IL - SVM - report

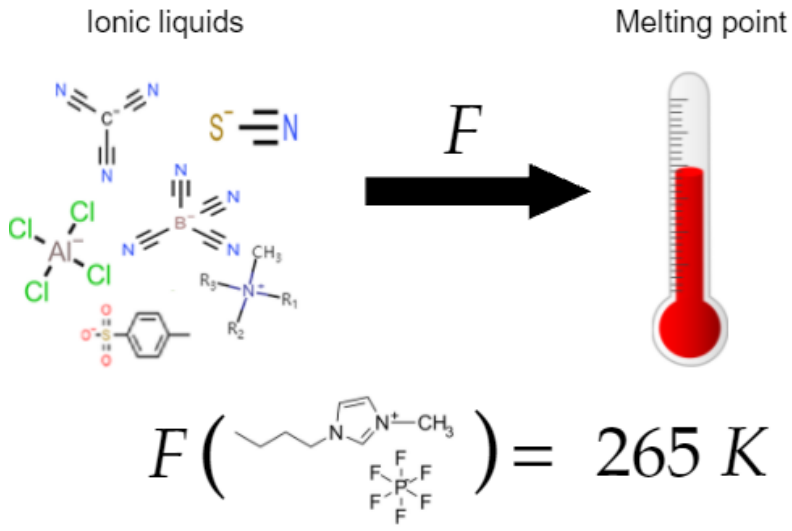
October 7, 2021

1 Introduction

Ionic liquids have a wide range of potential applications, for example as heat transfer and storage media or for carbon dioxide capture and purification of natural gas. But,

- the chemical structure is not yet fully understood,
- there are many different groups and types of ionic liquids,
- experimental studies are time-consuming and expensive to measure properties of all ionic liquids.

To assist the researcher in finding a suitable ionic liquid for the application, we attempt to build a predictive model $\boldsymbol{F}$ for the melting point of ionic liquids.



2 Theoretical background

Based on the ideas of Kai Wang et al. in [3] we use a support vector machine (SVM) to build the model $\boldsymbol{F}$. A SVM is a supervised machine learning algorithm that analyze data for classification and regression analysis. In the remaining part of this section, we follow the presentation of SVM in ([2], Chapter 12).

Let $\{\mathbf{x}_i, y_i\}_{i=1}^N$ be the training set, $\mathbf{x}_i \in \mathbb{R}^n$ and $y \in \mathbb{R}$. Suppose we consider approximation of the regression function in terms of a set of basis functions $\{h_m\}_{m=1}^M$, that is,

$$\boldsymbol{F}(x) = \sum_{m=1}^{M} \beta_m h_m(x) + \beta_0, \quad \text{for some } \beta_0, \beta_1, \ldots, \beta_M \in \mathbb{R}.$$

Note that the simple case of linear regression corresponds to the following situation $M = 1$ and $h_1(x) = h(x) = x$.

To estimate $\beta_0, \boldsymbol{\beta} = \beta_1, \ldots, \beta_M$ we minimize

$$H(\boldsymbol{\beta}, \beta_0) = \sum_{i=1}^{N} V(y_i - \boldsymbol{F}(x_i)) + \frac{\lambda}{2} \sum_{m=1}^{M} \beta_m^2, \tag{1}$$

where λ is a *cost* parameter and $V(r)$ is a error measure. Two common choices for V are

$$V_\epsilon(r) = \begin{cases} 0, & \text{if } |r| < \epsilon, \\ |r| - \epsilon, & \text{otherwise }, \end{cases}$$

and (due to Huber)

$$V_{H,c}(r) = \begin{cases} r^2/2, & \text{if } |r| < c, \\ c|r| - c^2/2, & \text{otherwise }. \end{cases}$$

However, for any choice of $V(r)$, the solution $\hat{\boldsymbol{F}}(x) = \sum \hat{\beta}_m h(x)_m + \hat{\beta}_0$ has the form

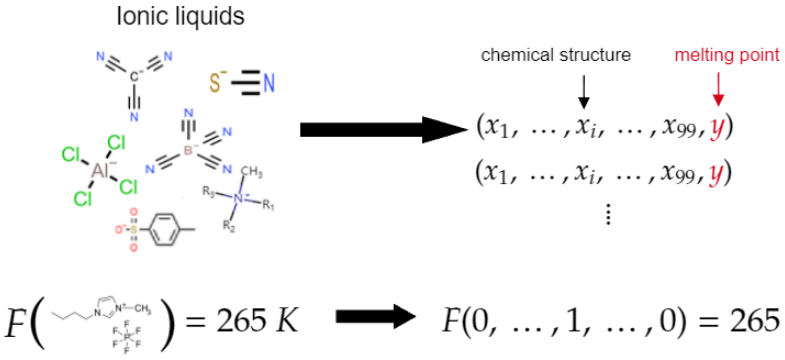
$$\hat{\boldsymbol{F}}(x) = \sum_{i=1} \alpha_i K(x, \mathbf{x}_i) + b$$

with $K(x, y) = \sum h_m(x) h_m(y)$. We call $K(x, y)$ the Kernel function and the vectors $\mathbf{x}_i$ for which the coefficient α_i is nonzero are called support vectors.
To find the coefficients α_i we have to solve a certain minimization problem that results from (1). There are different variations of this minimization problem leading to different variations of the SVM. We refer to ([1], Section 2.4 and 2.5) for more information. In practice we do not choose the functions h_m but the kernel function $K(x, y)$ and a variation of the minimization problem for the coefficients α_i. The following functions are common choices for the kernel functions

polynomial:	$K(u,v) = (\gamma\langle u,v\rangle + c)^d$
radial basis:	$K(u,v) = \exp(-\gamma\lvert u-v\rvert^2)$
sigmoid:	$K(u,v) = \tanh(\gamma\langle u,v\rangle + c)$

3 The Data

We use the same dataset as Kai Wang et al. in [3] consisting of 768 ionic liquids and their melting points. The chemical structure of ionic liquids is "encoded" into a vector form to obtain a mathematical formulation. Each variable corresponds to a particular chemical constituent that an ionic liquid may have. The value of this variable indicates how many of these components the ionic liquid has. It should be noted that this "coding" only counts the number of chemical constituents and does not take into account any chemical properties of the constituents or their possible interactions with each other.



Since the format of the variables $x_{i,j}$ and the responses are of different types (discrete and continuous, respectively) and on a different scale, we need to normalize the data set. We have chosen the following normalization method

$$\frac{x_{i,j} - \min_j x_{i,j}}{\max_j x_{i,j} - \min_j x_{i,j}}, \quad \text{for } i = 1,\ldots,N,\ j = 1,\ldots,n.$$

Here $\min_j x_{i,j}$ and $\max_j x_{i,j}$ denotes the minimum and maximum, respectively, over all entries in the dataset for the variable j.

4 Computation

To effectively perform the SVM algorithm we use the *LIBSVM* package in Python, see [1] or click me.

4.1 Prepare the data

In order to use this package, we need to put the data into the correct format in Python. We have to separate the data into two lists, one for the variables and one for the responses. The list of responses simply contains all the responses in the correct order $[y_1, y_2, \ldots, y_N]$. The list for the data has the following form

$$[\{1 : x_{1,1},\ 2 : x_{1,2}, \ldots, n : x_{1,n}\}, \{1 : x_{2,1}, \ldots, n : x_{2,n}\}, \ldots, \{1 : x_{N,1}, \ldots, n : x_{N,n}\}]$$

where $x_{i,j}$ is the value of the $j-$th variable of the $i-$th observation $\mathbf{x}_i \in \mathbb{R}^n$.

The *LIBSVM* package contains the function *svm_read_problem()* that can read a txt-file and bring the data in the right format. To use this function one simply has to give the path to the txt-file as input.

Listing 1: *Prepare the data*

```
y_training , x_training = svm_read_problem('Path/training_set.txt')
y_test , x_test = svm_read_problem('Path/test_set.txt')
```

However, the data in the txt-file must also be in the correct format. Each row line corresponds to an observation from the dataset and has to be written as follows

$$\begin{array}{cccccc} y_1 & 1 : x_{1,1} & 2 : x_{1,2} & \ldots & n : x_{1,n} \\ y_2 & 1 : x_{2,1} & 2 : x_{2,2} & \ldots & n : x_{2,n} \\ & & \vdots & & \\ y_N & 1 : x_{N,1} & 2 : x_{N,2} & \ldots & n : x_{N,n} \end{array}$$

The following R code writes a dataset into a txt-file with the correct format.

Listing 2: *Prepare the data*

```
# N = number of observations
# n = number of variables
# dataset = the dataset (e.g. the training set)

txt_vec = 1:N
for (i in 1:N) {
  txt_vec_temp = 1:100
  txt_vec_temp[1]=paste(dataset$response[i])
  for (j in 2:n) {
    j_temp = j-1
    txt_vec_temp[j]=paste(j_temp,as.matrix(dataset)[i,j_temp],sep = ":")
  }
  txt_vec[i]=paste(txt_vec_test_temp,collapse = " ")
}

conn<-file("dataset_file_name.txt")
writeLines(txt_vec, conn)
close(conn)
```

4.2 Train the model

As soon as we put the data we can train the model. To do so we use the function *svm_train*. The first two arguments of this function are the lists with the response and the variables respectively. In the third argument we select the options of the algorithm.

Listing 3: *Train the model*

```
model = svm_train(y_training, x_training, 'options')
```

We do not explain all possible options, instead we explain the options we have chosen for building the predictive model for the melting point of the ionic liquids and refer to "README_LIBSVM.pdf" (page 3, 'svm-train' Usage) for more information. For our model we have chosen the following options

$$\text{'options'} = \text{'}-\text{s } 4 \ -\text{t } 2 \ -\text{g } 0.0272 \ -\text{c } 65\text{'}$$

The fist argument in the options $-$s 4 specifies which the type of SVM we are training, in this case it is $\nu-$ Support Vector Regression, other options would be $\epsilon-$ Support Vector Regression or different variations of classification SVMs (originally SVM was developed for classification problems).
In the second argument $-$t 2 we choose the kernel $K(u, v)$ of the SVM. We have chosen a radial basis function $K(u, v) = e^{-\gamma|u-v|^2}$.
The parameter γ in the definition of the kernel function $K(u, v)$ is chosen with the third option $-$g 0.0272.
Finally, the option $-$c 65 chooses the cost-parameter in the minimization problem mentioned in the section about the theoretical background.
In the output one can find information about the solution of the minimization problem (number of iterations of the algorithm, some parameters) and the number of support vectors. If one wants to access more information, e.g. a list of all support vectors, the model can be saved with the function *svm_save_model()*. The first input of the function is the name of the file where one wants to save the model, and the second argument is the previously trained model.

Listing 4: *Save the model*

```
svm_save_model('svm_model_file_name',model)
```

4.3 Test the model

In the next step we use the test set to test the performance of the model. To do so we use the function *svm_predict()*. The first two arguments are the lists with the responses and variables, respectively. In the third argument we specify the model we want to use to make the predictions of the test set.

Listing 5: *Test the model*

```
p_labs, p_acc, p_vals = svm_predict(y_test, x_test, model)
```

	Training set	Test set	Validation set
MSE	386.85	515.47	6036.95
R^2	0.87	0.825	0.213

Note that one can also use this function for general predictions. That is, if one wants to use the model to make predictions for data for which we have no responses, in this case use [ ] in the first argument.
Regarding the output: p_labels is a list of the predicted values for responses, p_acc is a tuple including mean squared error (MSE) and squared correlation coefficient R^2, we refer to ([1], Chapter 3) for the definitions. The third output p_vals is only interesting if one uses SVM for classification.
Note that the predictions need to be rescaled. This can be done either directly in Python or one can use the following code to create a txt.file with the predictions and then use another programming language (e.g. R).

Listing 6: *Save the predictions*

```
openfile = open('predictions_file_name.txt', 'w')
for item in p_labs:
    openfile.write(str(item) + "\n")
openfile.close()
```

-functions -train -parameters -¿ choice - output -predict/test -output -¿ formulas

5 Results

The evaluation of the predictions gives the following results.
Here the validation set consists of 41 observations obtained by the measurements from Caitlin Blum and her team. It is immediately noticeable that the model performs much worse with this validation set. The validation set contains ionic liquids with higher melting points (about 500) compared to the training and test set, and the model performs very poorly in predicting this melting point, resulting in a high error. We will discuss these problems in more detail in the next section.

In the project we want to use ionic liquids as phase change materials (PCM). For this application we search for ionic liquids with a melting point between $24^oC - 40^oC$ $(296 - 312K)$. With the model $\boldsymbol{F}$ we predicted the melting point of group of about 1500 ionic liquids that are suitable for the application. By examining the predictions, we were eventually able to reduce the size of the group to about 300 (20%).

6 Discussion

The next step would be to improve the model. However, since the project was stopped by the chemists and Caitlin is no longer working with us, we have also stopped the project and only give some ideas to improve the model.

We have seen that the model performs much worse on the validation set. We suspect that this is because the validation set is very different from the training set and test set. In contrast, the training set and the test set are more similar. For example, the validation contains in percentage ionic liquids with higher melting point: 22% above $400K$ and 9.8% above 500% in the validation set and 7.3% above $400K$ and 0.22% above 500% in the training set and test set together, respectively. Moreover, the errors in the prediction for high melting points are large. We conclude that the model would be more flexible if we had a more heterogeneous training set.
Another way to improve the model would be to choose other parameters γ and C or even another kernel function $K(u, v)$. In [3] the proposed the following parameters $\gamma = 0.044$ and $C = 5.66$. We obtain better results with the parameter mentioned earlier: $R^2 = 0.781747$ for the training set and $R^2 = 0.801475$, respectively, with the parameters proposed in [1]. However, possible there are better choices for the parameters γ and C. To find a better choice for the parameters one could use Cross-validation and Grid-search, but again, since the project is over, we decided to not use this methods. We refer to "guide_LIBSVM.pdf" Section 3.2 for more information.

References

[1] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: a library for support vector machines". In: *ACM transactions on intelligent systems and technology (TIST)* 2.3 (2011), pp. 1–27.

[2] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York, 2001.

[3] Kai Wang et al. "Machine learning-based ionic liquids design and process simulation for CO2 separation from flue gas". In: *Green Energy & Environment* 6.3 (2021), pp. 432–443.