CosolventGPR_HittorfProcessing

February 23, 2022

1 Processing ternary compositions from all Hittorf polarization experiments

This workbook uses the trained Gaussian processes regression model to predict ternary compositions, given the sets of measured electriolyte physicochemical properties from Hittorf polarization experiments, as summarised in the *data* folder of this repository.

[2]:
```
# import packages
import os, sys, platform
import numpy as np
import pandas as pd
import scipy as sp
import GPy
import matplotlib.pyplot as plt
from matplotlib import gridspec
import warnings
warnings.filterwarnings('ignore')
print('Python version', sys.version)
print('Running on', platform.system())

# colours (From Birmingham With Love)
jade = np.array([0, .66, .436]) # statue green
blue = np.array([.057, .156, .520]) # hey there mr blue
brown = np.array([.515, .158, .033]) # did someone order CDM?
red = np.array([.85, .20, 0]) # tikka masala
gold = np.array([1, .67, .14]) # Staffordshire hoard
claret = np.array([.429, .073, .238]) # claret
grey = np.array([.585, .612, .675]) # library grey
black = np.array([0,0,0]) # this is a black
```

```
Python version 3.10.2 (v3.10.2:a58ebcc701, Jan 13 2022, 14:50:16) [Clang 13.0.0
(clang-1300.0.29.30)]
Running on Darwin
```

1.0.1 Defining functions

```
[3]: #General GPy function
     #=========================================
     def gpy_func(X, y):
         """
         Function performs the GP regression.
         The inputs X and y are the input and output pairs.
         """
         n = X.shape[0] # number of data points
         d = X.shape[1] # dimension of input

         # build kernel
         k_rbf = GPy.kern.RBF(input_dim=d,
                              ARD=True,
                              lengthscale=X.std(axis=0),
                              variance=y.var()/2)

         kernel = k_rbf

         # priors
         prior_uniform = GPy.priors.Uniform(0,1000)
         prior_gamma = GPy.priors.Gamma(a=1,b=1)

         # likelihood
         lik = GPy.likelihoods.Gaussian()

         # initially construct the model
         gpm = GPy.core.GP(    X=X,
                               Y=y,
                               likelihood=lik,
                               kernel=kernel)

         # set priors
         for j in range(X.shape[1]):
             gpm.kern.lengthscale[[j]].set_prior(prior_uniform, warning=False)
             gpm.kern.lengthscale[[j]] = X[:,j].std()*(2**.5) # start within the␣
      ↪prior

         gpm.kern.variance.set_prior(prior_gamma, warning=False)
         gpm.likelihood.variance.set_prior(prior_gamma, warning=False)

         # optimize the hyperparameters
         for i in range(0,20): # OPTIMIZE 10x,20x?
             gpm.optimize()

         return gpm
```

```
#Train GPM with physicochemical data
#==========================================
def load_GPM(training_file):
    """
    Function organises the physicochemical training data
    into input and outputs, and trains the gpm using the
    gpy_func()
    """

    #Training data organising
    F = pd.read_csv(training_file)

    df = F.sort_values(["xEC", "xLiPF6", "Temp"], ascending = (False, False,␣
↪True)) #Groups same composition properties together in ascending temperatures
    data = df.to_numpy()

    properties = data[:,5:]
    compositions = data[:,[2,4]] #Change to 2,4 for just LiPF6 and EMC␣
↪independent compositions, infer EC later

    properties = np.array(np.split(properties,np.arange(5,len(data),5)))␣
↪#Subarray every 5 temps
    properties = np.array([i.flatten() for i in properties])

    compositions = np.array(np.split(compositions,np.arange(5,len(data),5)))␣
↪#Subarray every 5 temps
    compositions = np.array([np.mean(i,0) for i in compositions])

    #GPM training
    Y = compositions #Model training outputs
    X = properties #Model training inputs

    n = X.shape[0] #number of rows of data
    d = X.shape[1] #dimensions of each data point

    gpm = gpy_func(X,Y)

    return gpm, compositions, properties
```

[4]:
```
#Process Hittorf results into composition predictions
#=====================================================
def process_hittorf(all_csvs, initial_composition, gpm):
    '''
    Takes Hittorf experiment csv raw data and runs it through
    the trained GPM model. Outputs initial and final (predicted)
    compositions on the anodic and cathodic sides.
    '''
    df_result = pd.DataFrame()
    for hittorf_csv in all_csvs:

        #Separate raw experimental data
        exp_df = pd.read_csv(hittorf_csv) # \todo synth_data directory change
        exp_df = exp_df[['d','v','k']]
        anodic_data = exp_df.iloc[0:5].to_numpy().flatten().reshape(1,15)␣
↪#Flatten first 5 rows
        cathodic_data = exp_df.iloc[5:].to_numpy().flatten().reshape(1,15)␣
↪#Flatten last 5 rows

        #Perform prediction
        x_anodic, var_anodic = gpm.predict(anodic_data) #PERFORM PREDICTIONS␣
↪***********
        x_anodic = np.append(x_anodic.flatten(),1-np.sum(x_anodic))
        sd_anodic = np.sqrt(var_anodic)

        x_cathodic, var_cathodic = gpm.predict(cathodic_data) #PERFORM␣
↪PREDICTIONS ***********
        x_cathodic = np.append(x_cathodic.flatten(),1-np.sum(x_cathodic))
        sd_cathodic = np.sqrt(var_cathodic)

        #Make result df
        df = pd.
↪DataFrame(columns=['side','filename','xLi','xEMC','xEC','stdev'],␣
↪index=['anodic','neutral','cathodic'])

        df.loc['anodic'] = pd.Series({'side': 'anodic',
                                      'filename':hittorf_csv,
                                      'xLi':x_anodic[0],
                                      'xEMC':x_anodic[1],
                                      'xEC':x_anodic[2],
                                      'stdev':sd_anodic[0][0]})

        df.loc['neutral'] = pd.Series({'side': 'neutral',
                                      'filename':hittorf_csv,
                                      'xLi':x_Li_neut,
                                      'xEMC':x_EMC_neut,
                                      'xEC':x_EC_neut,
                                      'stdev':0})

        df.loc['cathodic'] = pd.Series({'side': 'cathodic',
                                      'filename':hittorf_csv,
                                      'xLi':x_cathodic[0],
                                      'xEMC':x_cathodic[1],
                                      'xEC':x_cathodic[2],
                                        'stdev':sd_cathodic[0][0]})

        #Append to df_result

        df_result = df_result.append(df,ignore_index = False)

    return df_result


#Process transference numbers
#==========================================
def process_transference(df_result,all_csvs):
    '''
    Takes the resulting composition dataframe from process_hittorf(),
    along with raw hittorf density measurements to calculate transference
    numbers based on density alone and based on the GP inferred salt fractions
    '''
    df_result = pd.DataFrame()
    for hittorf_csv in all_csvs:

        #Hittorf Transference properties
        Q = 0.5*60*60*20/1000 #[C] charge passed during Hittorf Experiment
        V = 0.004 #L of Hittorf chamber
        F = 96485 #C/mol
        Me = 155.905 #g/mol
        Ve = (Me - 87.1)/(1159) #L/mol
        def c2d(c): #Density to molarity correlation
            return 1000*(0.0871*c+1.159) #g/L
        def d2c(d):
            return ((d/1000)-1.159)/0.0871 #mol/L

        i_df = df[df.filename==hittorf_csv] #Single experiment df
        exp_df = pd.read_csv(hittorf_csv)

        d_neut = np.mean(exp_df[exp_df.Temp == 25].d)*1000 #g/L
        d_ano = np.max(exp_df[exp_df.Temp == 25].d)*1000 #g/L
        d_cat = np.min(exp_df[exp_df.Temp == 25].d)*1000 #g/L

        c_neut = d2c(d_neut)
        c_ano = d2c(d_ano)
        c_cat = d2c(d_cat)
        dn_a = abs(V*(c_neut-c_ano))
        dn_c = abs(V*(c_neut-c_cat))

        # Transference number based on density changes alone
        tp_a_trad = 1-(dn_a*F)/(Q*(1-(c_neut*Ve)))
        tp_c_trad = 1-(dn_c*F)/(Q*(1-(c_neut*Ve)))
        tp_dens = np.mean([tp_a_trad,tp_c_trad])
        tp_dens = np.repeat(tp_dens,3)

        # Cosolvent Mass Frac Transference Calc - using the above cell data on␣
↪density etc
        x_neut = i_df[i_df.index == 'neutral'].xLi[0]
        x_ano = i_df[i_df.index == 'anodic'].xLi[0]
        x_cat = i_df[i_df.index == 'cathodic'].xLi[0]

        n_neut = (x_neut*V*d_neut)/Me #moles in the compartments [L]*[g/L]/[g/
↪mol]
        n_ano = (x_ano*V*d_ano)/Me #moles in the compartments [L]*[g/L]/[g/mol]
        n_cat = (x_cat*V*d_cat)/Me #moles in the compartments [L]*[g/L]/[g/mol]

        dn_a = abs(n_neut-n_ano)
        dn_c = abs(n_neut-n_cat)

        tp_a = 1-(dn_a*F)/(Q*(1-(c_neut*Ve)))
        tp_c = 1-(dn_c*F)/(Q*(1-(c_neut*Ve)))
        tp_frac = np.mean([tp_a,tp_c])

        tp_frac = np.repeat(tp_frac,3)

        i_df['tp_dens'] = tp_dens
        i_df['tp_frac'] = tp_frac

        df_result = df_result.append(i_df,ignore_index = False)

    return df_result
```

1.0.2 Initial 0.6 M LiPF6 in 1:1 EC:EMC solution

[5]:
```
# 55050 RESULTS
#====================================================================

#Training data
training_file = 'trainingset/Ternary_Physicochemical_Training.csv'

#Starting composition
x_Li_neut = 0.065 #mass frac LiPF6
x_EC_neut = 0.4675 #mass frac EC
x_EMC_neut = 1-x_Li_neut-x_EC_neut #mass frac EMC
initial_composition = [x_Li_neut, x_EMC_neut, x_EC_neut]

#Experiment filenames
```

```
folder = 'data/'
all_csvs = ['55050_1.csv','55050_2.csv','55050_3.csv']
all_csvs = [folder + csv for csv in all_csvs]

#Initiating gpm model
gpm, compositions, properties = load_GPM(training_file)

#Processing experimental data
df = process_hittorf(all_csvs, initial_composition, gpm)
df = process_transference(df,all_csvs)
df.to_csv('results/Result_Summary_55050.csv',index=False)
display(df)
```

```
              side         filename       xLi      xEMC       xEC     stdev  \
anodic      anodic  data/55050_1.csv  0.072163  0.467108  0.460729  0.002549
neutral    neutral  data/55050_1.csv     0.065    0.4675    0.4675         0
cathodic  cathodic  data/55050_1.csv  0.059126   0.47171  0.469164  0.002526
anodic      anodic  data/55050_2.csv  0.071462  0.467643  0.460895  0.002547
neutral    neutral  data/55050_2.csv     0.065    0.4675    0.4675         0
cathodic  cathodic  data/55050_2.csv   0.05997  0.470544  0.469486  0.002526
anodic      anodic  data/55050_3.csv  0.072122  0.467071  0.460806  0.002554
neutral    neutral  data/55050_3.csv     0.065    0.4675    0.4675         0
cathodic  cathodic  data/55050_3.csv  0.059185  0.471472  0.469343  0.002527

           tp_dens   tp_frac
anodic    0.408451  0.419290
neutral   0.408451  0.419290
cathodic  0.408451  0.419290
anodic    0.503809  0.488914
neutral   0.503809  0.488914
cathodic  0.503809  0.488914
anodic    0.414791  0.423728
neutral   0.414791  0.423728
cathodic  0.414791  0.423728
```

1.0.3 Initial 1.0 M LiPF6 in 1:1 EC:EMC solution

[6]:
```
# 55100 RESULTS
#=================================================================

#Training data
training_file = 'trainingset/Ternary_Physicochemical_Training.csv'

#Starting composition
x_Li_neut = 0.125 #mass frac LiPF6
x_EC_neut = 0.4375 #mass frac EC
x_EMC_neut = 1-x_Li_neut-x_EC_neut #mass frac EMC
```

```python
initial_composition = [x_Li_neut, x_EMC_neut, x_EC_neut]

#Experiment filenames
folder = 'data/'
all_csvs = ['55100_1.csv','55100_2.csv',
            '55100_3.csv','55100_4.csv']
all_csvs = [folder + csv for csv in all_csvs]

#Initiating gpm model
gpm, compositions, properties = load_GPM(training_file)

#Processing experimental data
df = process_hittorf(all_csvs, initial_composition, gpm)
df = process_transference(df,all_csvs)
df.to_csv('results/Result_Summary_55100.csv',index=False)
display(df)
```

```
              side          filename      xLi      xEMC       xEC     stdev  \
anodic      anodic  data/55100_1.csv  0.133092  0.44512  0.421788  0.002598
neutral    neutral  data/55100_1.csv     0.125   0.4375    0.4375         0
cathodic  cathodic  data/55100_1.csv  0.117866  0.433463  0.448671  0.002611
anodic      anodic  data/55100_2.csv   0.13289  0.444008  0.423103  0.002591
neutral    neutral  data/55100_2.csv     0.125   0.4375    0.4375         0
cathodic  cathodic  data/55100_2.csv  0.118003  0.434202  0.447795  0.002605
anodic      anodic  data/55100_3.csv  0.133582  0.447466  0.418951  0.002641
neutral    neutral  data/55100_3.csv     0.125   0.4375    0.4375         0
cathodic  cathodic  data/55100_3.csv  0.118228  0.433228  0.448544  0.002608
anodic      anodic  data/55100_4.csv  0.133127  0.443573  0.423301  0.002585
neutral    neutral  data/55100_4.csv     0.125   0.4375    0.4375         0
cathodic  cathodic  data/55100_4.csv  0.118068  0.434567  0.447365  0.002602

           tp_dens   tp_frac
anodic    0.723972  0.280565
neutral   0.723972  0.280565
cathodic  0.723972  0.280565
anodic    0.691100  0.293828
neutral   0.691100  0.293828
cathodic  0.691100  0.293828
anodic    0.783145  0.278903
neutral   0.783145  0.278903
cathodic  0.783145  0.278903
anodic    0.664787  0.283908
neutral   0.664787  0.283908
cathodic  0.664787  0.283908
```

1.0.4 Initial 1.6 M LiPF6 in 1:1 EC:EMC solution

[7]:
```
# 55150 RESULTS
#=================================================================

#Training data
training_file = 'trainingset/Ternary_Physicochemical_Training.csv'

#Starting composition
x_Li_neut = 0.185 #mass frac LiPF6
x_EC_neut = 0.4075 #mass frac EC
x_EMC_neut = 1-x_Li_neut-x_EC_neut #mass frac EMC
initial_composition = [x_Li_neut, x_EMC_neut, x_EC_neut]

#Experiment filenames
folder = 'data/'
all_csvs = ['55150_1.csv','55150_2.csv','55150_3.csv']
all_csvs = [folder + csv for csv in all_csvs]

#Initiating gpm model
gpm, compositions, properties = load_GPM(training_file)

#Processing experimental data
df = process_hittorf(all_csvs, initial_composition, gpm)
df = process_transference(df,all_csvs)
df.to_csv('results/Result_Summary_55150.csv',index=False)
display(df)
```

```
              side           filename       xLi      xEMC       xEC     stdev  \
anodic      anodic  data/55150_1.csv  0.194742  0.418107  0.387151  0.002896
neutral    neutral  data/55150_1.csv     0.185    0.4075    0.4075         0
cathodic  cathodic  data/55150_1.csv  0.177159  0.399471   0.42337  0.002933
anodic      anodic  data/55150_2.csv  0.194795  0.417088  0.388117  0.002843
neutral    neutral  data/55150_2.csv     0.185    0.4075    0.4075         0
cathodic  cathodic  data/55150_2.csv  0.177046  0.400354    0.4226    0.0029
anodic      anodic  data/55150_3.csv  0.195139  0.416577  0.388284  0.002855
neutral    neutral  data/55150_3.csv     0.185    0.4075    0.4075         0
cathodic  cathodic  data/55150_3.csv  0.176814  0.401151  0.422036   0.00284

          tp_dens   tp_frac
anodic    0.836769  0.115435
neutral   0.836769  0.115435
cathodic  0.836769  0.115435
anodic    0.775549  0.100854
neutral   0.775549  0.100854
cathodic  0.775549  0.100854
anodic    0.727948  0.067490
neutral   0.727948  0.067490
```

```
cathodic  0.727948  0.067490
```