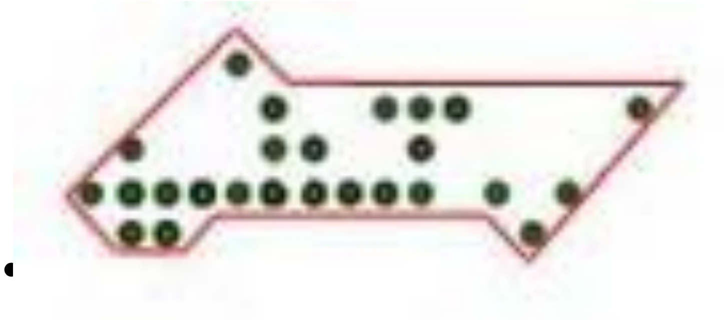
Automatically Creating Functions

```
def IsInClass(x,y):
  if ( (y > (2*x + 10)) \
    and (y > (0.3*x + 4.5)) \
    ...
    and (x < 5)):
    return true
  else:
    return false
```



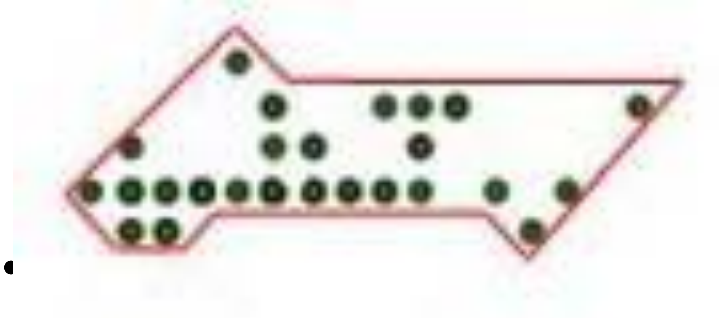
Evolving Parameters

(y > (2x + 10) and
(y > (0.3x + 4.5)) ...



Evolving Parameters

```
(y > (2x + 10) and
(y > (0.3x + 4.5)) ...
```

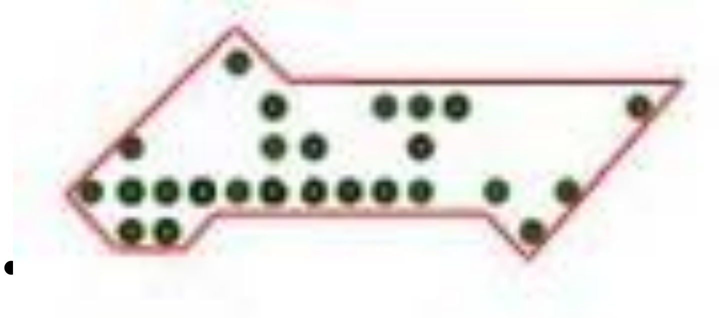


$y > \boldsymbol{\beta_1} x + \boldsymbol{\alpha_1}$

$y > \boldsymbol{\beta_2} x + \boldsymbol{\alpha_2}$ ...

Evolving Parameters

```
(y > (2x + 10) and
(y > (0.3x + 4.5)) ...
```



$y > \boldsymbol{\beta}_1 x + \boldsymbol{\alpha}_1$

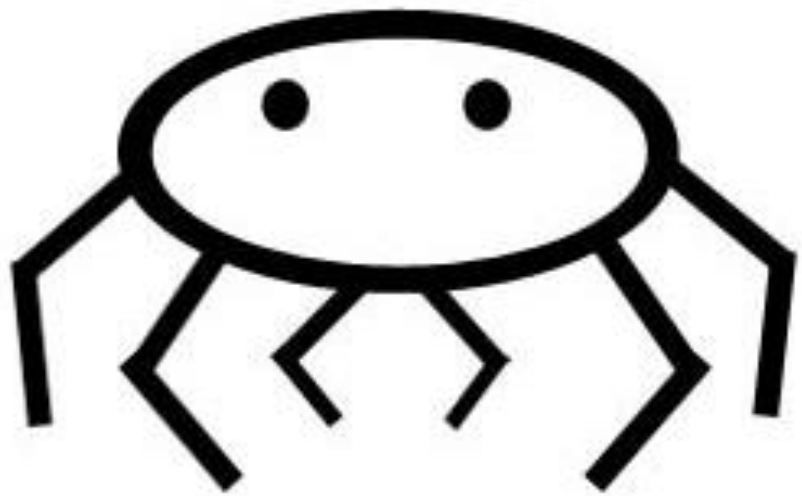
$y > \boldsymbol{\beta}_2 x + \boldsymbol{\alpha}_2$ ...

= Genetic Programming (GP)

Two-slide Introduction to Genetic Algorithms (Part 1)

Two-slide Introduction to Genetic Algorithms (Part 1)

Number legs = 6

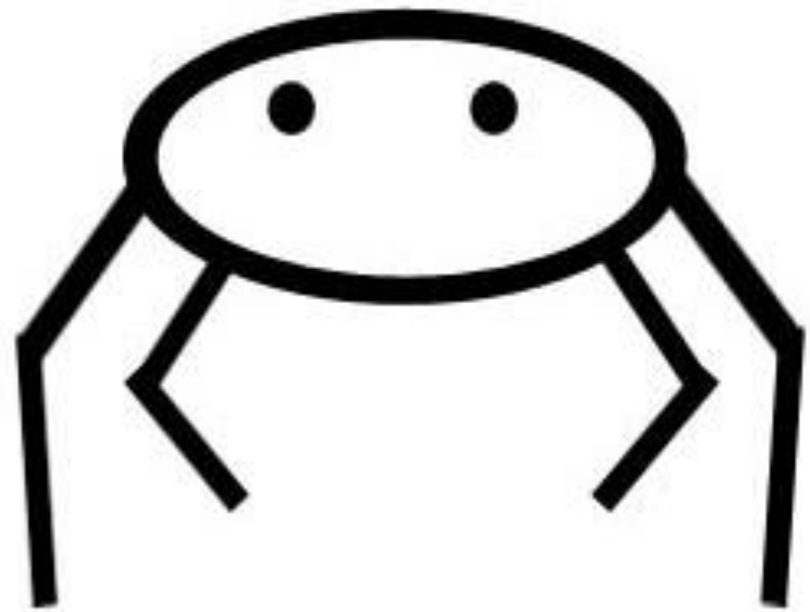


N6

Two-slide Introduction to Genetic Algorithms (Part 1)

Number legs = 4

Length legs = 8



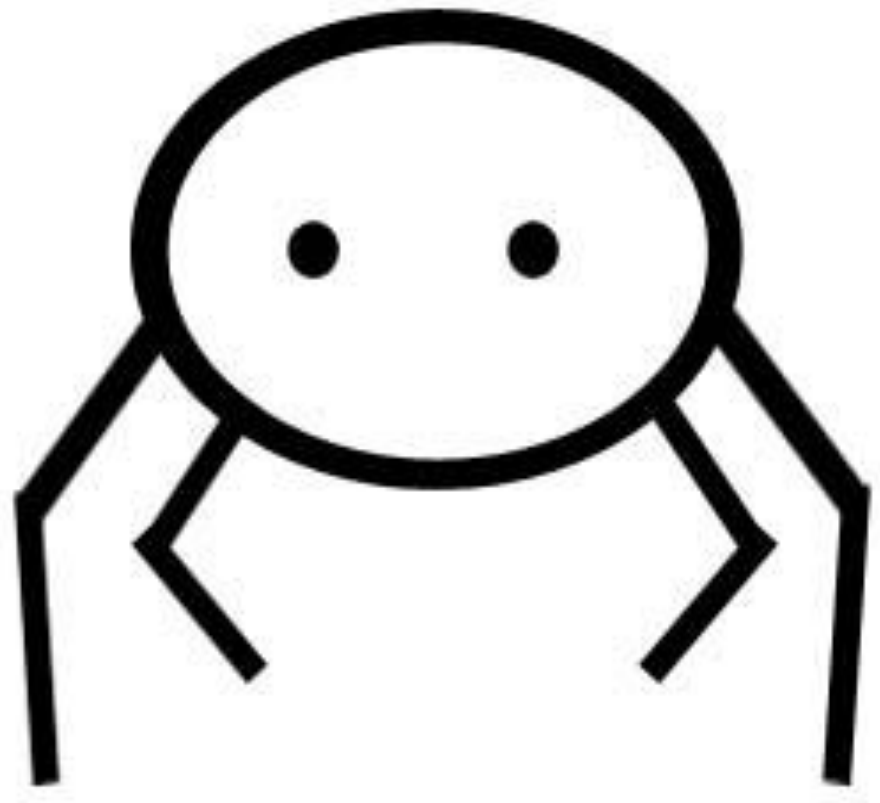
N4 **L8**

Two-slide Introduction to Genetic Algorithms (Part 1)

Number legs = 4

Length legs = 8

Size = 6



N4 L8 **S6**

Two-slide Introduction to Genetic Algorithms (Part 1)

Number legs = 0

Length legs = 8

Size = 3

Energy = 20

N0 L8 S3 **E20**

Two-slide Introduction to Genetic Algorithms (Part 2)

N6 L4 S3 E10

N4 L8 S3 E10

N4 L8 S6 E10

N0 L8 S3 E20

Initial Population

Two-slide Introduction to Genetic Algorithms (Part 2)

N6 L4 S3 E10

N4 L8 S3 E10

N4 L8 S6 E10

N0 L8 S3 E20

N6 L4 S3 E10 = 26

N4 L8 S3 E10 = 14

N4 L8 S6 E10 = 32

N0 L8 S3 E20 = 0

Fitness Function

Two-slide Introduction to Genetic Algorithms (Part 2)

N6 L4 S3 E10

N4 L8 S3 E10

N4 L8 S6 E10

N0 L8 S3 E20

N6 L4 S3 E10 = 26

N4 L8 S3 E10 = 14

N4 L8 S6 E10 = 32

N0 L8 S3 E20 = 0

Selection

N6 L4 S3 E10

N4 L8 S6 E10

Two-slide Introduction to Genetic Algorithms (Part 2)

N6 L4 S3 E10

N4 L8 S3 E10

N4 L8 S6 E10

N0 L8 S3 E20

N6 L4 S3 E10 = 26

N4 L8 S3 E10 = 14

N4 L8 S6 E10 = 32

N0 L8 S3 E20 = 0

N6 L4 S3 E10

N4 L8 S6 E10

N7 L4 S3 E10

Mutation

Two-slide Introduction to Genetic Algorithms (Part 2)

N6 L4 S3 E10

N4 L8 S3 E10

N4 L8 S6 E10

N0 L8 S3 E20

N6 L4 S3 E10 = 26

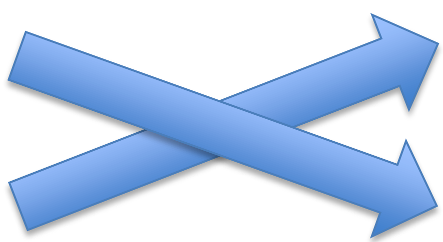
N4 L8 S3 E10 = 14

N4 L8 S6 E10 = 32

N0 L8 S3 E20 = 0

N6 L4 S3 E10

N4 L8 S6 E10



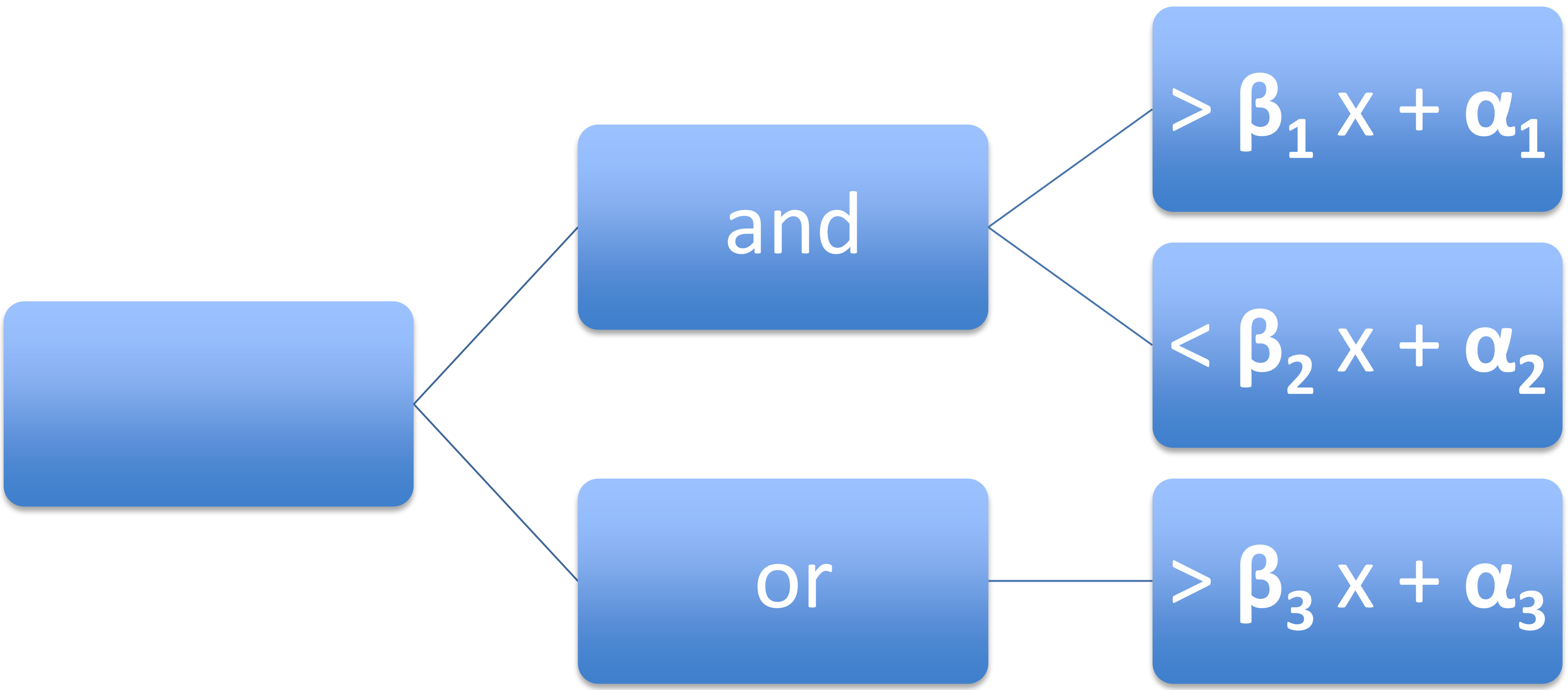
N7 L4 S3 E10

N6 L4 S6 E10

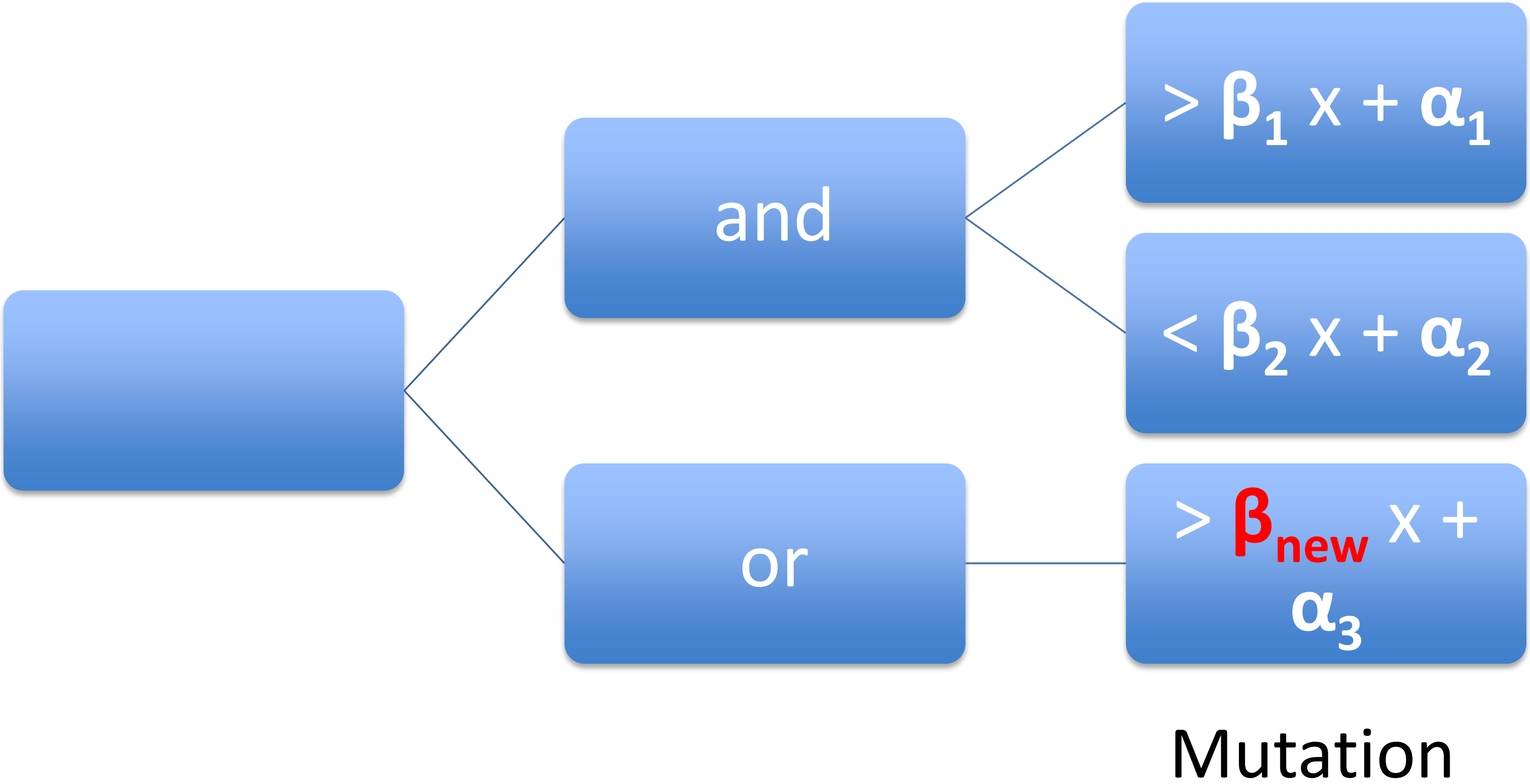
N4 L8 S3 E10

Crossover

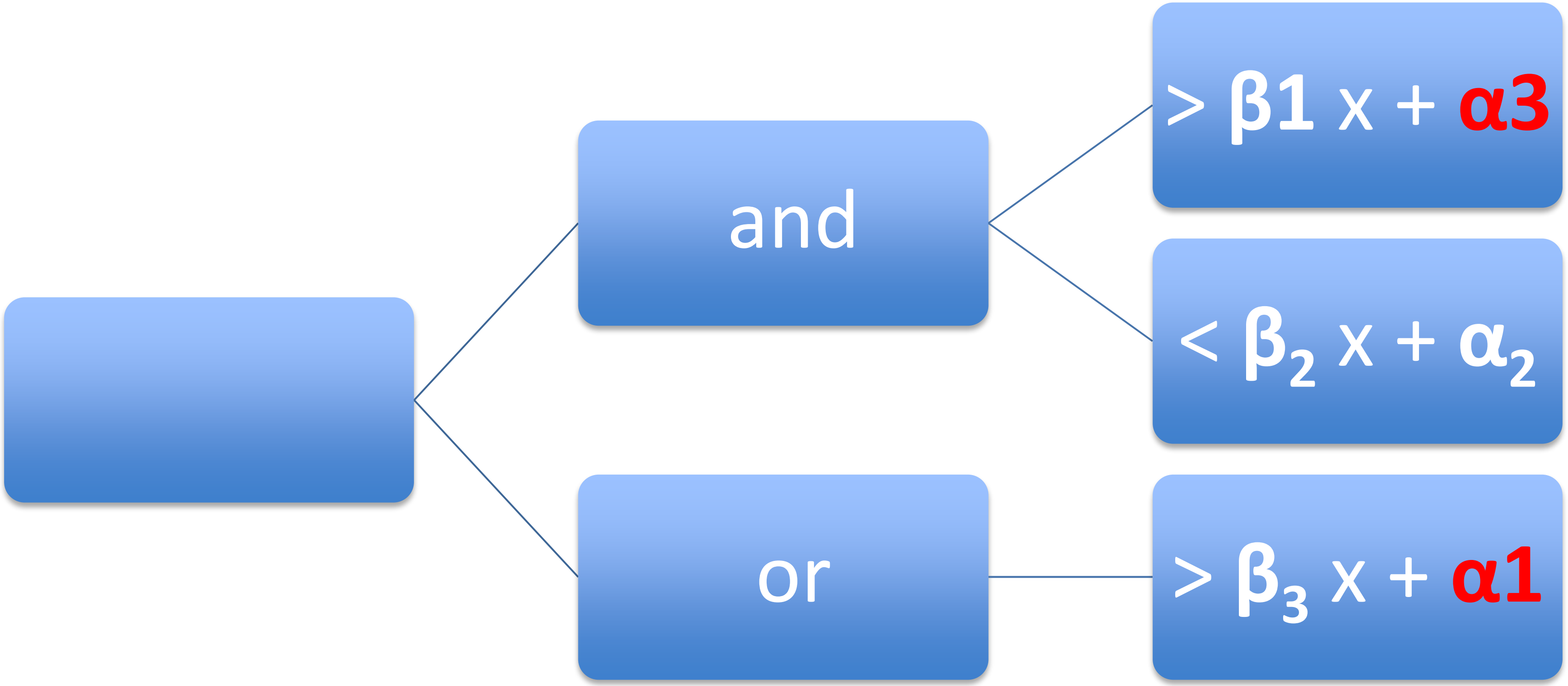
Syntax Tree-Based GP



Syntax Tree-Based GP



Syntax Tree-Based GP



Crossover

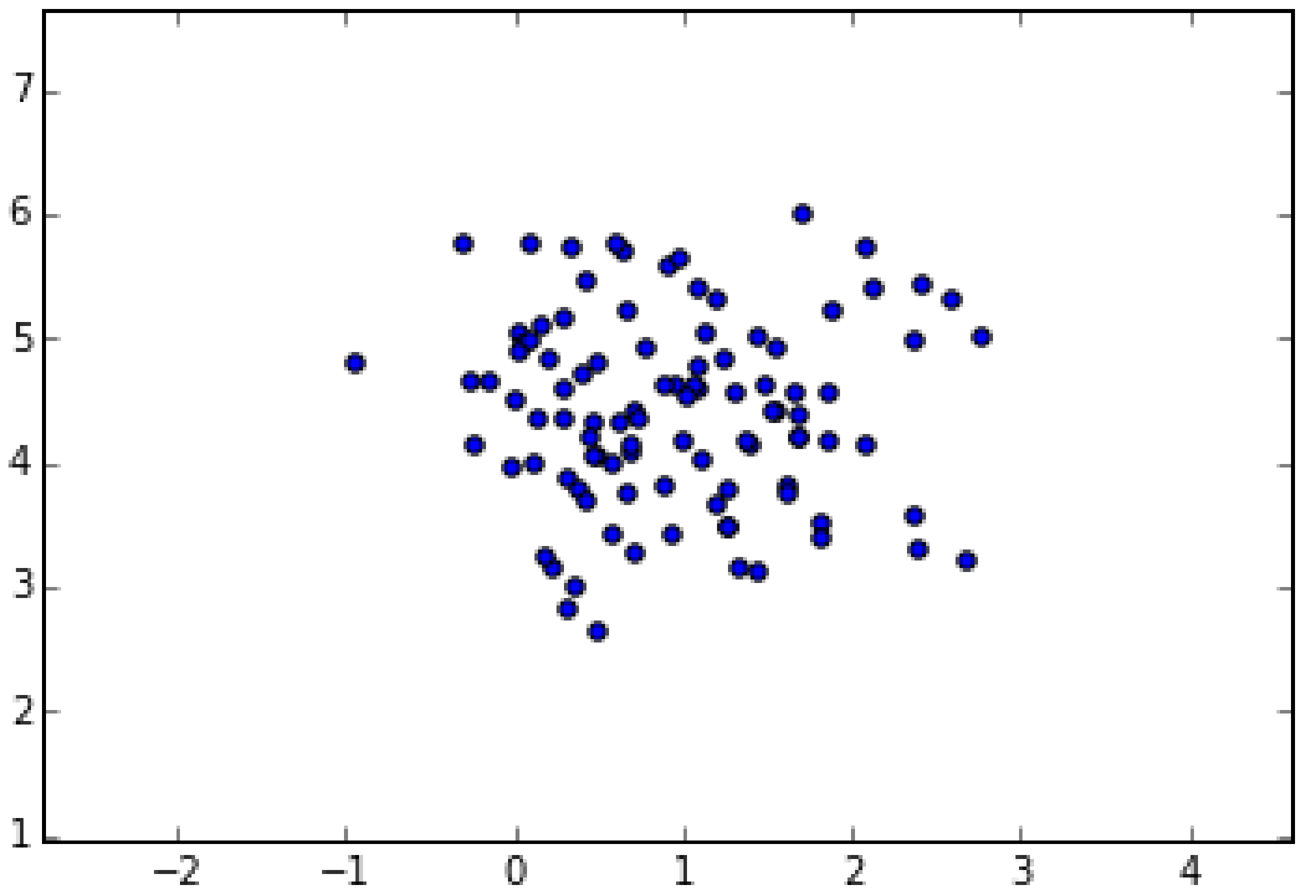
My Python Code

- Randomly create an initial population of 12 linear candidates

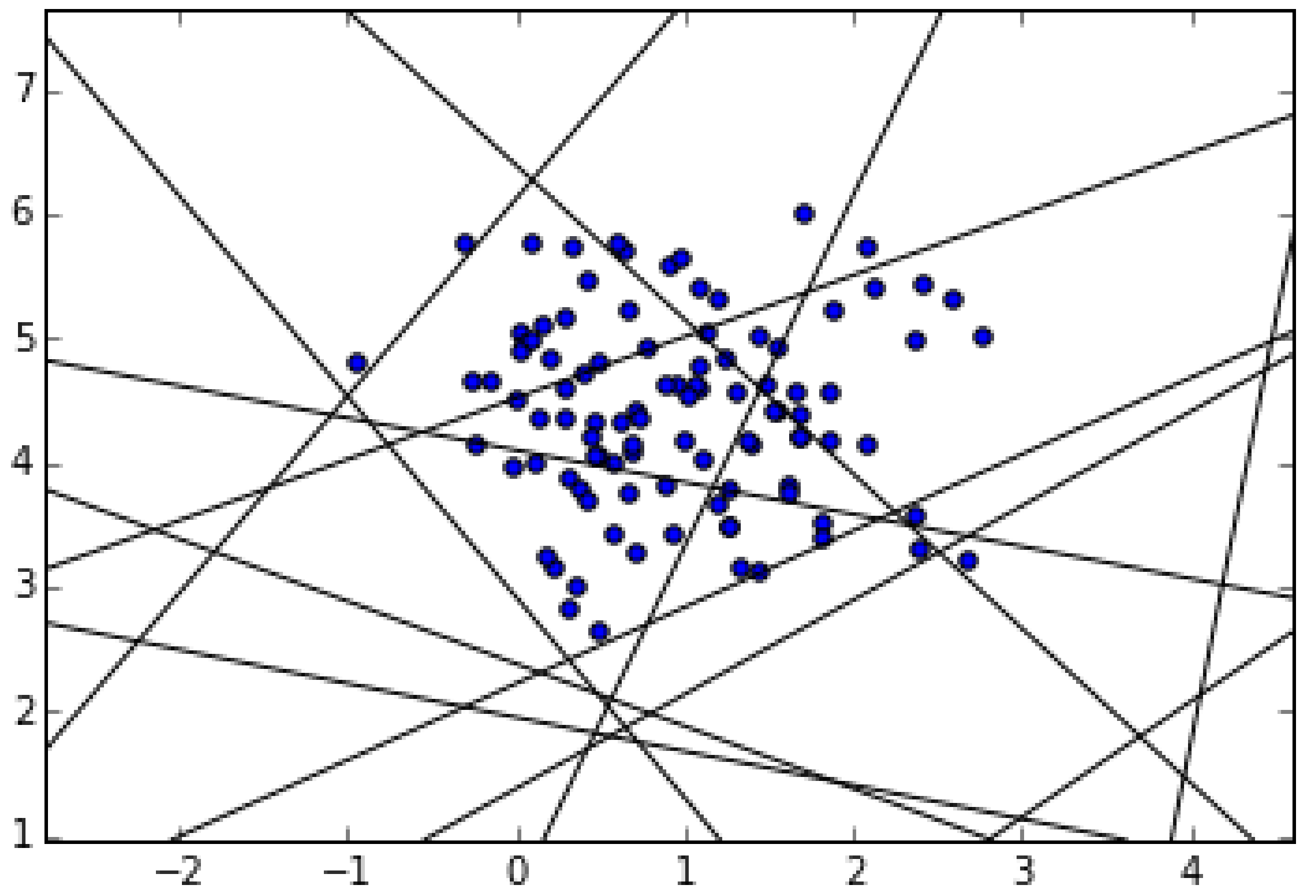
- Run fitness function on all 12
- Select top 2 candidates
- Mutate each four times (+α, -α, +β, -β)
- Crossover twice
- Repeat until error is small enough for next step

(which is to add or remove a terminal from the tree)

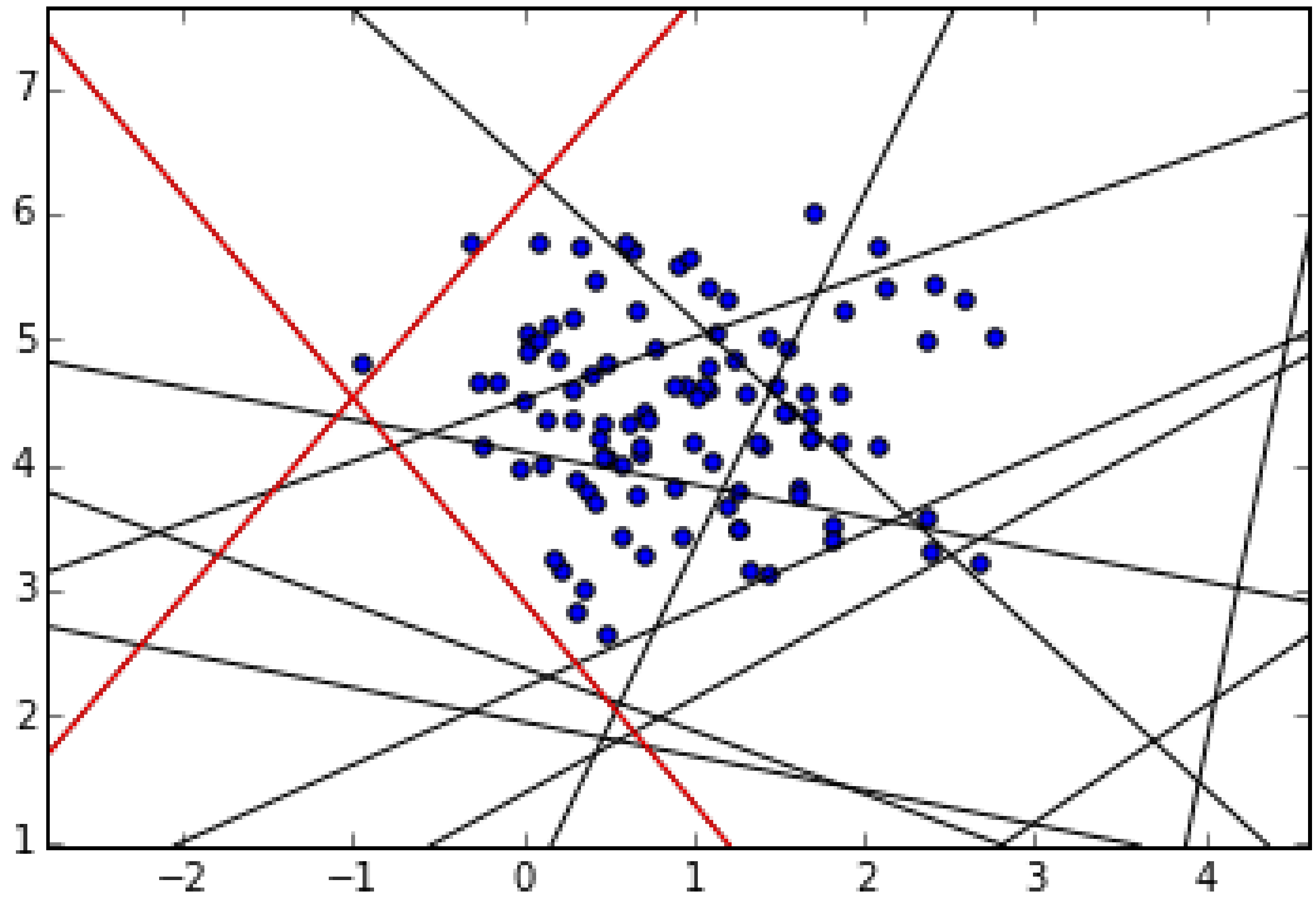
Sample Run of Hill-climbing



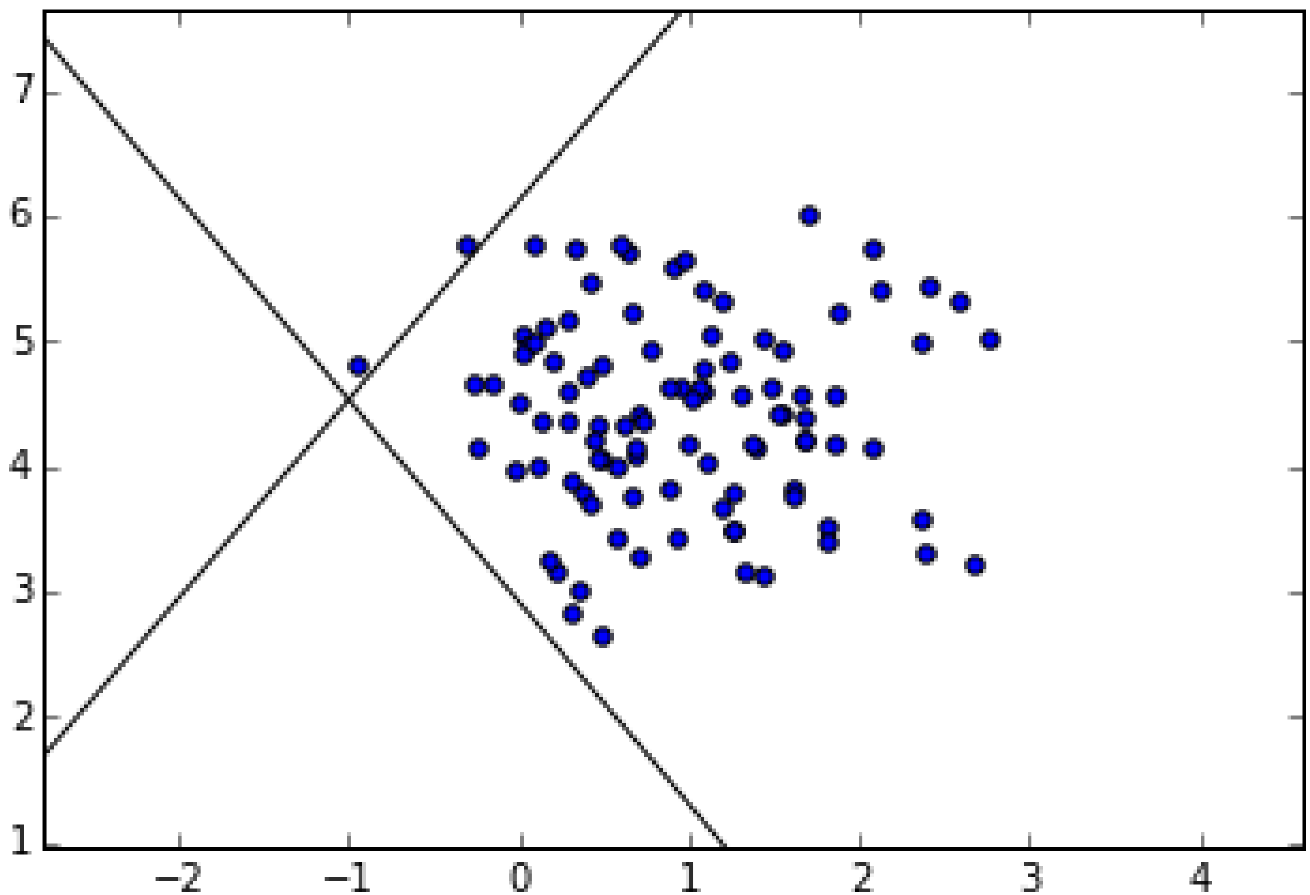
Sample Run of Hill-climbing



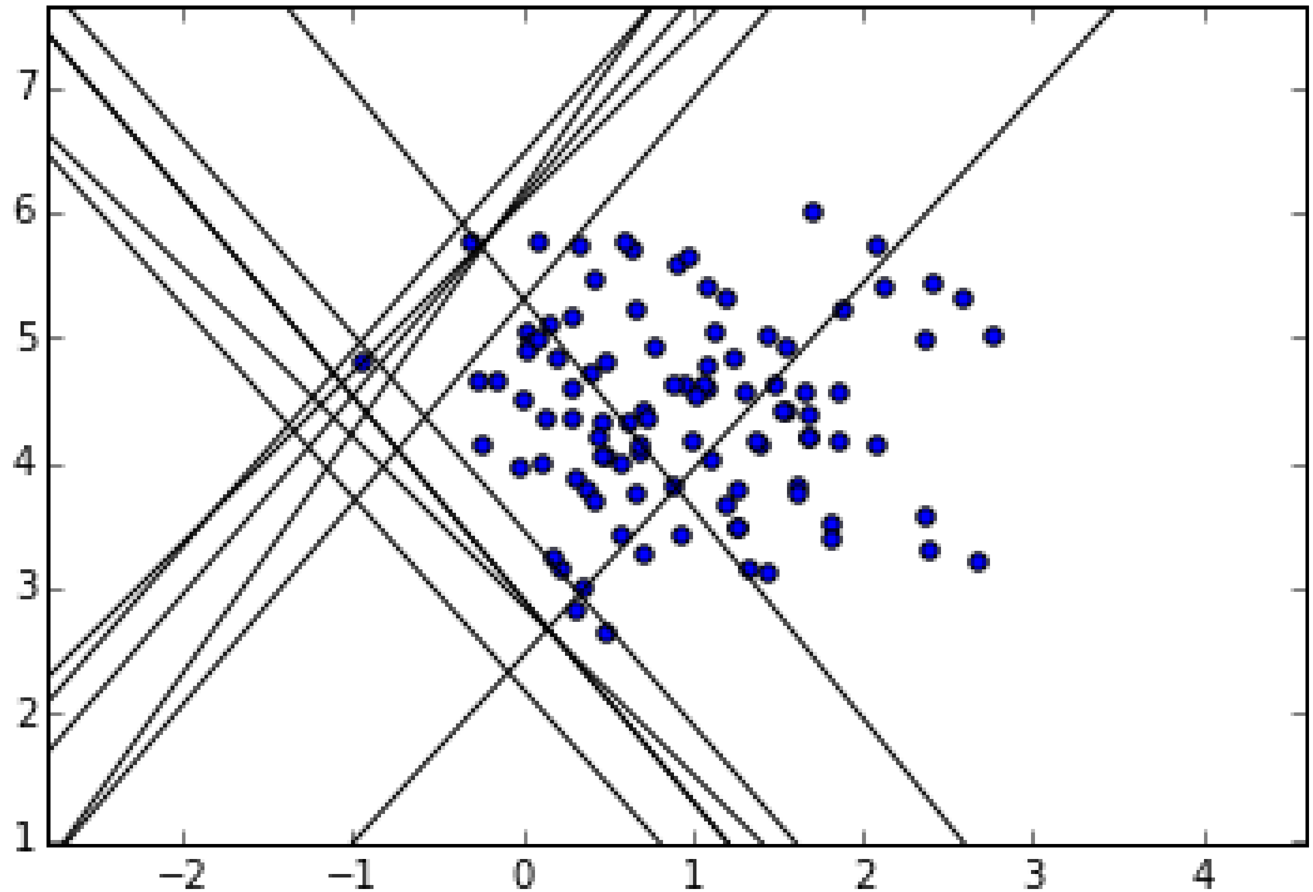
Sample Run of Hill-climbing



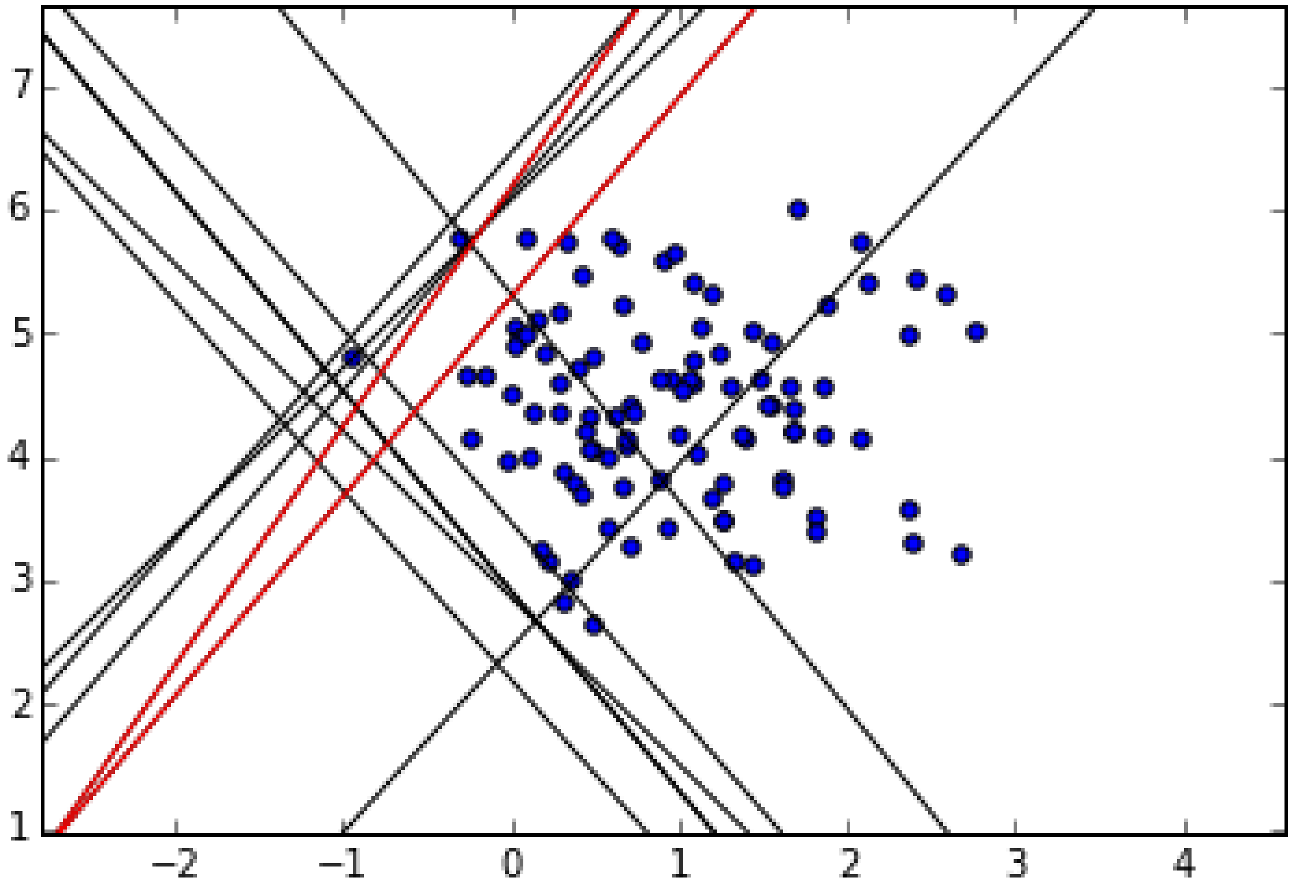
Sample Run of Hill-climbing



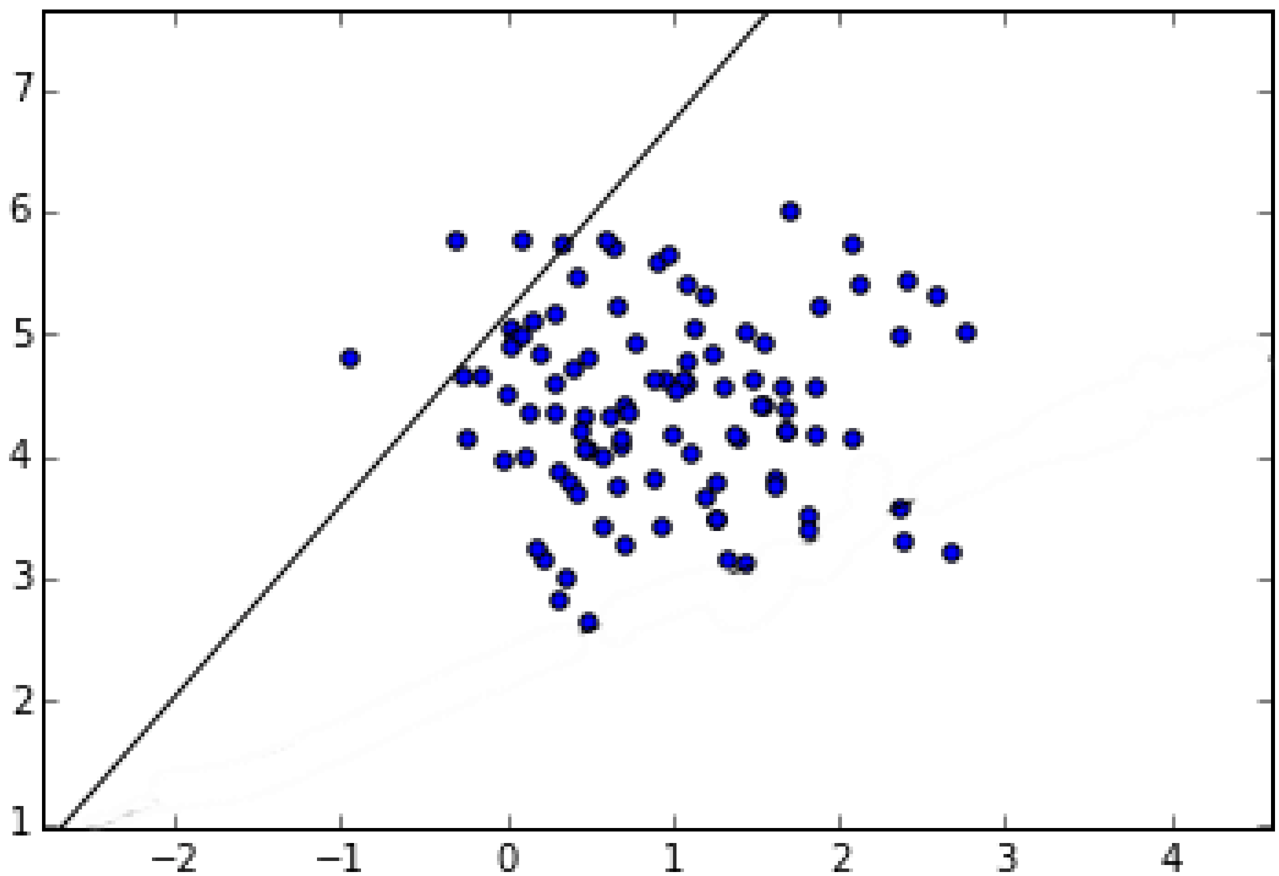
Sample Run of Hill-climbing



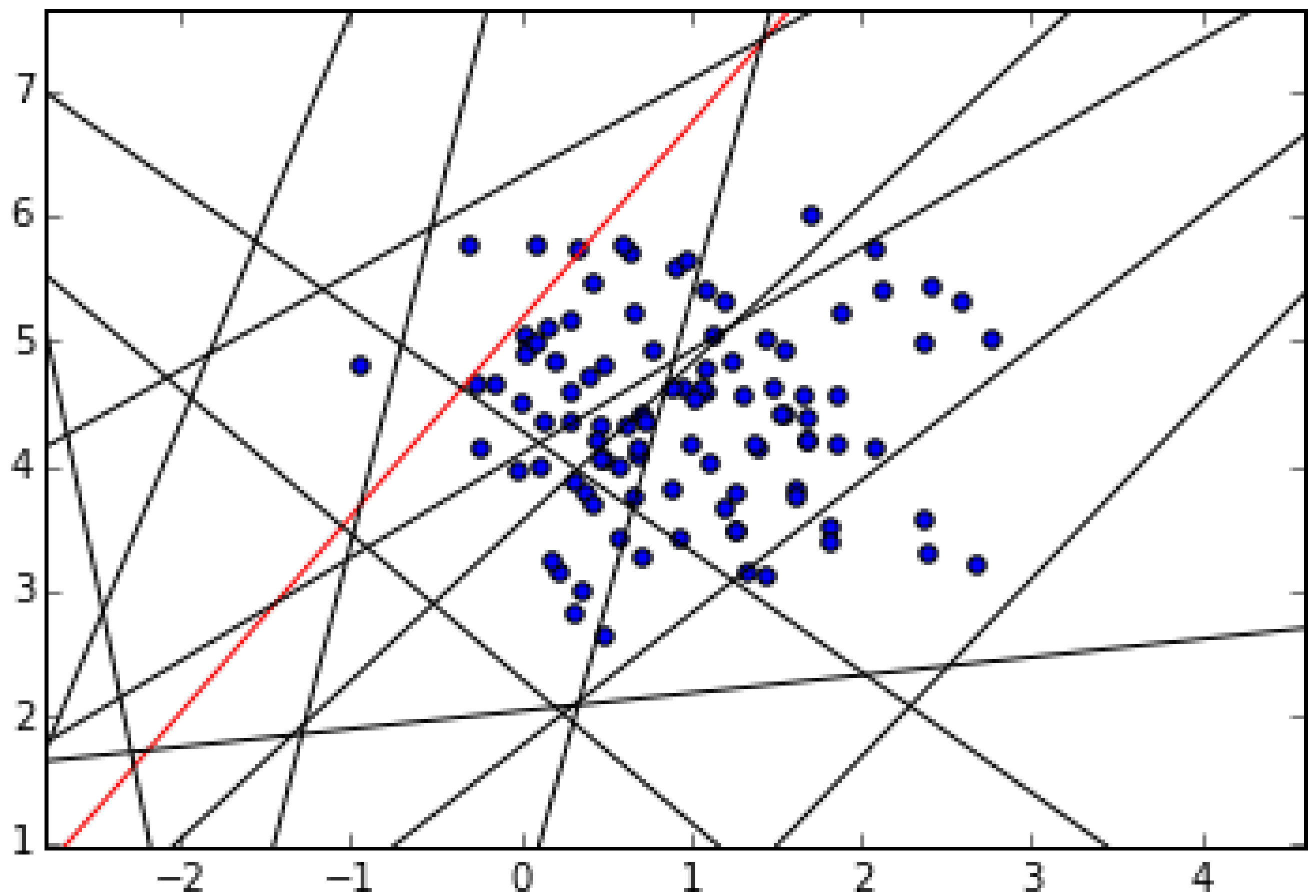
Sample Run of Hill-climbing



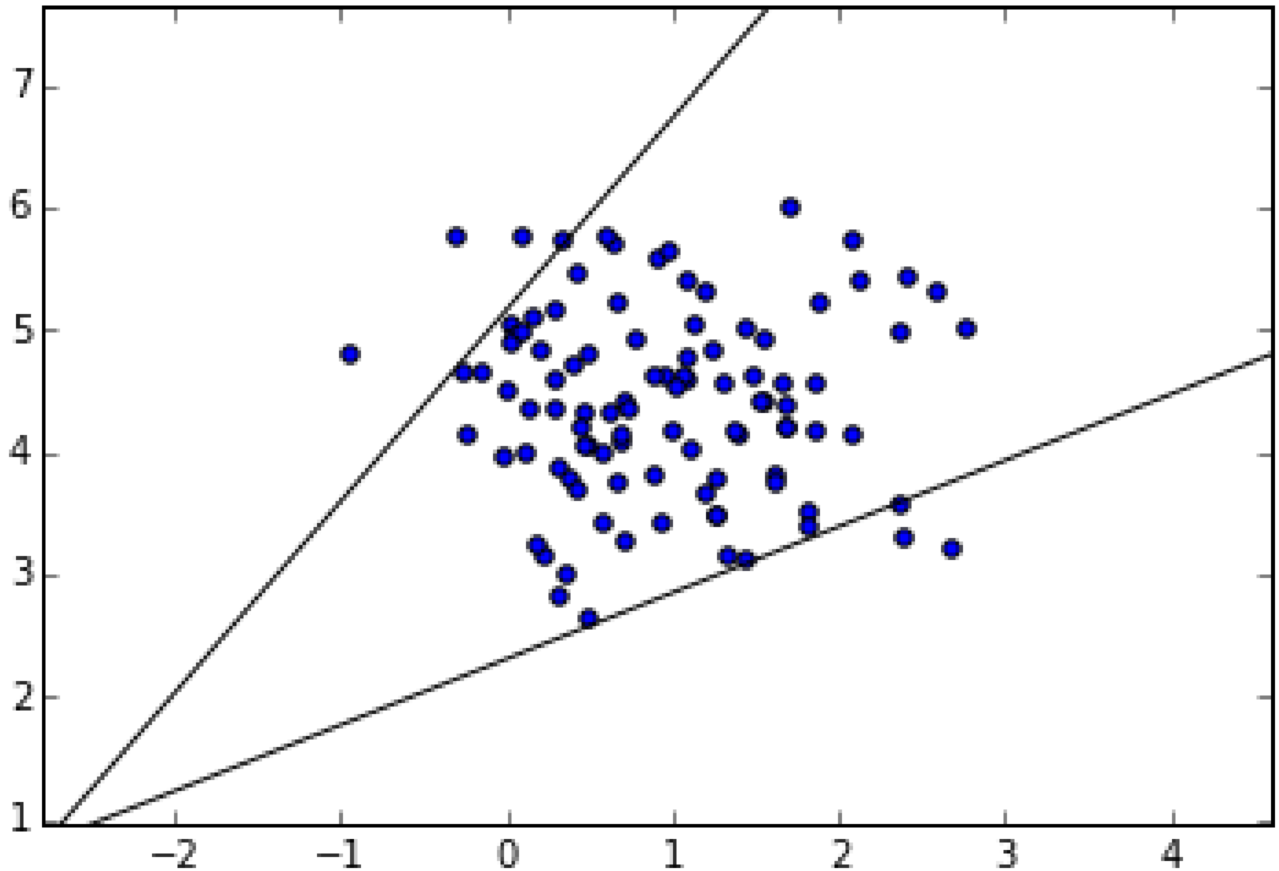
Sample Run of Hill-climbing



Sample Run of Hill-climbing

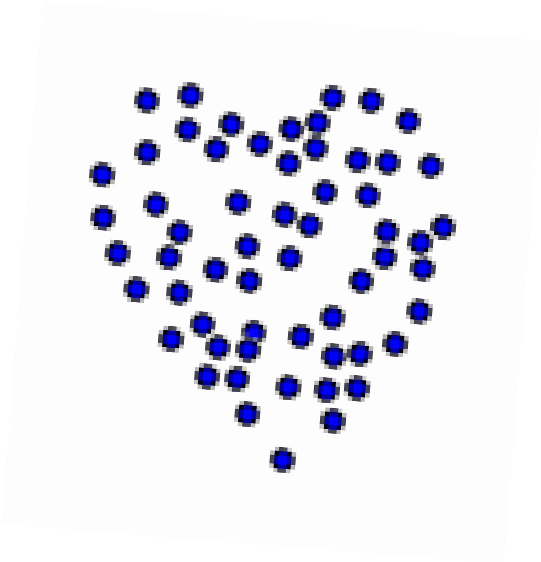


Sample Run of Hill-climbing



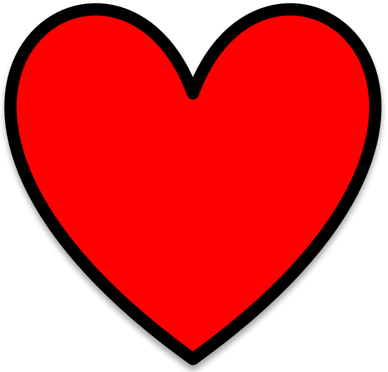
Future Work

- Evolving other shapes that aren't linear



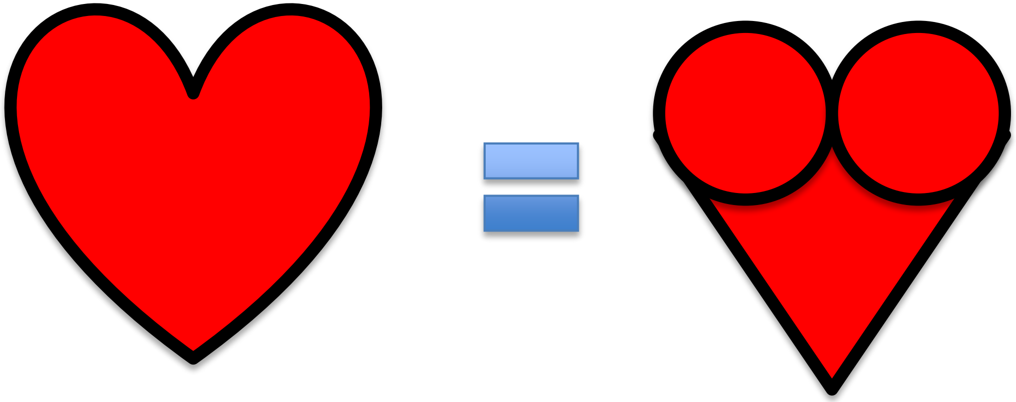
Future Work

- Evolving other shapes that aren't linear



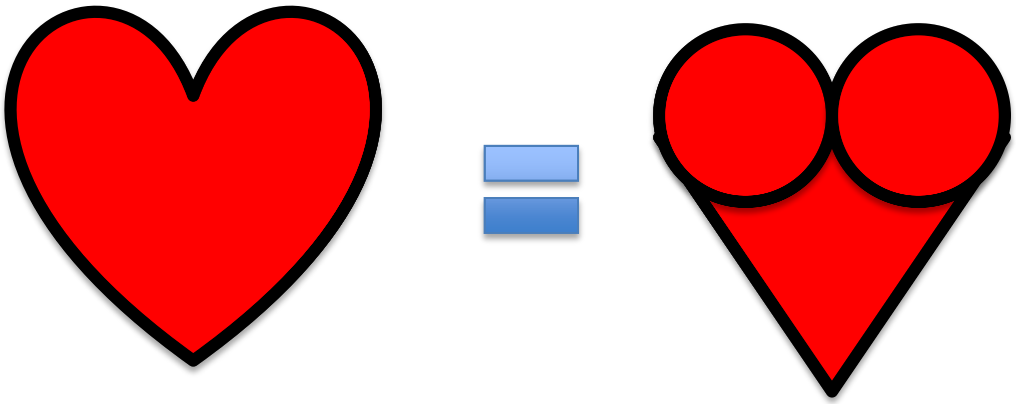
Future Work

- Evolving other shapes that aren’t linear



Future Work

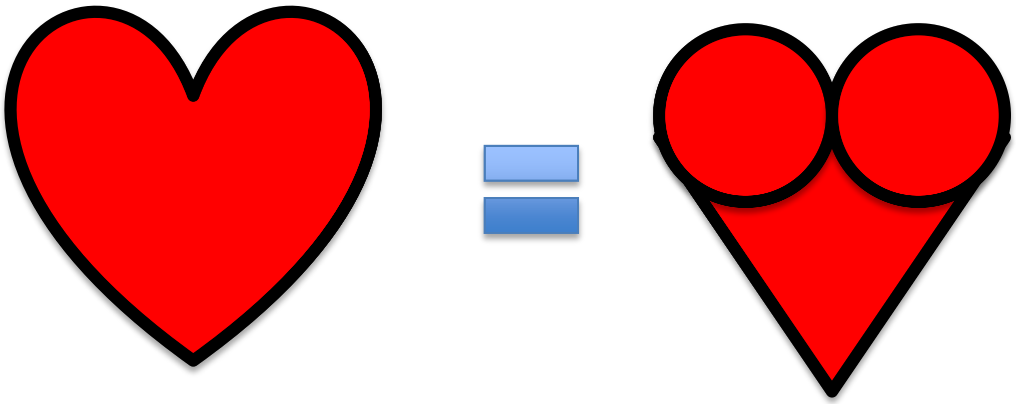
- Evolving other shapes that aren't linear



Definition of a circle: $(x-h)^2 + (y-k)^2 = r^2$.

Future Work

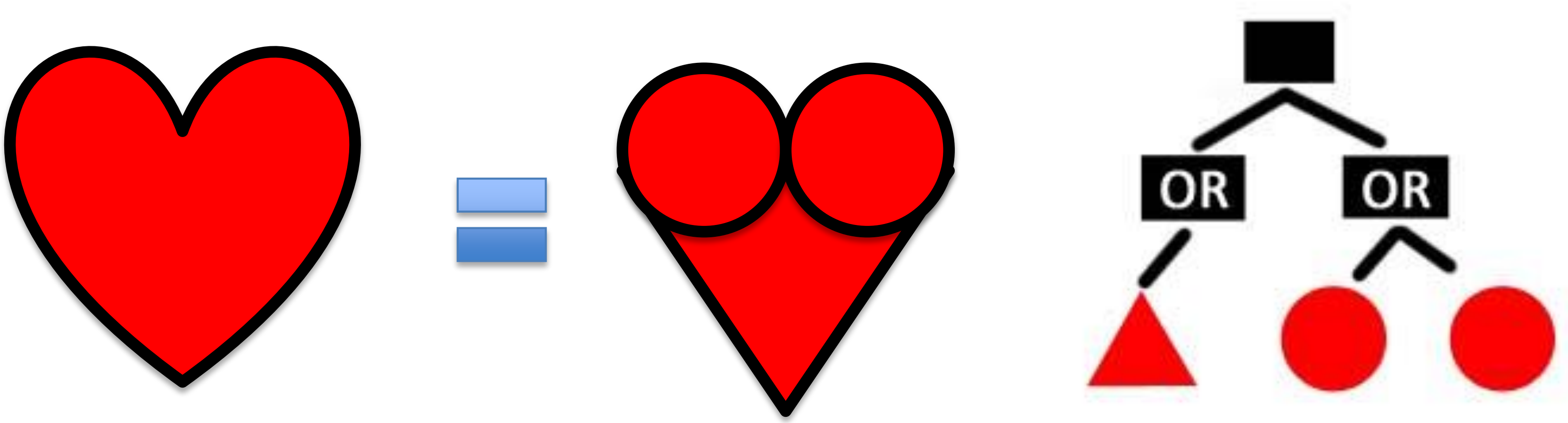
- Evolving other shapes that aren't linear



Definition of a circle: $(x-{\color{red}h})^2 + (y-{\color{red}k})^2 = {\color{red}r}^2$.

Future Work

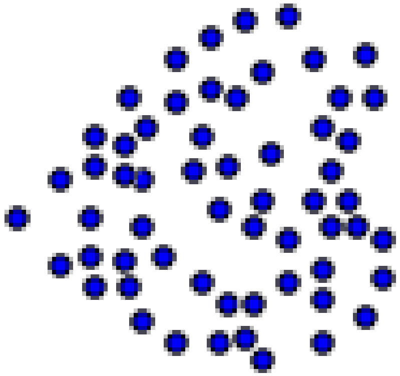
- Evolving other shapes that aren't linear



Definition of a circle: $(x-h)^2 + (y-k)^2 = r^2$.

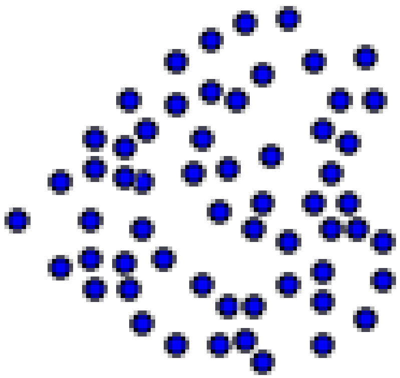
Future Work

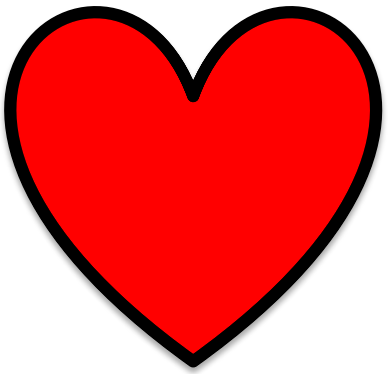
- Database look-up



Future Work

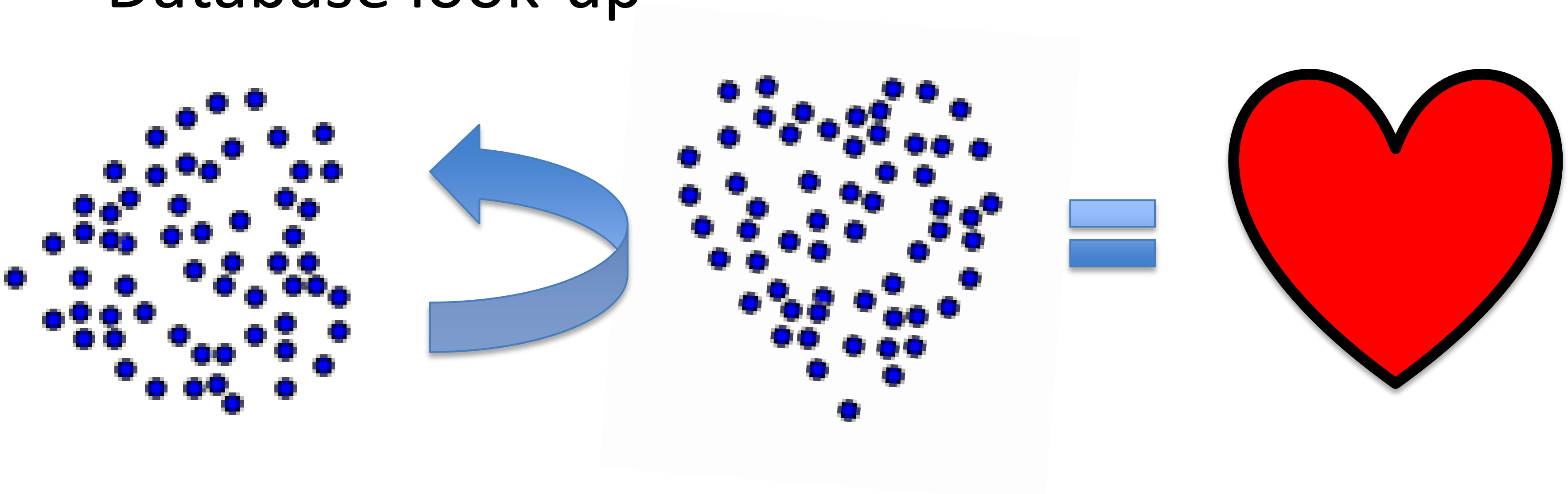
- Database look-up





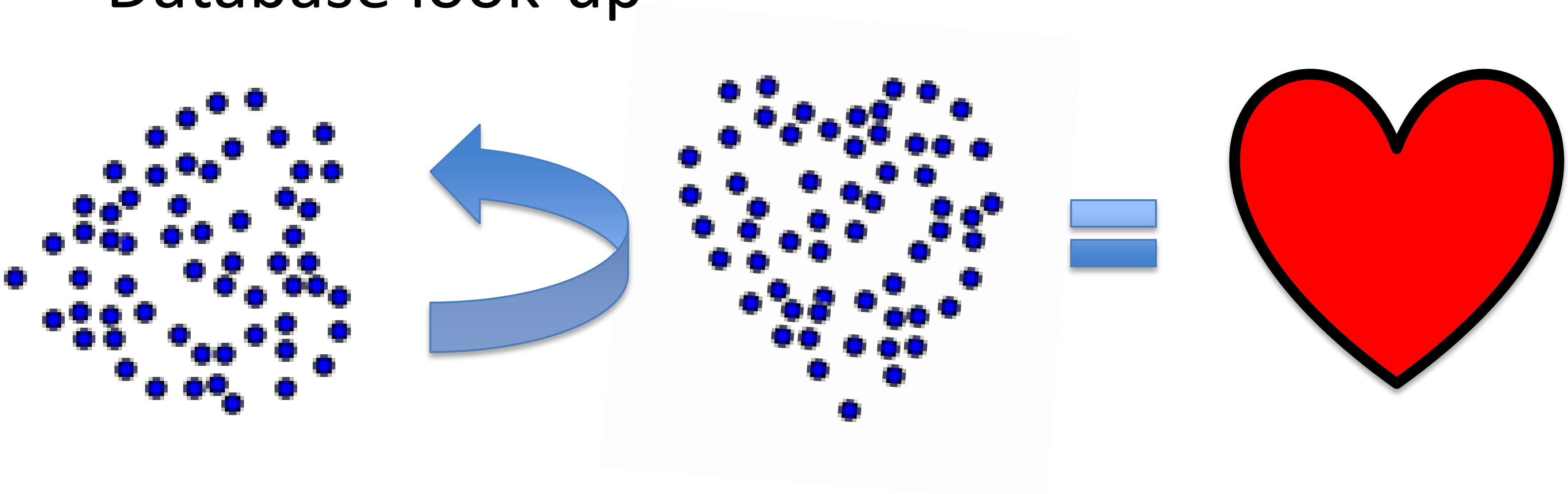
Future Work

- Database look-up

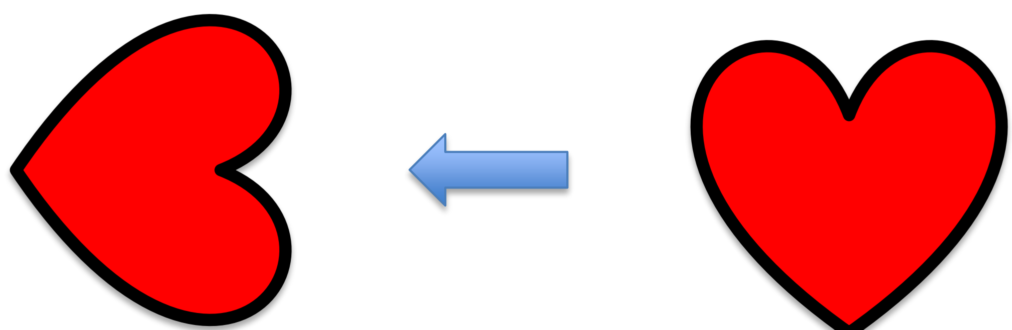


Future Work

- Database look-up



- Enables bi-directional search



Future Work

- Automatically turn results into python function
- Recode for multi-dimensional data
- Mutate parameters based on error delta
- Speed up search (aka smash into centroid)
- Concave shapes (“or” as well as “and”)
- Study initial population size, distribution
- Play with function size reward
- Density
- Look at Specificity vs. Sensitivity vs. size trade-off
  - A “three-legged stool” and difficult to tune

Backup Slides

Fitness Function

$= (((1-\alpha) + (1-\beta)) / 2)$ * function size reward

= ((specificity + power (or sensitivity))/2) * size

more on this next ↑

Where:

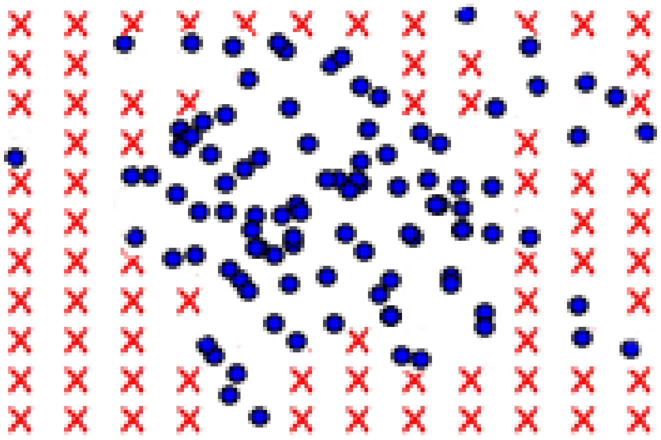
α = false positive rate

β = false negative rate

Function goes from 0 (worst) to 1 (best)

Creating Dummy Data Vs. Whitespace

- First attempt: create dummy data (with same density as class data



- Final solution: let the amount of “whitespace” determine the false positive rate (the specificity)

Crossover and Deleting/Adding Leaves

- Adding leaves
  - Once the error reaches a steady state, a new linear candidate may be added
- Deleting leaves
  - Or randomly, a candidate may be introduced that has a leaf (or an entire subtree) missing
  - Prevents overfitting

Mutation and Error Rate

- Save the previous fitness value to calculate a good “next mutation”
- Another good idea is to “smash” the line towards the centroid of the class until it hits the edge of the data