Solve a major physics problem by finding how to predict the dynamics of glass-forming liquids : a challenge

Paul Bertin
Mathematiques Vision Apprentissage
ENS Cachan
paul.f.bertin@gmail.com

Abstract

Ce travail est effectué dans le cadre du cours *Face à la malédiction de la dimensionnalité* enseigné par M. Mallat au Collège de France. Nous présentons ici notre approche du challenge portant sur la prédiction de la dynamique des particules lors de la transition vitreuse. Nous considérons les particules de manière individuelle, extrayons diverses *features* puis effectuons la classification à l'aide d'un perceptron multi-couches. Afin de prendre en compte les corrélations spatiales entre les particules, nous adoptons ensuite une méthode *en cascade*.

1 Introduction

Dans ce challenge, on se propose d'étudier du point de vue du *Machine Learning* un problème qui occupe les physiciens depuis plusieurs décennies (1) : la prédiction de la dynamique des particules dans les liquides proches de la transition vitreuse. Nous disposons de 1877 exemples d'entraînement indépendants et de 510 exemples de test, chacun de ces exemples représente l'état d'un système de 1000 particules (800 de type A et 200 de type B) à un instant donné. Pour chaque particule, nous disposons donc de ses trois coordonnées de position et de vitesse. L'objectif est alors de prévoir la dynamique future des particules de type A de manière binaire : immobile ou mobile.

L'enjeux du challenge est particulièrement intéressant car il s'agit d'apprendre à appréhender les interactions entre les particules et à inférer les lois qui régissent leur dynamique. En un sens, il s'agit d'apprendre à intégrer les équations de Newton, sans justement que l'algorithme ne connaisse les équations de Newton. Des problématiques similaires ont déjà été abordées. *Battaglia et al.* (2) introduisent un *interaction network*, et *Chang et al.* (3) proposent de simuler la dynamique des objets grâce à un réseau de neurones.

Les méthodes évoquées précédemment sont complexes à mettre en oeuvre et il nous a paru trop ambitieux de tenter de les appliquer à ce problème. De plus la forme sous laquelle le problème est posé, à savoir une classification binaire, nous a amené à nous tourner vers des méthodes plus répandue telles que les perceptrons multi-couches. Néanmoins, la complexité de la tâche en elle-même (*i.e.* la fonction reliant l'état du système aux *labels* des particules) impose une étape cruciale d'extraction de *features* à la main.

Par ailleurs, au vu du relativement petit nombre d'exemples d'entraînement, nous avons décidé d'adopter une approche au niveau des particules considérées de manière individuelles, ce qui nous donne alors un ensemble d'entraînement d'environ $1,5e6$ exemples.

2 Extraction de *features*

Il s'agit d'une étape cruciale. En effet, les classifieurs que nous utilisons étant relativement simples, il sont a priori incapables d'extraire des régularités aussi complexes que celles en jeu ici uniquement à partir des positions et vitesses des particules. Il va donc falloir trouver des *indicateurs* intéressants qui fourniront au classifieur de l'information qui sera reliée de manière plus direct au label à prédire.

Nous avons exploré divers *features* décrites ci-dessous. Pour la plupart des *features*, nous considérons de manière indépendante les particules de type A et celles de type B. En vue de la classification, ces features sont mises les unes à la suite des autres de manière gloutonne.

2.1 Histogramme de densité

Comme suggéré par le *benchmark*, nous calculons pour chaque particule l'histogramme de la densité des particules en fonction de la distance à la particule. Les particules A et B sont considérées séparemment. Nous ne conservons que la partie de l'histogramme correspondant au voisinage immédiat de la particule (jusqu'aux $\frac{4}{15}^{eme}$ de la distance maximale).

2.2 Divergence des vitesses des voisins

Pour qu'une particule réussisse à sortir du puit de potentiel dans lequel elle est bloquée, on peut imaginer que ses voisins oscillent en phase, et que lorsqu'ils sont tous écartés au maximum, la particule arrive à s'échapper. Pour cela nous avons calculé pour chaque particule une sorte de divergence des vitesses des particules voisines.

Plus précisemment, pour une particule p nous notons $\vec{p_i}$ le vecteur allant de p au i^{eme} voisin. Soit $\vec{v_i}$ la vitesse du i^{eme} voisin. Deux particules sont considérées comme voisines si elles sont à une distance inférieure à d_v (que nous avons fixé arbitrairement à $d_v = 2$). Nous avons alors calculé $\sum_{voisins} \frac{\vec{p_i}}{||\vec{p_i}||} . \vec{v_i}$.

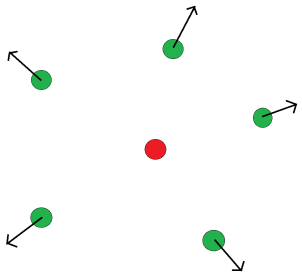

Figure 1: Une particule (en rouge) et ses voisins (en vert). Les flèches indiquent les vitesses. Ici la divergence est positive.

2.3 Estimation des forces de répulsions

De manière à mieux appréhender la dynamique, nous avons voulu fournir au classifieur une estimation de la force des voisins agissant sur la particule. Même si la forme du potentiel interatomique est complexe, nous avons modélisé les interactions de manière grossière par des forces de répulsion décroissant en $\frac{1}{r^2}$ (cas d'une répulsion électrostatique). A posteriori, nous aurions sans doute du prendre en compte les forces d'attraction également.

Comme précédemment, pour une particule p nous notons $\vec{p_i}$ le vecteur allant de p au i^{eme} voisin. Pour chaque particule, nous avons calculé le vecteur $\sum_{voisins} -\frac{K}{dist(p,i)^2} \cdot \frac{\vec{p_i}}{||\vec{p_i}||}$ où K est une constante choisie arbitrairement.

2.4 Estimation du potentiel environnant

Dans le même ordre d'idée que précedemment, nous souhaitons obtenir une estimation du paysage de potentiel environnant (généré par les voisins) pour une particule donnée. A la position x, le potentiel estimé vaut $P(x) = \sum_{voisins} \frac{K'}{dist(x,i)}$. Nous avons donc calculé l'évolution de ce potentiel le long des trois axes X, Y et Z (dans une fenêtre $[-1, 1]$ centrée autour de la particule considérée). Nous avons échantilloné tous les 0.05.

Notre espoir est que cela donne une estimation raisonnable du paysage environnant.

2.5 Valeurs agrégées

Pour chaque exemple de départ (état du sytème de 1000 particules) nous avons calculé les vitesses moyennes ainsi que la variance des vitesses. Nous avons donné ces valeurs agrégées comme *feature* à chacune des particules du système en question.

Notez que l'utilisation de ces valeurs agrégées nécessite d'être vigilant lors de la séparation *train* / *test*. Toutes les particules d'un même exemple doivent toutes être soit dans le *train* soit dans le *test*.

3 Classification

3.1 Preprocessing

Nous avons pris soin lors de la génération des *features* d'avoir des valeurs relativement cohérentes sur l'ensemble du dataset (idéalement de l'ordre de 1). Nous n'avons pas effectué de centrage ni de normalisation de nos *features* car cela avait tendance à diminuer les performances.

Cela peut se comprendre notamment dans le cas de l'histogramme de densité : les parties les plus éloignées de la particule ont tendance à avoir des valeurs plus faibles et ont également moins d'importance a priori pour la classification. Les normaliser augmente donc le bruit qu'elles contiennent. C'est pourquoi nous avons décidé de ne conserver qu'une partie de l'histogramme, toujours sans normaliser.

3.2 Classifieurs

Nous avons commencé par tester différents classifieurs sur un sous ensemble de nos données (Linear SVM, AdaBoost, Random Forest, MLP) et après avoir tenté d'optimiser les hyperparamètres pour ces différents modèles, nous avons choisi de nous tourner vers les Perceptrons multicouches (ou MLP).

Nous avons utilisé des réseaux de neurones avec des couches complètement connectées et relativement peu profonds (deux couches cachées minimum). En effet, utiliser des réseaux plus profonds demande d'être très précautionneux (problème de *vanishing gradient* etc).

Nous avons utilisé l'algorithme de descente de gradient ADAM (4) avec les paramètres $\beta_1 = 0.9$, $\beta_2 = 0.999$ et $\epsilon = 1e - 8$ ainsi qu'une régularisation de norme 2 avec $\lambda = 0.03$. Nous arrêtons l'algorithme au bout de 500 itérations.

3.3 Résultats

λ	**hidden layer sizes**	**train acc.**	**test acc.**
0.03	(100)	0.599	0.586
0.05	(100, 200)	0.597	0.583
0.03	(200)	0.597	0.584

Table 1: Parameters and Accuracies for the different multilayer perceptron models

On présente ici certains des meilleurs resultats obtenus avec les perceptrons multicouches. On remarque que les *accuracies* sont très similaires, ce qui indique a priori que l'information contenue dans nos *features* qui est susceptible d'être extraite par nos modèles de perceptrons est insuffisante.

Il faudrait donc soit utiliser des modèles radicalement plus sophistiqués, soit extraire de nouvelles *features*.

4 Mise en cascade

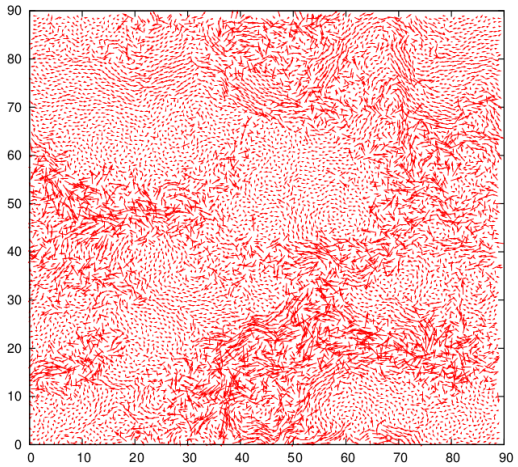

Figure 2: Tiré de (1). Visualisation des déplacements des particules lors d'une simulation. On observe des corrélations spatiales entre les dynamiques des particules.

La littérature nous indique qu'il y a des corrélations spatiales entre les dynamiques des particules. La grande faiblesse de notre modèle jusqu'à présent est qu'il ne prend pas en compte ces corrélations.

Pour y remédier, nous avons adopté une méthode de classification *en cascade*. Un premier classifieur est entrainé. Ensuite pour chaque atome, nous ajoutons comme *feature* la probabilité prédite pour ses quatres plus proches voisins. Notez que le nombre de voisins pris en compte a été fixé arbitrairement. Nous réentrainons ensuite un deuxième classifieur avec ces *features* additionnelles.

Nous pouvons répéter ce procédé autant de fois qu'on le souhaite. Expérimentalement, nous avons observé une amélioration d'environ 1% de l'accuracy, jusqu'à saturer à environ 0.595.

L'idée est qu'à chaque itération, on augmente le champs réceptif de chaque particule. A la première itération, le champs réceptif correspond à ses quatre plus proches voisins. A la deuxième itération, le champs réceptif correspond aux quatre plus proches voisins ainsi qu'aux voisins des voisins.

itération	λ	**hidden layer sizes**	**train acc.**	**test acc.**
1	0.03	(100)	0.599	0.586
2	0.03	(100)	0.601	0.589
3	0.03	(100)	0.602	0.590
4	0.03	(100)	0.604	0.595
5	0.03	(100)	0.604	0.594

Table 2: Parameters and Accuracies for the different multilayer perceptron models

5 Conclusion

En extrayant des features plus avancées et en prenant en compte les corrélations entre particules, nous avons réussi à améliorer le *benchmark*. Nous sommes à présent quasiment à 10% au-dessus du hasard au lieu de 6%. Sur la plateforme, nous avons réussi à obtenir un score de 0.599623 (1^{ere} position).

Néanmoins, ces résultats restent très insatisfaisants, et il serait judicieux d'essayer d'inférer les interactions physiques des particules.

References

[1] L. Berthier and G. Biroli, "Theoretical perspective on the glass transition and amorphous materials," *Rev. Mod. Phys.*, vol. 83, pp. 587–645, Jun 2011.

[2] P. W. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. Kavukcuoglu, "Interaction networks for learning about objects, relations and physics," *CoRR*, vol. abs/1612.00222, 2016.

[3] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum, "A compositional object-based approach to learning physical dynamics," *CoRR*, vol. abs/1612.00341, 2016.

[4] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.