Julia で学ぶ計算論的神経科学

山本 拓都

2024 年 8 月 11 日

目次

0.1 シナプス結合強度の不均一性 4

0.1 シナプス結合強度の不均一性

(概要) シナプス結合強度は正規分布に従わず，対数正規分布のような裾の重い分布 (heavy-tailed distribution) に従う．すなわち，多くの弱いシナプス結合と少数の強いシナプス結合が存在する．これは電気生理学的な EPSP (シナプス後電位) の測定や電子顕微鏡によるコネクトームの解析により明らかになっている．本章では (Lynn, Holmes & Palmer, Nature Physics. 2024) で提案された，シナプス結合が裾の重い分布に従うネットワークを生成する発達モデルを紹介する．コードに関しては https://github.com/ChrisWLynn/Heavy_tailed_connectivity を参考に作成した．

```
using PyPlot, Random, Distributions, LinearAlgebra, StatsBase
using ProgressMeter
using Graphs, GraphPlot
using SpecialFunctions
rc("axes.spines", top=false, right=false)
```

まず，活動に非依存的なモデルを紹介する．このモデルではネットワーク全体のシナプス結合強度の和 (S) が一定になるようにシナプスの枝刈りと 2 通りの増強を行う．枝刈りの過程ではシナプスの結合強度に「よらず」，一様にシナプスが選択されて枝刈りされる．その際に減少した結合強度分を次のように他のシナプス結合に付与する．確率 p でシナプス結合強度に応じた確率でシナプスを選択し，増強する．確率 $1-p$ でランダムなシナプスを増強する．前者の過程により，強いシナプス結合はより増強される (The rich get richer).

```
function generate_scale_free_network(;
        N=200, s_avg=1, p=0.5, num_iter=1000,
        num_prune_edge=50)
    # set params
    num_max_edge = N * (N - 1)
    S = s_avg * num_max_edge

    # initialization
    edge_indices = findall((ones(Int64, N, N) - I(N)) .== 1);
    indices_sample = sample(1:num_max_edge, S, replace=false);
    A = zeros(N, N);
    A[edge_indices[indices_sample]] .= 1;

    @showprogress for t in 1:num_iter
        # pruning
        prune_indices = sample(1:num_max_edge, num_prune_edge, replace=false)
        ΔS = Int(sum(A[edge_indices[prune_indices]]))
        A[edge_indices[prune_indices]] .= 0;

        # hebbian update
        ΔS_hebb = Int(sum(rand(Binomial(1, p), ΔS)))
        hebb_indices = sample(1:num_max_edge, Weights(A[edge_indices]), ΔS_hebb, ↵
            replace=false)
```

```
            # random update
            ΔS_rand = ΔS - ΔS_hebb
            rand_indices = sample(1:num_max_edge, ΔS_rand, replace=false);

            growth_indices = [hebb_indices; rand_indices]
            A[edge_indices[growth_indices]] .+= 1;
        end

        A_flat = vcat(A[edge_indices]...);
        order_dict = sort(countmap(A_flat));
        order_arr = hcat(collect(keys(order_dict)), collect(values(order_dict)));
        order_arr[:, 2] ./= S;
        return A, A_flat, order_arr
    end
```

ニューロン数 $(N = 20)$ でグラフを描画しよう．少数の強い結合が存在していることが分かる．

```
A_small, A_small_flat, _ = generate_scale_free_network(;N=20, s_avg=1, p=0.5, ↵
    num_iter=1000, num_prune_edge=1)
```

```
g = SimpleDiGraph(A_small)
gplot(g, layout=circular_layout, arrowlengthfrac=0, edgelinewidth=A_small_flat, ↵
    plot_size=(5cm,5cm))
```

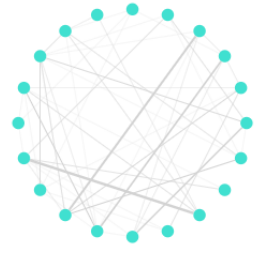

図 1　cell006.png

ニューロン数を $(N = 200)$ に増やし，各結合強度の出現頻度を描画しよう．

```
A, _, order_arr = generate_scale_free_network(N=200, s_avg=1, p=0.5, num_iter=5000, ↵
    num_prune_edge=50);
```

各結合強度の出現頻度を予測する理論的式は以下で与えられる．

```
P(s, s_avg, p) = gamma(s + s_avg * (1/p - 1)) / gamma(s + s_avg * (1/p - 1) + 1/p + 1)
s_arr = exp10.(range(1e-8, stop=2,length=50));
P_arr = P.(s_arr, 1, 0.5);
```

```
figure(figsize=(3,3))
scatter(order_arr[:, 1], order_arr[:, 2], label="Numeric")
plot(s_arr, P_arr, c="k", label="Analytic")
xscale("log")
yscale("log")
legend()
ylim(minimum(order_arr[:, 2]), )
xlabel("Connection strengths")
ylabel("Probability")
tight_layout()

#hist(A_flat, bins=exp10.(range(0,stop=2,length=10)), log=true, density=true);
#xscale("log")
```

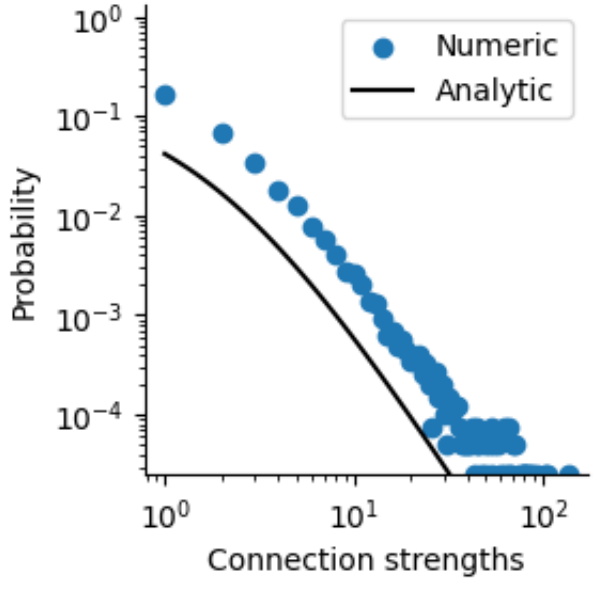

図 2　cell011.png