CS/ECE/ISyE 524 — Introduction to Optimization — Spring 2020

Find my Gaussian: a Classifier motivated by Crystal Structure Identification

Joe Gorka (jgorka@wisc.edu) & Aditya Singh (asingh76@wisc.edu)

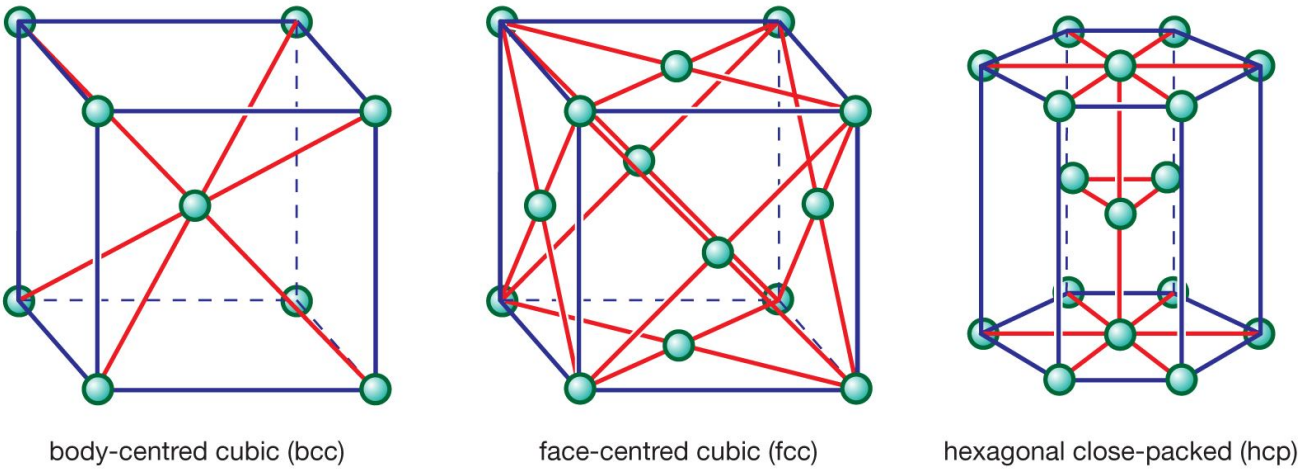
1. Introduction

1.1 Crystal Structures

Crystals are solid materials made up of atoms, molecules or ions that possess a highly ordered geometry. This geometric structure arises as a result of the physical/chemical properties of the crystal material. Some common examples of crystals include snowflakes, table salt and diamonds.

It's important to note that the geometric structure isn't uniform for all crystals - the orientations of the constituent particles depend strongly on their physical interactions with neighboring particles ; as such, the particles can form a multitude of different crystal structures based on the nature of their interactions.

Most common crystals that we encounter in our day to day lives usually have the structures of body-centered cubic (BCC), face-centered cubic (FCC) or hexagonal-closed packed (HCP). The diagram below shows the structuring of these three different types of crystals:



Motivated by the applications of crystals in synthesis and development of nanoparticles and biomaterials, the field of condensed matter physics employes computational modeling and simulations to understand the physical interactions that govern the different forms of crystal structures.

1.2 Molecular Simulations

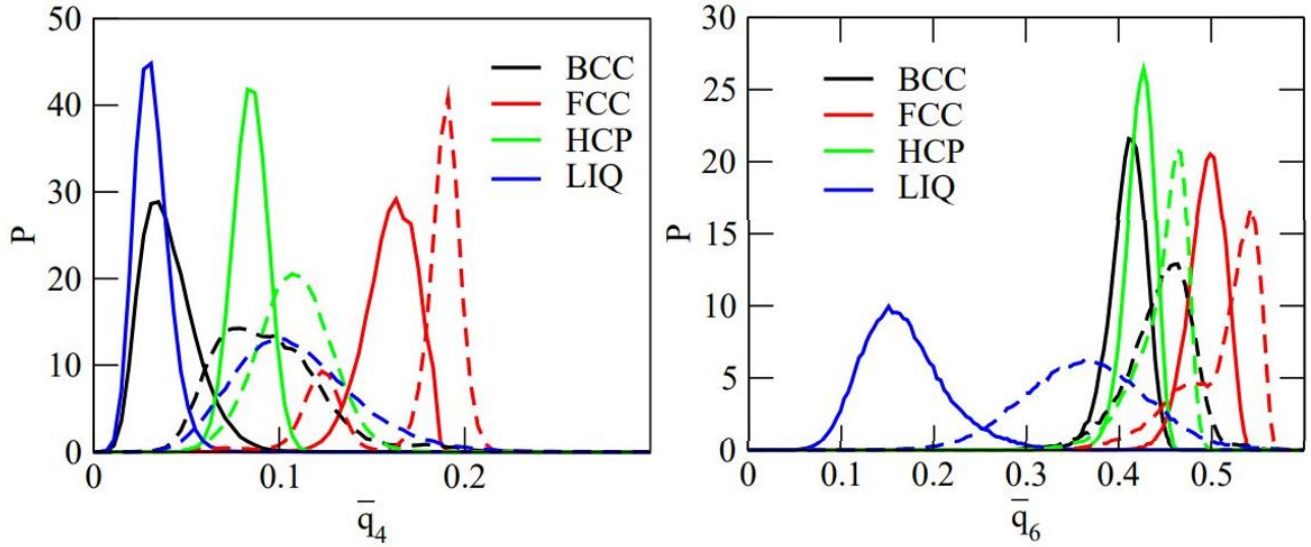
The use of computational simulations to discover and understand physical phenomenon is an integral part of the fields of physics, chemistry and biochemistry. The methodologies used to perform molecular level simulations make use of Markov-Chain Monte Carlo, dynamic integration of Newtonian equations or a combination of both.

Since the early 60s, condensed matter physicists have been using these methods to study different types of crystal structures. However, there exists a major problem in performing computational simulations on crystals - the problem of fluctuations and metastability. One can think of running simulations as microscopically observing a crystal for a short period of time. Since what we are observing is dynamic in nature, there is no guarantee that it will retain the same state that we started with. In other words, if we are running a simulation of atoms that form FCC crystals for instance, there's a good chance that some atoms will change their orientation and form other crystal structures. This makes it hard to analyze the physical and chemical properties of a specific crystal structure.

1.3 Order Parameters

Steinhardt et al. (1983) tried solve this predicament by assigning certain "order paramteters" to each atom in the simulation. An order parameter can be thought of as a function of the positions of an atom relative to its neighbors. In this case, the order parameters q_4 and q_6 developed by Stenhardt et al. were used to classiy the crystal structure formed by the atoms. Although this parameter was useful in distinguishing between liquid and other crystal structures, it proved to be insufficient in distinguishing between other crystal structures.

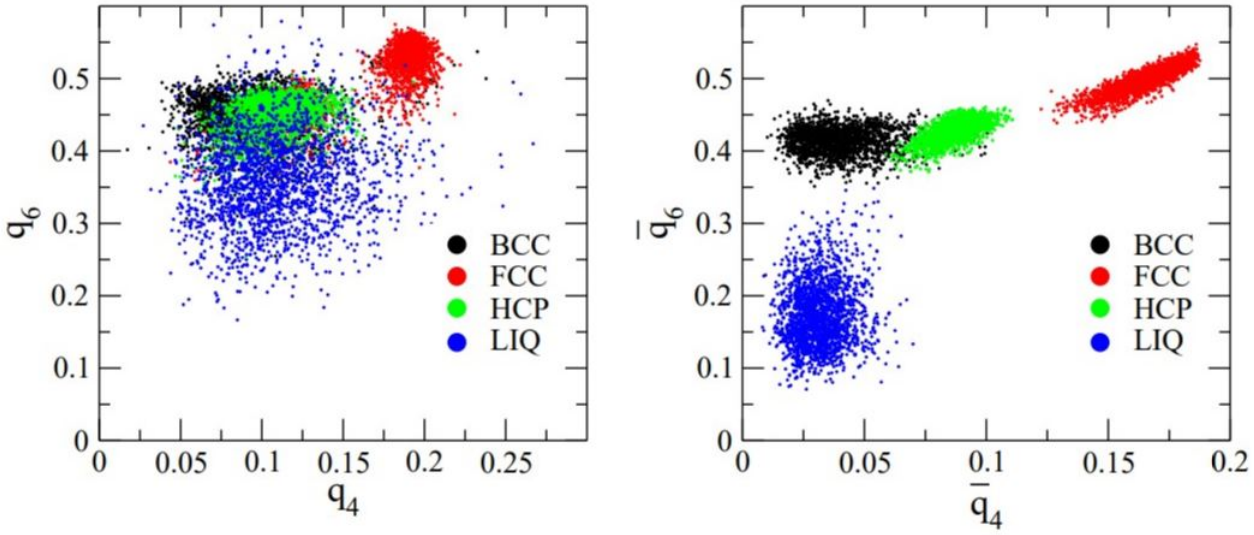
Lechner & Dellago (2008) further developed two more order parameters, $\bar{q}_4$ and $\bar{q}_6$ that performed considerably better in distinguishing different crystal structures (as seen in the figure below).



Left: Probability distributions of $\bar{q}_4$ (solid lines) and q_4 (dashed lines) for the FCC, BCC and HCP crystals and for the undercooled liquid (LIQ) in the Lennard-Jones system. Right: Probability distributions of $\bar{q}_6$ (solid lines) and q_6 (dashed lines) for the same phases.

Even with their new parameters, the distribution corresponding to the different structures show a strong overlap. For instance, one can see in the left image that the distribution of the order parameter $\bar{q}_4$ for liquid and BCC have almost a 90% overlap. Similarly, in the image in the right, the distribution of the order parameter $\bar{q}_6$ for HCP and BCC have more than 90% overlap. This suggests that we need more than 1 order paramtemer to accurately classify the atoms.

This problem exists when two order parameters are considered as well; If we observe the distributions using a combination of any two of the four order parameters, it is still not trivial to accuractely classify the crystal structure (as seen in the image below).



Comparison between the q_4-q_6-plane (left) and the $\bar{q}_4$-$\bar{q}_6$-plane (right) for the Lennard-Jones system in three different crystalline structures and in the liquid phase. Each point corresponds to a particular particle, where 2000 points from each structure were chosen randomly.

This problem is the motivation of our project - if we strictly use either one, or some choice of two order parameters, we will not be able to obtain the maximum possible accuracy in identifying crystal structures. Hence we have developed weighting system that assigns certain weight to all 4 of the order parameters and finds probabilities based of these weights. The weights are obtained by performing a Mixed Integer Linear Programming Optimization.

2. Methods

2.1 Data

Molecular Dynamics Simulation were performed for each 4 different types of structures - BCC, FCC, HCP and liquid. The simulations were performed at favorable conditions to ensure that all the atoms retained their starting structures. Each generated set contained 1500 atoms belonging to different structures, giving us a total of 6000 atoms with values for q_6, q_4, $\bar{q}_4$ and $\bar{q}_6$. The final training file contained the 4 order parameters and the accurate label for all 6000 atoms.

In addition, 24,000 additional atoms were generated using the same method, 1000 of which were selected randomly to function as a test set.

2.2 Gaussian Distributions

The important aspect of this classification problem is the fact that the 4 order parameters follow a gaussian distribution for different crystal structures. This is clearly evident in Fig. 2 as well as Fig. 3, in which we can see the distribution of the order parameters corresponding to different structures. The mean and standard deviation values are given by Lechner & Dellago (2008) for the 16 gaussians (gaussians formed by 4 crystal structures for all 4 different types of order parameters).

Once we obtain the order parameters for all atoms, we preprocess the data to calculate the probability of an atom belonging to a crystal structure according to an order parameter. We do this by:

1. For each atom, we calculate the z-value, which is $= \frac{x-\mu}{\sigma}$. Here x is the order parameter of interest and μ and σ are the mean and standard deviation for different crystal structures using that order parameter. This gives us a 4x4 matrix for each atom, where the rows correspond to the different crystal structure. For instance the row corresponding to BCC looks like: ## $\left(\frac{|q_4 - \mu_{q4}^{bcc}|}{\sigma_{q4}^{bcc}} \quad \frac{|\bar{q}_4 - \mu_{\bar{q}4}^{bcc}|}{\sigma_{\bar{q}4}^{bcc}} \quad \frac{|q_6 - \mu_{q6}^{bcc}|}{\sigma_{q6}^{bcc}} \quad \frac{|\bar{q}_6 - \mu_{\bar{q}6}^{bcc}|}{\sigma_{\bar{q}6}^{bcc}} \right)$

1. Once we get this matrix we do a probability calculation based of the z-value. Since statistics dictates that the probability of a discrete point in a continuous distribution is 0, we calculate the probability of a point lying outside the z-value. In other words, if the z-value is 1, this means that the point is 1 standard deviation away from the mean - as such, the probability of observing a point 1 standard deviation away from the mean is 0.3173 (or 0.15865 due to symmetry). More rigorously we can compute this by calculating the interal: ## $P(z \geq Z) = 1 - \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{\frac{-u^2}{2}} du$ Therefore the probability that a point lies outside the gaussian is used as the probability that the point belongs to that gaussian. We construct a 4x4 matrix similar to the one above, which now stores the probability instead of the z-value.
2. We normalize the data for each order parameter. Basically this means that for all atoms x & order parameters q, the probabilities are normalized such that $P(x = BCC|q) + P(x = FCC|q) + P(x = HCP|q) + P(x = LIQ|q) = 1$

2.3 MILP Optimization

Once we get the Nx4x4 matrix, where N = number of atoms, we finally frame the problem as a Mixed-Integer Linear Programming Optimization. Our classification for atom x is as follows:

Class = $Max_{c=BCC,FCC,HCP,LIQ} \ (w_1 P(x = c|q_4) + w_2 P(x = c|\bar{q}_4) + w_3 P(x = c|q_6) + w_4 P(x = c|\bar{q}_6)$

As mentioned before, we try to find the weights that give us the best accuracy in classifications us MILP. We use two binary matrices, one of size Nx3 and another of size Nx1. The first binary matrix is used to as a workaround to check if the final probability of the correct structure is the maximum. This is done by calculating the differences between the final probability between correct structure and the other three structures for each atom. This gives a vector $d \in R^3$. Now the 3 binary elements are used to check if these differences are positive. That is, $diff_{xi} = 1 \iff d_i \geq 0$, where $i \in \{1, 2, 3\}$ and $x = 1...N$

Finally, we know that the point is correctly classified if and only if the final probability of the correct structure is greater than the final probability of all the final structures.

Decision variables :

- $w \in R^4$ (weights of each order parameter)
- $accuracy \in Z^N$ (Binary variable; it is = 1 iff the atom is classified correctly, else it is 0)
- $diff \in Z^{Nx3}$ (Binary variable; it is = 1 iff the difference between the final probability between the correct structure and the other 3 structure is ≥ 0)

Constraints :

- $(P_n w)_{\hat{i}} - (P_n w)_i \leq M(diff_{nz})$, where P is the Nx4x4 probability matrix and $n = 1..N$, $\hat{i}$ is the correct class, $i = \{FCC, BCC, HCP, LIQ\}$ where $i \neq \hat{i}$, $z = \{1, 2, 3\}$ and M is the upper bound
- $(P_n w)_{\hat{i}} - (P_n w)_i \geq m(1 - diff_{nz}) + diff_{nz}$, here all the parameters are similar to the one above and m is the lower bound. These two constraints are used to formulate the iff statement.
- $\sum_{i=1}^{3} diff_{ni} - 3 \geq m(1 - accuracy_n)$ for $n = 1, 2...N$
- $\sum_{i=1}^{3} diff_{ni} - 3 \leq M(accuracy_n) - (1 - accuracy_n)$ for $n = 1, 2...N$ These two constraints ensure that the accuracy of the atom is 1 (i.e.) the atom is correctly classified if and only if the difference between the correct structure and all other 3 structures is ≥ 0

Objective:

$$Max_{accuracy,w,diff} \ \sum_{n=1}^{N} accuracy_n$$

3. Solution

We implement the Mixed Integer Linear Problem discussed above in the code below. The probability calculations are done in the function calcPriors(dataset). Once the calculations are done, the weights and the training set accuracy is displayed in the cell below it. CPLEX Optimizer is used since it offers the best performance on problems involving a large number of decision variables.

In [10]:
```
using LinearAlgebra, JuMP, PyPlot, CPLEX, CSV, Statistics, SpecialFunctions, Printf,StatsBase
```

In [132]:
```
#Mean and standard deviation of q4, q4bar, q6, q6bar (columns) for BCC, FCC, HCP & LIQ structures (rows)
mu = [0.089988 0.033406 0.440526 0.408018;
0.170880 0.158180 0.507298 0.491385;
0.107923 0.084052 0.445384 0.421813;
0.109049 0.031246 0.360012 0.161962]

std = [0.026831 0.010782 0.034791 0.020516;
0.032787 0.014346 0.043301 0.020566;
0.019476 0.009434 0.028992 0.015965;
0.031992 0.008786 0.066518 0.039360];
```

In [198]:
```
#This cell reads in training dataset, w/ a single row atom = [q4,q4bar,q6,q6bar,True Crystal Structure]
#10000 atoms are chosen randomly from holdout set

N = 6000
test_N = 10000

dataset = CSV.read("training.csv");
holdset = CSV.read("allfiles.csv");
testset = holdset[sample(axes(holdset, 1), test_N; replace = false, ordered = true), :]

answervec = strip.(dataset[:,5]);
testanswervec = strip.(testset[:,5]);

dataset = convert(Matrix,dataset[:,1:4]);
testset = convert(Matrix,testset[:,1:4]);

numvec = zeros(Int8, N);
testnumvec = zeros(Int8,test_N)
#create answer vector of integers for training set
for i in 1:size(numvec)[1]
    if (answervec[i]=="bcc")
        numvec[i] = 1
    end
    if(answervec[i]=="fcc")
        numvec[i] = 2
    end
    if(answervec[i]=="hcp")
        numvec[i] = 3
    end
    if(answervec[i]=="liq")
        numvec[i] = 4
    end
end
#do the same for the test set
for i in 1:test_N
    if (testanswervec[i]=="bcc")
        testnumvec[i] = 1
    end
    if(testanswervec[i]=="fcc")
        testnumvec[i] = 2
    end
    if(testanswervec[i]=="hcp")
        testnumvec[i] = 3
    end
    if(testanswervec[i]=="liq")
        testnumvec[i] = 4
    end
end
```

In [173]:
```
function calcPriors(dataset)
    N = size(dataset)[1]
    prob = zeros(N,4,4)
    for i = 1:N
        for j = 1:4
            val = dataset[i,j]
            for k = 1:4
                z = abs(val - mu[k,j]) / (std[k,j])
                prob[i,k,j] = 0.5 * (1- erf(z/sqrt(2)) )
            end
        end
    end
    return prob
end
```

Out[173]:
```
calcPriors (generic function with 1 method)
```

In [210]:
```
upper = 10000;#upper bound for setting the binary constraint
lower = -upper;#lower bound for setting the binary constraint
epsilon = 0.0001;#epsilon for float type

prob = calcPriors(dataset)

m = Model(CPLEX.Optimizer)
@variable(m, w[1:4] )
@variable(m, accuracy[1:N], Bin)
@variable(m, diff[1:N,1:3],Bin)
for i = 1:N
    prob_matrix = prob[i,:,:] * w
    label =  numvec[i]
    index = 1;
    for j = 1:4
        if(j != label)
            @constraint(m, prob_matrix[label] - prob_matrix[j] <= upper*diff[i,index])
            @constraint(m, prob_matrix[label] - prob_matrix[j] >= lower*(1-diff[i,index]) + epsilon*diff[i,index])
            index += 1
        end
    end
    @constraint(m, sum(diff[i,:]) - 3 >= lower*(1-accuracy[i]))
    @constraint(m, sum(diff[i,:]) - 3 <= upper*accuracy[i] - (1-accuracy[i]))
end
@objective(m, Max, sum(accuracy))
solve = optimize!(m)
print(m)
```

```
Excessive output truncated after 6525198 bytes.
```

In [218]:
```
params = ["q4", "q4_hat", "q6", "q6_hat"]
weights = zeros(4)
for i = 1:4
    weights[i] = value(w[i])
    @printf("Weight for %6s : %16.14f\n", params[i], weights[i])
end
println()
a = 0
for i = 1:N
    a += value(accuracy[i])
end
@printf("Accuracy percentage on training set: %5.2f", (a/N * 100))
```

```
Weight for     q4 : 0.00179573139982
Weight for q4_hat : 0.01076708947860
Weight for     q6 : 0.00239546670095
Weight for q6_hat : 0.00653748041155

Accuracy percentage on training set: 100.00
```

The weights (although mentioned above) are summarized below:

$w_{q_4} = 0.00179573139982$

$w_{\bar{q}_4} = 0.01076708947860$

$w_{q_6} = 0.00239546670095$

$w_{\bar{q}_6} = 0.00653748041155$

4. Results & Discussion

4.1 Test Dataset

Using these weights, all the 6000 datapoints in the training set were classified correctly. However it is important to note that these weights could be overfitted to the training set (which seems likely as 100% accuracy is achieved). In order to ensure that overfitting is not an issue, we further classify 10,000 datapoints that were not in the training dataset below.

The function classify(testset, weightVec) uses the classifies the datapoints using the equation:

$Max_{c=BCC,FCC,HCP,LIQ}\ (w_1 P(x=c|q_4) + w_2 P(x=c|\bar{q}_4) + w_3 P(x=c|q_6) + w_4 P(x=c|\bar{q}_6)$, where the weights are calculated from the MILP optimization performbed above.

In [219]:
```
function classify(dataset,weightVec)
    probs = calcPriors(dataset)#grab 3d array of priors
    weightedProbs = [probs[i,:,:]*weightVec for i in 1:size(probs)[1]]
    return [x[2] for x in findmax.(weightedProbs)]#returns predicted structure (by integer index)

end
```

Out[219]:
```
classify (generic function with 1 method)
```

In [220]:
```
#check how many correct
pred = classify(testset,weights)
M = size(testset)[1]
correct = 0;
for i in 1:M
    if(pred[i]==testnumvec[i])
        correct = correct+1
    end
end
@printf("Accuracy percentage on test set: %5.2f", (correct/M * 100))
```

```
Accuracy percentage on test set: 93.20
```

The test set accuracy is found to be 93.20%, which indicates that the classification model did not suffer from overfitting to the training set. Such a high accuracy in crystal structure identification is a signifcant feat, suggesting that both our classification model, as well as the numerical weights perform excellently when it comes to classifying data to an underlying distribution.

4.2 Weights & Interpretability

One could reasonably argue that employing traditional classification methods such as neural netorks or support vector machines would, in the least, perform better than our model. We would like to argue that one of the motivation of our project was to identify which order parameters perform the best in classifying data. The weights obtained using MILP optimzation directly convey the importance of a specific order parameter in classifying data. This offers interpretibility to a degree that cannot be trivially obtained using traditional machine learning models.

Our weights also clearly demonstrate that the order parameters designed by Lechner & Dellage (2008), especially $\bar{q}_4$ performs considerably better than the one developed by Steinhardt et al. (1983). Obtaining this information based on traditional machine learning techniques requires dimensional reduction techniques such as L1 regularization or feature elimination that can further complicate the classification.

In conclusion, we successfully classify a dataset to its underlying gaussian distribution using a model that employs weights to estimate the imporance of its different features.

5. Future Work

This work has multiple implications, both in terms of the application to crystal structure identification as well its application to other classifications problems that require mapping of dataset to the an underlying distribution.

In terms of crystal structure identification, one can perform the same classification model on pairs of crystal structures instead of all at the same time, and then classify them accordingly. This would provide a better classification accuracy (since there are only two classes present). Moreover, in reality, a simulation does not usually contain more than 3 crystal structures, so such a calculation could considerably increase accuracy.

Finally, a classification model that weights parameters to map dataset to an underlying distribution is extremely useful due to its interpretability (as discussed before). This classification can easily be extended to other distribution functions such as Exponential, Poisson and Binomial Distributions, which would trivial simple since it would only involve chainging the calcPriors() function to a function that calculates the probability based on the given model.

6. References

Steinhardt, P. J., Nelson, D. R., & Ronchetti, M. (1983). Bond-orientational order in liquids and glasses. Physical Review B, 28(2), 784.

Lechner, W., & Dellago, C. (2008). Accurate determination of crystal structures based on averaged local bond order parameters. The Journal of chemical physics, 129(11), 114707.