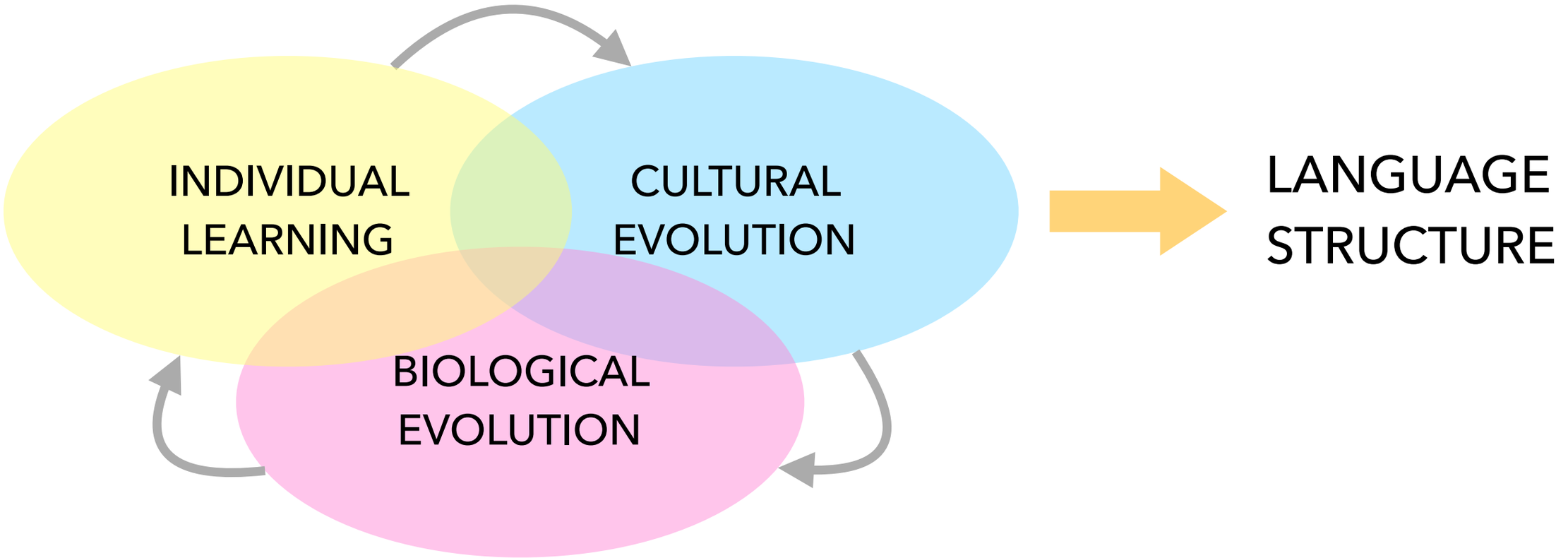
Simulating Language
Lecture 15: Summary and feedback

Simon Kirby

simon@ling.ed.ac.uk

How can we explain language structure?



Biological evolution

BIOLOGICAL EVOLUTION

Evolution of innate signalling by natural selection

Associative networks, genetic algorithms

Mutual benefit

Reciprocal altruism

Kin selection

Individual learning

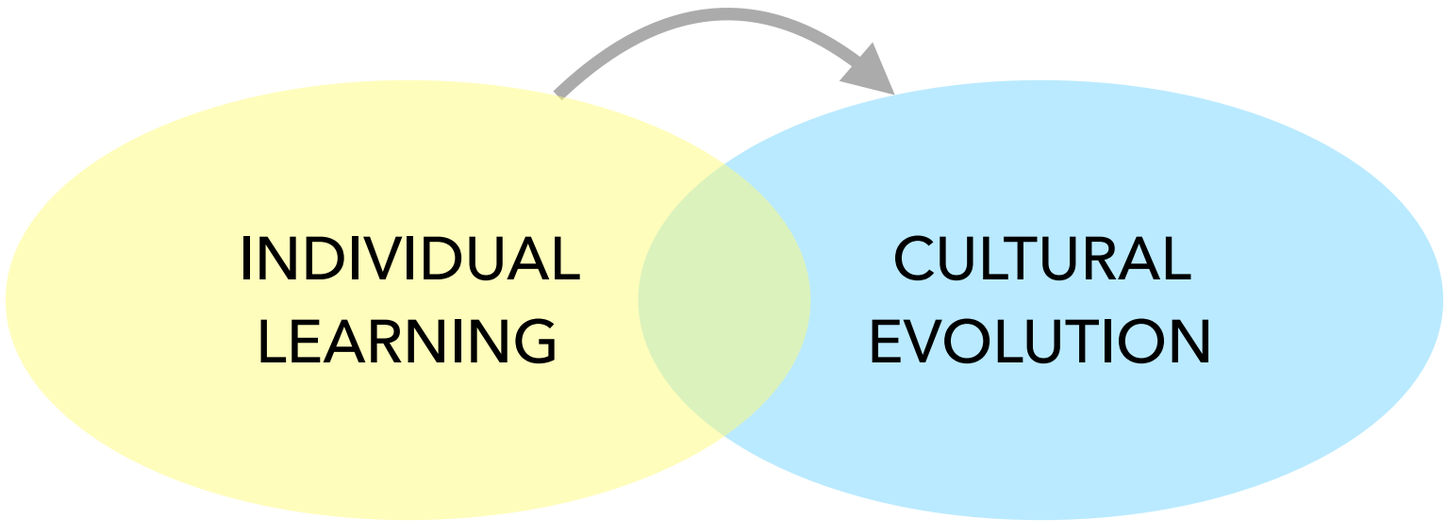
INDIVIDUAL LEARNING

Learned signalling

Weight update rules

Learning bias

Cultural evolution through iterated learning



Construction/maintenance/ acquisition

Bottlenecks

Cultural evolution of compositionality

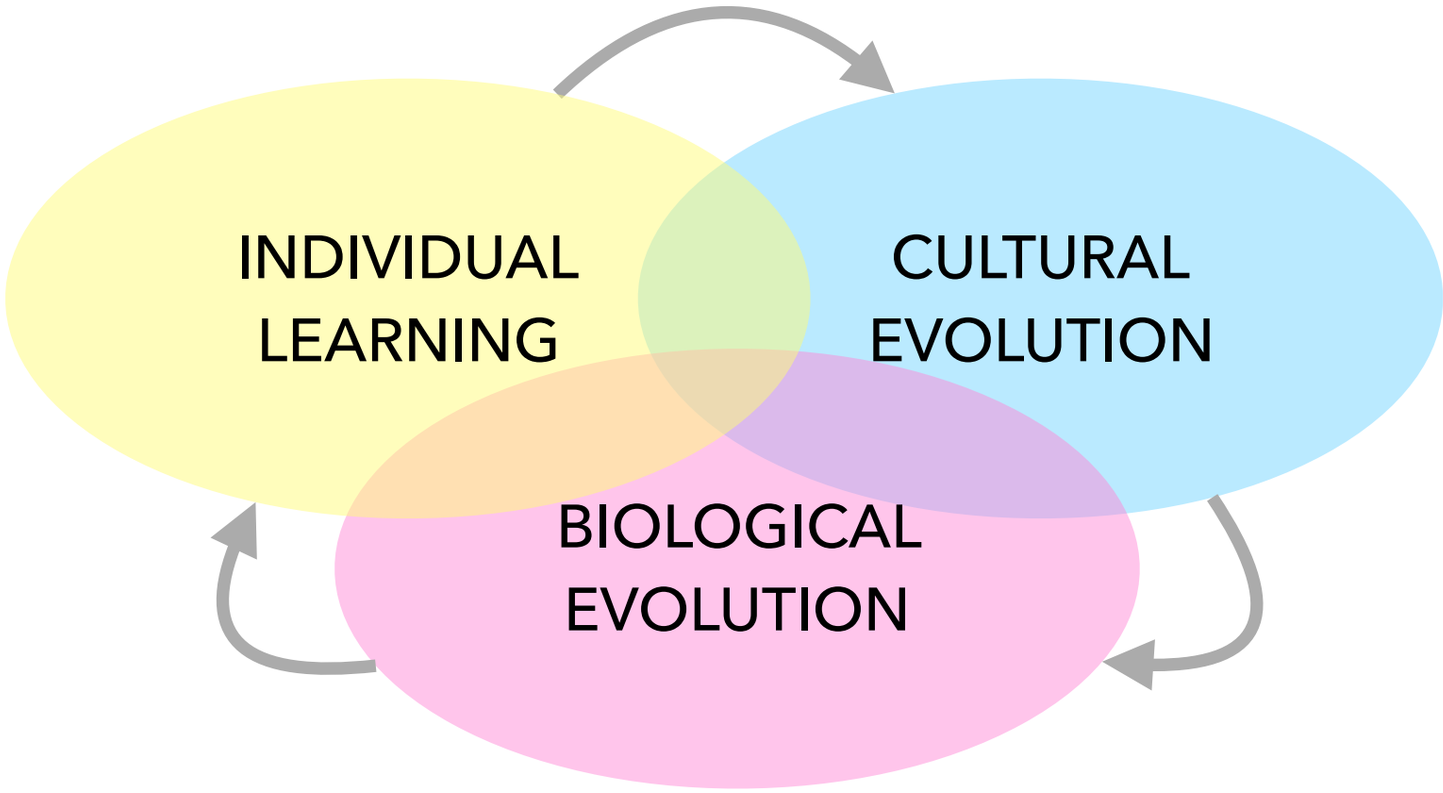
Bayesian learning

Regularisation, word order

MAP, sampling, multiple teachers

Bias amplification

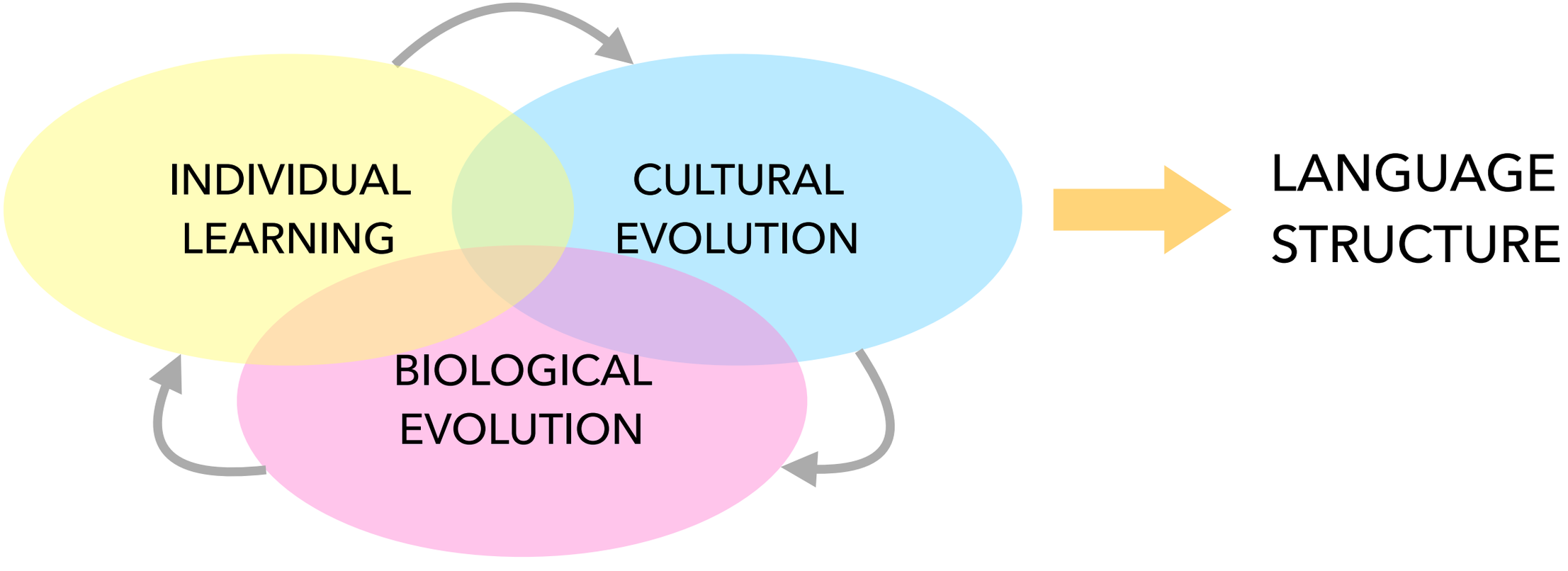
Gene-culture co-evolution



Masking, unmasking

Weak biases vs. strong constraints

Domain generality vs. specificity



My view:

The unique structural properties of language are the inevitable result of cultural evolution operating on weak, domain-general biases favouring compressible representations.

Biological evolution has given our species the capacity for culture. The rest follows for free.

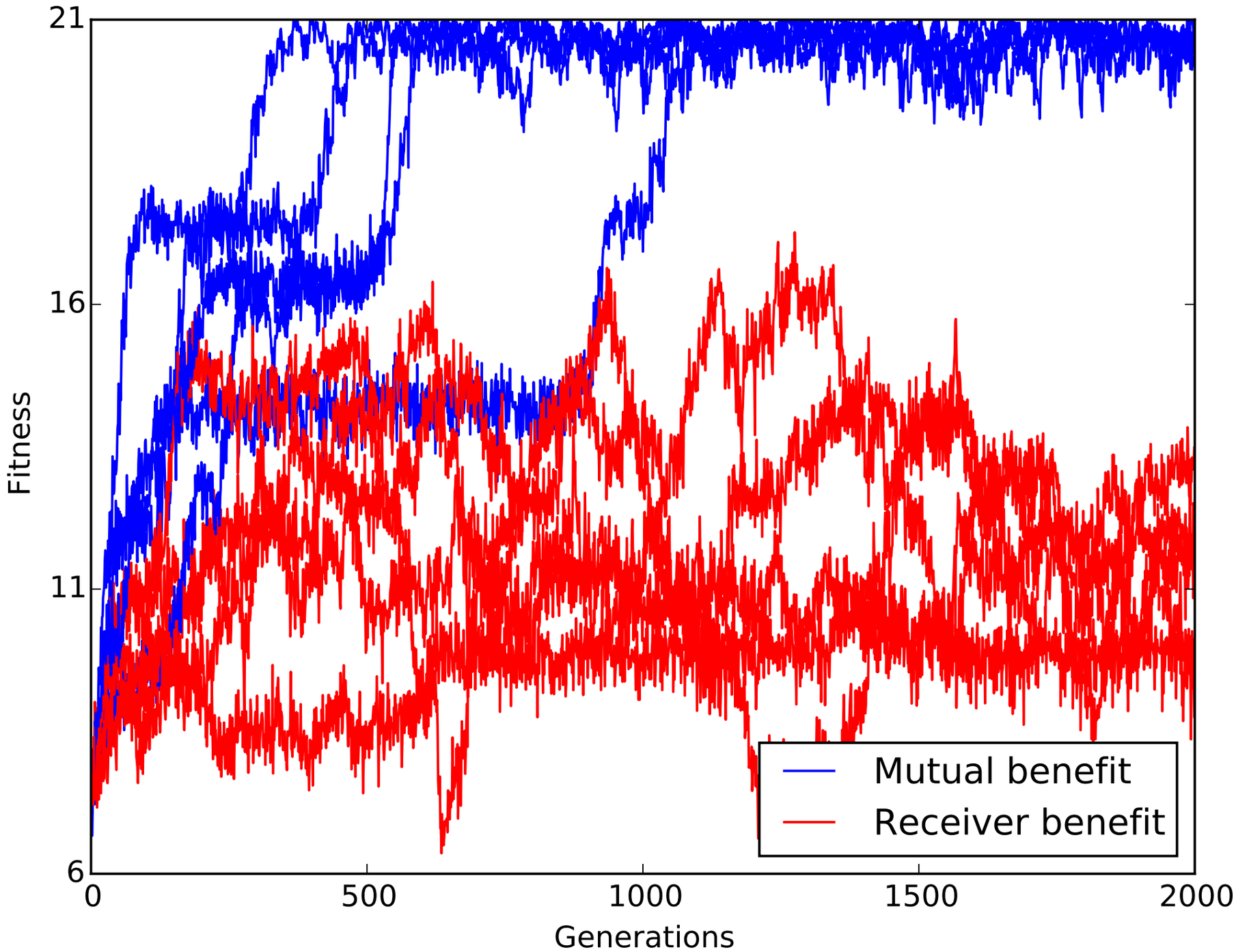
Feedback

1. What can we conclude about the evolution of communication from the simulations in Oliphant (1996)? Illustrate your answer with results from a partial replication of one of Oliphant's simulation models.

2. What is "innate" in the systems being modelled by the **evolution1.py** simulation? What about the **learning2.py** simulation? What do these correspond to in the real world? (You do not need to present any simulation results for this question.)

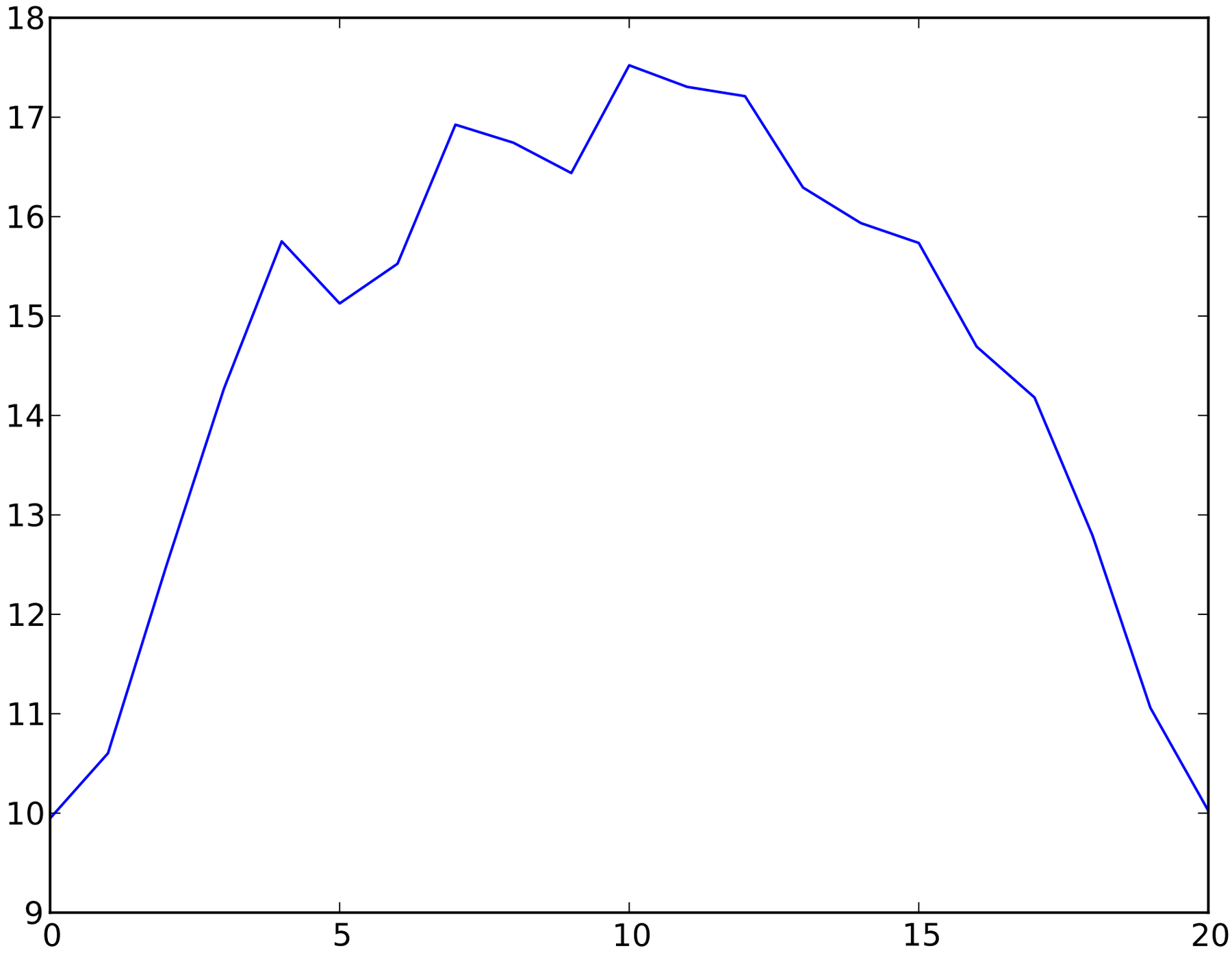
3. When Smith (2002) talks about "learners", "maintainers", and "constructors", what does he mean? Illustrate your answer with simulation results.

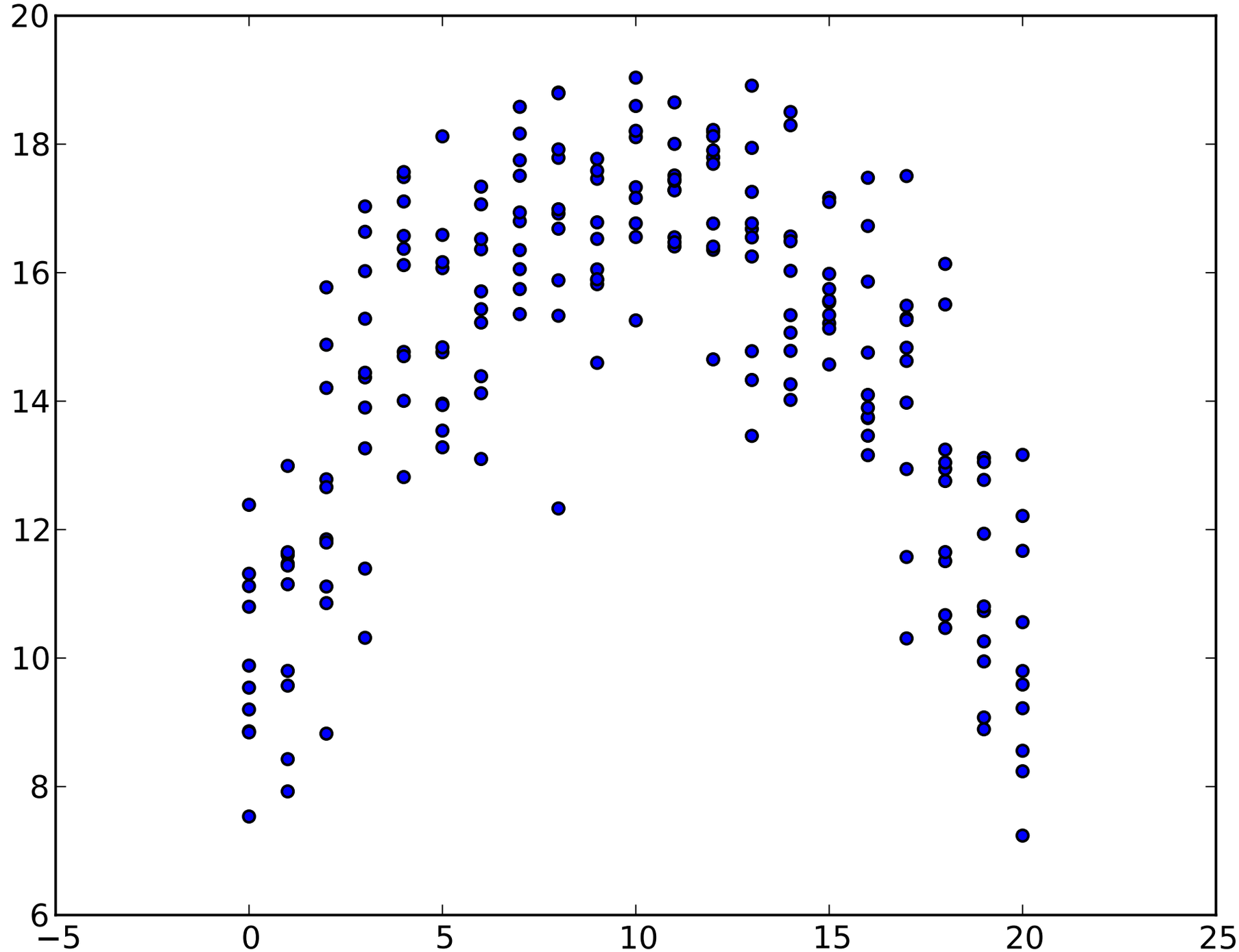
From: http://www.lel.ed.ac.uk/lec/wp-content/uploads/plotting_advice1.pdf



Add code to evolution1.py

```
def many_sims(generations,n):
    final_results=[]
    for i in range(n):
        final_results.append(simulation(generations)[1][-1])
    return final_results

all_data=[]
graph_data=[]

for send_weighting in range(21):
    receive_weighting=20-send_weighting
    res=many_sims(1000,10)
    average=sum(res)/len(res)
    print average
    all_data.append(res)
    graph_data.append(average)

plt.plot(graph_data)

plt.show()
```


18
17
16
15
14
13
12
11
10
9
0
5
10
15
20


20
18
16
14
12
10
8
6
−5
0
5
10
15
20
25