FINAL PROJECT REPORT

JARUPULA GOPAL
EP18BTECH11004

Unsupervised learning for local structure detection in colloidal systems

AIM

We introduce a simple, fast, and easy to implement unsupervised learning algorithm for detecting different local environments on a single-particle level in colloidal systems

INTRODUCTION

We present a new avenue to detect self-assembly products and introduce an unsupervised machine learning algorithm based on bond-orientational order parameters combined with neural-network-based autoencoders and Gaussian mixture models (GMMs). Autoencoders are a standard technique for nonlinear dimensionality reduction, while mixture models are probabilistic models for identifying distinct clusters within a data set. Using these methods, our algorithm can autonomously classify particles in different groups based on their local order, making it easy to detect any self-assembly product in the system. The goal here is to identify local environments on a single-particle level—meaning that this method can be used to study processes such as nucleation, grain boundary characteristics, and coexistences. This algorithm has been designed to be computationally fast, easily scalable to very large data sets, and extremely easy to implement. To test its performance, we examine a number of different colloidal systems, ranging from spheres to binary mixtures to anisotropic particles

ABOUT ALGORITHM

In this section, we describe in detail the algorithm we use to classify local environments. We start by summarizing the main steps of our approach and then follow with a detailed description of each step in separate subsections. **The overall process consists of three step**s. First, we require a method to capture the local environment of each particle in a set of *local order parameters*. For this, we make use of bond orientational order parameters. This set of order parameters is in general high-dimensional and may contain significant amounts of redundant and irrelevant information. In order to extract the most relevant information, the second step of our approach makes use of a **dimensionality reduction technique,** namely, **a neural-network based autoencoder**. Once trained, the autoencoder projects the original (high-dimensional) input vectors onto a lower-dimensional subspace encoding the features with the largest variations in the input data. Ideally, in this subspace, particles with similar local environments are grouped together. Finally, we apply a **clustering algorithm** (Gaussian mixture models) in order to identify the distinct clusters of local environments in this lower-dimensional subspace

Bond order parameters

To characterize the local environment of each particle, we use the averaged bond order parameters (**BOPs)**. First, we define for any given particle i the complex quantities

$$q_{lm}(i) = \frac{1}{N_b(i)} \sum_{j \in \mathcal{N}_b(i)} Y_l^m(\mathbf{r}_{ij}),$$

where Y m l (rij) are the spherical harmonics of order l, with m an integer that runs from m = −l to m = +l. Additionally, rij is the vector from particle i to particle j, and Nb(i) is the set of nearest neighbors of particle i, which we will define later. Note that Nb(i) contains Nb(i) particles. Then, we can define an average q¯lm(i) as

$$\bar{q}_{lm}(i) = \frac{1}{N_b(i)+1} \sum_{k \in \{i, \mathcal{N}_b(i)\}} q_{lm}(k),$$

where the sum runs over all nearest neighbors of particle i as well as particle i itself. Averaging over the nearest neighbor values of qlm results effectively in also taking next-nearest neighbors into account. Finally, we define rotationally invariant BOPs as

$$\bar{q}_l(i) = \sqrt{\frac{4\pi}{2l+1} \sum_{m=-l}^{l} |\bar{q}_{lm}(i)|^2},$$

which, depending on the choice of l, are sensitive to different crystal symmetries. The optimal set of BOPs to be considered strongly depends on the structures one wishes to distinguish. Since our method is meant to be applied to systems for which such prior knowledge is missing, in order to describe the local environment of one particle, we evaluate several q¯l with l ranging from 1 to 8. Note that in principle, one could consider a larger (or smaller) range of l. For all cases examined in this paper, however, we found 8 to be sufficient. Therefore, when considering one component systems, our description of the local environment of particle i is encoded into an 8-dimensional vector

Q(i) = ({q¯l(i)})

with l ∈ [1, 8]. When considering binary mixtures, i.e., systems with two species of particles, the same BOPs are evaluated both considering all the nearest neighbors of the reference particle (regardless of particles' species) and considering only the nearest neighbors of the same species as the reference particle. Hence, for binary mixtures, our description of the local environment of particle i is encoded into a 16-dimensional vector,

Q(i) = ({q¯l(i)}, {q¯ ss l (i)}),

where s indicates the particles' species. Here, {q¯l} represents the set of BOPs evaluated considering all the nearest neighbors of particle i, while the set {q¯ ss l } is evaluated considering only the nearest neighbors of the same species s as particle i. Thus far, we have not discussed the definition of a nearest neighbor, as used in the definition of the BOPs.

Unsupervised learning

What is Unsupervised Learning?

: Unsupervised learning *is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision.*

Why use Unsupervised Learning?

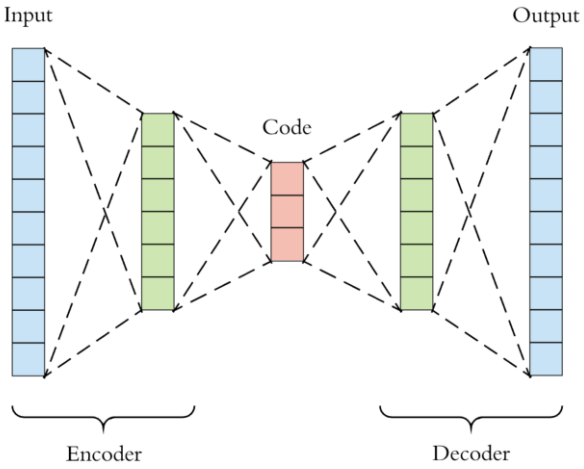
Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to **find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format**.

Nonlinear dimensionality reduction using neural-network-based autoencoders

In order to extract the relevant information contained in the vectors **Q(i),** we use neural-network-based autoencoders., we use neural-network-based autoencoders. An autoencoder is a neural network that is trained to perform the identity mapping, where the network inputs are reproduced at the output layer.

AUTOENCODERS

: An **autoencoder** is a type of artificial neural network used to learn efficient data codings in an unsupervised manner. The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore signal "noise".



The network may be viewed as consisting of two parts: an **encoder network**, which performs a nonlinear projection of the input data onto a low-dimensional subspace, and

A **decoder network** that attempts to reconstruct the input data from the low-dimensional projection.

Training of Autoencoders:

This is a more detailed visualization of an autoencoder. First the input passes through the encoder, which is a fully-connected ANN, to produce the code. The decoder, which has the similar ANN structure, then produces the output only using the code. The goal is to get an output identical with the input. Note that the decoder architecture is the mirror image of the encoder. This is not a requirement but it's typically the case. The only requirement is the dimensionality of the input and output needs to be the same. Anything in the middle can be played with.

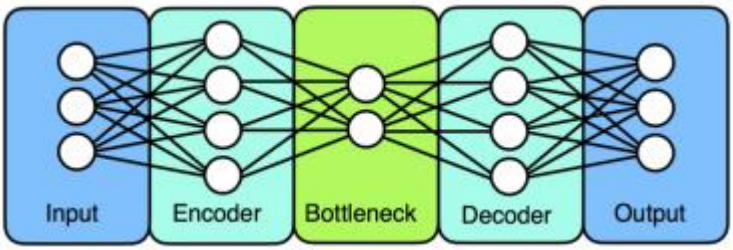
There are 4 hyperparameters that we need to set before training an autoencoder:

- **Code size**: number of nodes in the middle layer. Smaller size results in more compression.
- **Number of layers**: the autoencoder can be as deep as we like. In the figure above we have 2 layers in both the encoder and decoder, without considering the input and output.
- **Number of nodes per layer**: the autoencoder architecture we're working on is called a *stacked autoencoder* since the layers are stacked one after another. Usually stacked autoencoders look like a "sandwich". The number of nodes per layer decreases with each subsequent layer of the encoder, and increases back in the decoder. Also, the decoder is symmetric to the encoder in terms of layer structure.
- **Loss function**: we use *mean squared error (MSE).*

By training the autoencoder to perform the input reconstruction task over an ensemble of training examples, the encoder is forced to learn a low-dimensional nonlinear projection that

preserves the most relevant features of the data and from which the higher-dimensional inputs can be approximately reconstructed by the decoder.

we employ feedforward and fully connected autoencoders like the one presented in the figure.



The number of **input and output nodes, d**, is specified by the dimensionality of the input vectors, Q(i) ∈ R d, which are approximately reconstructed by the network in the output layer, Qˆ (i) ∈ R d. The bottleneck layer contains the low-dimensional projection to be learned by the encoder, Y(i) ∈ R c, whose dimensionality is controlled by the number of bottleneck nodes,c < d. **Nonlinearity is achieved by providing both the encoder and the decoder with a fully connected hidden layer with a nonlinear activation function**. Here, we set the number of nodes in the hidden layers to 10d and use a hyperbolic tangent as the activation function. For the bottleneck and output layers, instead, a linear activation function is used.

The internal parameters of the autoencoder, i.e., weights **W ≡ {wj}** and biases **B ≡ {bk},** are initialized with the normalized initialization., the biases are initialized to zero, while the weights are drawn from a normal distribution with zero mean and a variance that depends on the size of the layers. During the training, these parameters are optimized by minimizing the reconstruction error of the input data over a training set of N training examples. Specifically, we consider **the mean squared error (MSE) with the addition of a weight decay regularization term to control the magnitude of the network weights.**

$$E(\mathbf{W},\boldsymbol{B};\{\mathbf{Q}(i)\}) = \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{Q}(i)-\hat{\mathbf{Q}}(i)\right\|^2 + \lambda\sum_{j=1}^{M} w_j^2,$$

where M is the total number of weights, whose value depends on the dimension of the network, and we set λ = 10^−5. The function is minimized using minibatch stochastic gradient descent with momentum.

The optimal number of nodes in the bottleneck layer, c, which defines the unknown relevant dimensionality of the input data, can be determined by computing the reconstruction MSE and

looking for the existence of an elbow in the MSE as a function ofc. For convenience, we rescale the MSE by the **mean squared deviation (MSD)** of the vectors **Q(i)**

$$\mathrm{MSD} = \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{Q}(i) - \bar{\mathbf{Q}}\right\|^2,$$

where $\bar{Q}$ is the mean input vector. To detect the presence of an elbow, we use the L-method proposed by Salvador and Chan. For all systems examined in this work, we found a dimensionality ofc = 2 to be sufficient.

Once the autoencoder is trained, the encoder network alone is retained in order to perform the nonlinear mapping of the input vectors **Q(i)** onto the low-dimensional subspace defined by the bottleneck layer, **Y(i).**

Learning from the autoencoder

One of the main advantages of using a neural-network-based autoencoder over other nonlinear techniques for dimensionality reduction is that it furnishes an exact analytical mapping (and an approximate inverse mapping) between the original input space and its low-dimensional projection. In finding such a mapping, the autoencoder must understand which of all the BOPs given as input in the vectors Q(i) are the most relevant for the system under analysis. Extrapolating this knowledge would help us understand the relevant symmetries distinguishing the different environments possibly present in the system.

Several methods to assess the relative importance of input variables in neural network models have been proposed. Here, we consider the input perturbation and the improved stepwise methods. Both techniques require the use of a single trained model, avoiding having to repeat the training of the autoencoder multiple times

The input perturbation method assesses the variation in the MSE of the autoencoder by adding, in turn, a small amount of white noise to the kth input while holding all the other inputs at their observed values. Here, we set the white noise to 10% and 50% of each input. The input variables whose changes affect the output the most, leading to a large increase in the MSE, are the ones that have the most relative influence.

The improved stepwise method is very similar in spirit, but instead of adding noise to one of the inputs, it replaces all its values with its mean over the whole dataset. Also, in this case, the most relevant input variables are identified as the ones whose replacement causes the largest increase in the MSE.

In both cases, a quantitative measure of the relative importance, RIk , of the kth input can be obtained as

$$\mathrm{RI}_k = \frac{\Delta E_k}{\sum_{j=1}^{d} \Delta E_j},$$

where ΔEk is the variation in the MSE caused by the change applied to the kth input and the sum in the denominator runs over all the input variables.

Clustering:

Clustering is an unsupervised machine learning method of identifying and grouping similar data points in larger datasets without concern for the specific outcome. Clustering (sometimes called cluster analysis) is usually used to classify data into structures that are more easily understood and manipulated.

Gaussian mixture models (GMMs):

GMM is a probabilistic model that assumes that the observed data are generated from a mixture of a finite number of Gaussian distributions with unknown parameters.

Such parameters are optimized iteratively with the expectation-maximization (EM) algorithm in order to create a probability density function that agrees well with the distribution of the data. The number of Gaussian components in the mixture, **NG,** is usually found by minimizing the **Bayesian information criterion (BIC)**, which measures how well a GMM fits the observed data while penalizing models with many parameters to prevent overfitting. The output of a trained GMM is a list of probabilities, pij, corresponding to the posterior probabilities of the ith observation to arise from the jth component in the mixture model.

Elbow method (clustering)

In cluster analysis, the elbow method is a heuristic used in determining the number of clusters in a data set. The method consists of plotting the explained variation as a function of the number of clusters, and picking the elbow of the curve as the number of clusters to use. The same method can be used to choose the number of parameters in other data-driven models, such as the number of principal components to describe a data set.

In order to cluster together similar environments in the lowdimensional subspace found by the encoder, we use Gaussian mixture models (GMMs) as implemented in scikit-learn.

Scikit-learn is probably the most useful library for machine **learning** in Python. The **sk-learn** library contains a lot of efficient tools for machine **learning** and statistical modeling including classification, regression, clustering and dimensionality reduction.

The simplest form of clustering that can be applied consists in considering each mixture component as generating a separate cluster and assigning each observation to the component

with the highest posterior probability. However, while this procedure works perfectly for clusters that are really generated from a mixture of separate multivariate normal distributions, the clusters that underline our data are very often far from being Gaussian-distributed in space.

As a consequence, a single cluster in the data may be detected as two or more mixture components (if its distribution is indeed better approximated by a mixture of Gaussians than by a single Gaussian function), meaning that the number of clusters in the data may in general be different from the number of components found by minimizing the BIC.

To overcome this problem, we use the method proposed by Baudry et al. The idea is to first use the BIC in order to find a GMM with NG components that fits the data well. Then, a sequence of candidate clustering's with K = NG, NG − 1, . . ., 1 cluster is formed by successively merging a pair of components. At each step, the two mixture components to be merged are chosen so as to minimize the entropy of the resulting clustering, defined as

$$S_K = -\sum_{i=1}^{N}\sum_{j=1}^{K} p_{ij}\ln(p_{ij}),$$

where N is the number of observations and K is the number of clusters. Finally, the optimal number of clusters is found by looking for the existence of an elbow in the entropy SK as a function of K. Again, we detect the elbow with the L-method of Salvador and Chan.

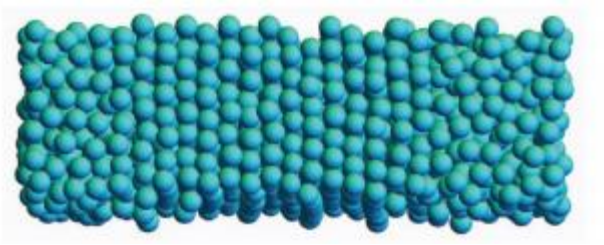
This procedure autonomously finds the number of clusters underlying the data, corresponding to the distinct particle environments present in the system under analysis.

RESULTS AND DISCUSSION

In this section, We first present in detail the whole procedure for the analysis of a “test” example for which we compare the results with a more standard, system-specific, methodology. A shorter, more concise, discussion is dedicated to the results obtained for the other systems analyzed, including systems with grain boundaries, anisotropic hard cubes, and binary mixtures.

Single-component hard spheres

we examine a snapshot from a **Monte Carlo (MC) simulation** of single-component hard spheres of diameter σ, which is

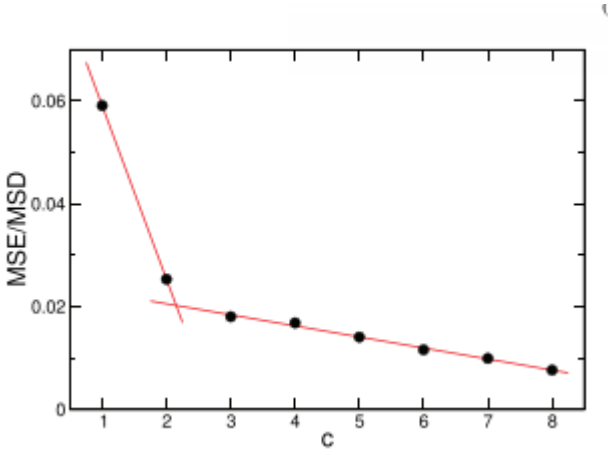


The simulation was performed in the canonical ensemble (constant number of particles N, volume V, and temperature T), which was initialized as a metastable coexistence between the fluid, hexagonal close-packed (HCP), and face-centered cubic (FCC) phases. The system contained N = 1536 particles and was at a number density ρσ3 = 1.01.

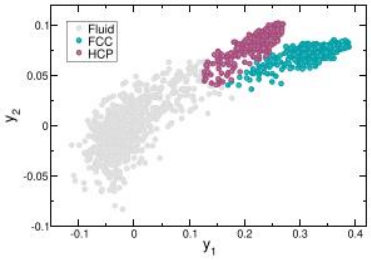
1.Analysis

Starting from the raw coordinates of each particle i in the system, we build the vectors Q(i) in Eq. 4 and use them as an input for the autoencoder. To find the optimal dimensionality of the bottleneck layer, c, we evaluate the reconstruction MSE of the autoencoder for c ∈ [1, 8].

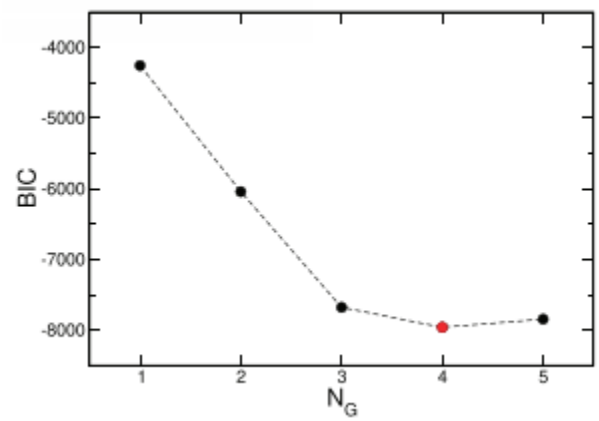
A plot of the rescaled MSE as a function of c is shown:



Solid lines, obtained with the L-method, clearly show the presence of an elbow at c = 2, indicating that a two-dimensional projection of the original input vectors is sufficient to preserve the relevant information. The projection learned by the encoder is depicted in



We then apply GMM in this two-dimensional space in order to identify the relevant clusters, i.e., the distinct particle environments. Following the method of Baudry et al., we first optimize the number of Gaussian components in the mixture model, NG, by minimizing the BIC. The BIC as a function of NG is shown



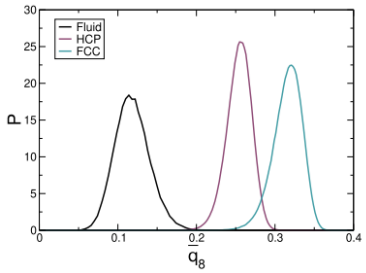
and has a minimum for NG = 4. Then, the optimal number of clusters, K, is found by successively merging a pair of components and looking for the existence of an elbow in the clustering entropy as a function of K.

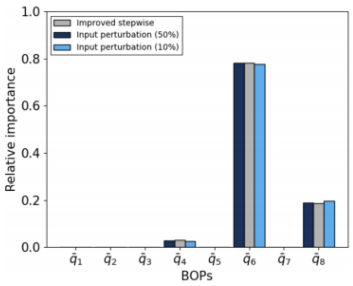
The elbow is detected at K = 3, meaning that the unsupervised learning identifies three relevant environments. Note that we know beforehand the three phases present in the system (FCC, HCP, and fluid) so that we can easily associate each cluster with the correct phase. An idea of the partitioning of space performed by the clustering is given, where the color of each point is determined by the cluster with the highest membership probability.

Learning from the autoencoder

As discussed, several methods to assess the relative importance of input variables in neural network models are available. Here, we employ the input perturbation and the improved stepwise methods in order to understand which BOPs were considered to be the most relevant by the autoencoder for the system in. The relative importance of the BOPs evaluated with these methods is shown. Only a small subset of three BOPs is found to be relevant and, as expected, $q^{-}6$ (RI $\sim$ 78%) and $q^{-}4$ (RI $\sim$ 3%) are part of it. Interestingly however, $q^{-}8$, which to our knowledge has never been used in the literature, appears to be more important (RI $\sim$ 19%) than $q^{-}4$. To understand why this is, we evaluated the $q^{-}8$ distributions in the FCC, HCP, and fluid hard-sphere phases from several snapshots. Such distributions are very similar to those obtained for $q^{-}6$, in the sense that they show a clear separation between the fluid and the crystal phases and only a small overlap between the two crystals.

If we now go back to the snapshot shown, which is roughly half fluid and half crystal, then it is easy to understand why q^-4 has a lower relative importance than q^-6 and q^-8.





CONCLUSIONS:

In summary, we have introduced a simple, fast, and easy to implement unsupervised learning algorithm for recognizing local structural motifs in colloidal systems. This algorithm makes use of standard BOPs to describe local environments, an autoencoder for dimensionality reduction, and GMMs for clustering the results. We have applied it to a wide variety of systems, ranging from simple isotopically interacting systems (hard spheres, WCA particles) to binary mixtures, and even anisotropic hard cubes. In all cases, the algorithm performed very well, and we were able to identify local environments to a similar precision as “standard”—manually tuned and system-specific—order parameters. Moreover, we exploited the analytical mapping defined by the autoencoder to extract extra information on the systems analyzed. Specifically, in all cases, we could identify the relevant symmetries underlying the main differences among the distinct particle environments found in the system. Interestingly, in the binary system we studied, this analysis revealed that the same species BOPs were the most important for distinguishing the different particle environments. Finally, in the last example on nucleation and crystal growth, we also explored the possible difficulties one can encounter when only a very small fraction of particles is in one specific environment, and we showed how including more snapshots, which covered a more balanced distribution of local environments, could benefit the results of the analysis in such cases.

REFERENCES

1 S. Auer and D. Frenkel, Nature 409, 1020 (2001).

2U. Gasser, E. R. Weeks, A. Schofield, P. N. Pusey, and D. A. Weitz, Science 292,

3M. Hermes, E. C. M. Vermolen, M. E. Leunissen, D. L. J. Vossen, P. D. J. van Oostrum, M. Dijkstra, and A. van Blaaderen, Soft Matter 7, 4623 (2011).

4 E. Sanz, C. Valeriani, T. Vissers, A. Fortini, M. E. Leunissen, A. Van Blaaderen, D. Frenkel, and M. Dijkstra, J. Phys.: Condens. Matter 20, 494247 (2008).