Immune Systems

Torregrossa François

October 31, 2016

1 Description de l'environnement

L'environnement NetLogo représente les cellules d'un individu (avec les patches oranges). Elles représentent d'une certaine manière l'état de santé d'un individu (moins il y en a, plus il est malade).
Deux agents interviennent dans cet environnement : les virus (antigenes) et les anticorps (antibodies). Bien évidemment, le but des anticorps est de lutter contre les virus afin de maintenir un maximum de cellules oranges, celui des virus est de les manger.

1.1 Paramètre et fonctions des agents de l'environnement

1.1.1 Les cellules oranges

On peut agir sur la rapidité de réapparition des cellule avec le paramètre **countdown_cell**. Comme les virus disposent d'un champs *energy* qui entraîne la mort du virus lorsqu'il atteint 0 et qu'ils rechargent en mangeant les cellules, ce paramètre permet de contrôler la quantité de virus présent dans l'environnement : un long temps de réapparition entraînera une quantité de virus réduite.

1.1.2 Les anticorps

Ils disposent des champs:

- *energy* qui sera initialisé à **energy_antibody** à leur naissance et qui les tuera lorsque *energy* atteint 0.
- *paratope* qui agit comme une clé de reconnaissance des virus (représenté par un entier).
- *epitope* qui agit comme une serrure à une clé de reconnaissance : l'anticorps est détruit s'il est rencontré par un autre ayant un *paratope* (clé) correspondant à son *epitope* (serrure).

Les anticorps ont pour fonction de détruire les virus lorsqu'ils les rencontrent à condition que les virus est un epitope correspondant au paratope de l'anticorps. Cette destruction entraîne une stimulation directe de l'anticorps qui va alors donner naissance à un nouvel

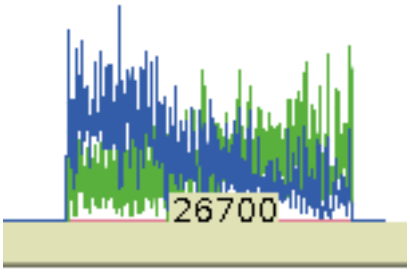

Figure 1: En vert les anticorps et en bleu les virus. Evolution du nombre d'entités en fonction du temps, en faisant varier le paramètre **energy_antibody** de 20 à 100

anticorps. C'est le seul moyen qu'ils ont pour se multiplier.
Ces derniers sont envoyés par le système immunitaire toutes les **immunity_rapidity** ticks, par lot de 10 avec un paratope "choisit" par l'algortithme selon le virus en présence. Modifier le paramètre **energy_antibody** permet de contrôler la quantité maximale d'anticorps et ainsi modifier la qualité de la réponse immunitaire.

1.1.3 Les virus

Ils disposent des même champs que les anticorps et de $countdown_r espawn$ qui donne un temps limite moyen entre la reproduction d'un virus. Ce champs est rempli par le paramètre **antigene_respawn**.
On peut changer le champ **epitope_antigene** pour faire varier l'épitope du virus en présence.
Attention! on veillera à ne pas changer ce paramètre lorsque des virus sont présents.

2 Le système immunitaire

2.1 Fonctionnement

Le système immunitaire dispose de 3 fonctions :

1. Envoyer 10 anticorps verts avec le paratope 1
2. Envoyer 10 anticorps jaunes avec le paratope 2
3. Envoyer 10 anticorps bleus avec le paratope 3

Tous les **immunity_rapidity** ticks, on décide d'en utiliser une. La sélection se fait en calculant la qualité de réponse des anticorps, i.e. en déterminant le nombre de virus qu'ils ont réussi à tuer entre deux phases de décision. C'est pourquoi, il y a trois variables globales ***green_killed***, ***yellow_killed*** et ***pink_killed***. On retient ensuite dans les matrices la réponse donnée par une fonction (= nombre de virus tués) en réponse à une attaque de virus, i.e. si elle est positive (virus tués) dans ***reward_matrix*** ou négative (aucun virus tués) dans ***penalty_matrix***, en incrémentant dans la bonne case. On mémorise égalemment dans ***specific_matrix*** la quantité de fois qu'on a fait appel à une fonction donnée en présence d'un virus donné.
Toutes ces données servent à calculer les "concentrations" des fonctions, i.e. quelle fonction utiliser en présence de quel virus. On peut faire une analogie entre notre système immunitaire et le système immunitaire réel selon l'hypothèse de Jerne.
Les fonctions sont les anticorps réels, le nombre total de virus est un virus réel, et dans la réalité, le système sélectionne un anticorps par des mécanismes de simulations successives par les virus et entre les anticorps eux-mêmes. Ces mécanismes sont représentés dans notre modèle par les matrices qui retiennent l'efficacité des fonctions. Par conséquent l'anticorps en plus grande concentration sera celui qui lutte le mieux contre le virus.
Pour calculer les concentrations des fonctions dans notre modèle (ce qui nous interesse ici puisque la fonction ayant une concentration maximale sera celle à envoyer), on utilise l'équation suivante:

$$\frac{dA_i(t)}{dt} = \left(\alpha \cdot \frac{1}{N} \cdot \sum_{j=1}^{N} m_{ij} a_j(t) - \alpha \cdot \frac{1}{M} \cdot \sum_{k=1}^{M} m_{ik} a_k(t) + \beta m_i - k_i \right) \cdot a_i(t) \qquad (2.1)$$

où :

- a_i représente la concentration de la fonction i, les deux sommes de gauche représente respectivement la stimulation par les autres fonctions et l'inhibition
- m_{ij} représente la stimulation de la fonction j par la fonction i. C'est ce terme précisément qui est calculé grace aux matrices précédentes.
- m_i représente la stimulation directe par un virus (i.e. ce terme vaut 1 lorsque le système immunitaire appelle la fonction correspondante et 0 sinon)

- k_i simule la mort naturelle des anticorps (peut être interprété comme un pourcentage de cellules qui meurent entre deux phases)
- M est le nombre de fonction inhibant i et N le nombre de fonction stimulant i. Dans notre cas $M = N$, mais il est possible d'imaginer certains cas où les fonctions ne sont pas forcément toutes inhibitrices les unes des autres.
- A_id est la concentration non normalisée a_i, il faut alors le passer dans la fonction sigmoid : $f(x) = \frac{1}{1+exp(0.5-x)}$

Pour calculer m_{ij} en présence du virus k on applique la formule suivante [2]:

$$m_{ij} = \frac{penalty\_matrix[i][k] + reward\_matrix[j][k]}{specific\_matrix[i][k] + specific\_matrix[j][k]} \tag{2.2}$$

Plus clairement :

- Au numérateur on somme la quantité de fois que la fonction i n'a pas été efficace avec le virus k avec la quantité de fois que la fonction j a été efficace avec le virus k.
- Au dénominateur, il y a le nombre de fois que chaque fonction a été appelée en présence du virus k.

On comprend assez bien que si la fonction i n'est pas efficace (grand nombre de pénalités), elle aura tendance à stimuler la fonction j, qu'elle soit efficace ou non, mais si elle l'est (important nombre de récompenses) elle aura une plus forte chance d'être appelé. L'ensemble constitue un poids entre 0 et 1 qui multipliera la concentration de la fonction i.

2.2 Paramètres liés au système immunitaire

On peut faire varier plusieurs paramètres.

- **immunity_rapidity** contrôle le temps entre deux phases de sélection de fonction. Ce paramètre permet essentiellement de réduire le nombre d'erreurs produites lors du contrôle de qualité d'une fonction (pénalité ou récompense). Plus il est grand, plus les anticorps ont le temps de tuer des virus s'ils le peuvent et de manifester à la fonction qu'ils sont positifs. En revanche, un délai trop court provoque des erreurs (affectation de pénalité à la fonction luttant contre le virus) car les anticorps n'auront pas le temps de tuer les virus. En fait, un délai trop long entraîne une convergence moins rapide vers la bonne fonction mais avec une très bonne précision, le délai court entraîne l'effet inverse.
- **natural_death** est le terme k_i de l'équation 2.1. Son rôle est relatif à la valeur de β. Si **natural_death** $<<$ β alors, la concentration des fonctions ne fera que d'augmenter parce que les autres termes ne sont pas assez fort pour faire décroitre

a_i. A l'inverse si **natural_death** $>>$ β, alors les concentrations ne feront que descendre globalement, et on n'arrivera pas à sélectionner la bonne fonction. Dans les cas où **natural_death** $\simeq$ β, on peut faire varier plus ou moins rapidement la vitesse de convergence.

- β est modifiable avec le paramètre **beta**
- α se modifie avec **alpha** et son rôle n'est pas explicitement déterminé.

References

[1] A. Ishiguro et al. A robot with a decentralized consensus-making mechanism bases on the immune system.

[2] A. Karageorgos G. Di Marzo Serugendo, M.-P. Gleizes. *Self-Organizing Software*. Springer, 2011.