

Цифровое моделирование физико-химических систем

Раздел 2. Свойства веществ

Лекция 2.1. Предсказание свойств индивидуальных веществ и смесей

Лектор: доцент кафедры
информационных компьютерных технологий, к.т.н.

Митричев Иван Игоревич

Москва

2024

Подходы к предсказанию свойств веществ и смесей

- Эмпирические (не имеют физической сути, Выражены математическими формулами) - корреляции
- Полуэмпирические (физические законы в виде формул с подобранными константами, параметрами – например, константами взаимодействия)
- Теоретические (системы математических уравнений, отражающих физические явления)
- ML-подход (обучение на данных), Есть эмпирические (регрессии), black-box (модель в явном виде не записывается)

Ab initio – «из первых принципов» - квантовохимические методы расчета структуры и свойств веществ

Свойства веществ и смесей

Физические: вязкость, плотность, температура плавления, температура кипения, диэлектрическая проницаемость, теплоёмкость, теплопроводность, электропроводность, показатель преломления и др.

Структурные: постоянные кристаллической решетки и углы, валентные углы, длины связей...

Физико-химические: энтальпия образования, энергия связи, дипольные моменты молекул...

Измеряются экспериментально.

1) Предсказание промежуточных значений (интерполяция) и значений вне диапазона измерений (экстраполяция)

2) Новое вещество: смеси, сплавы, новые соединения (например, сверхпроводники),

3) Дороговизна экспериментов

 **QSAR (в медицине)**

Основные классы методов для расчета свойств веществ и смесей

1. Корреляции

Вводится некоторое уравнение, которое хорошо описывает экспериментальные данные

Вязкость нефти (Lee)

$$\mu = 10^{-4} k_v \exp \left\{ x_v \left(\frac{\rho}{62.4} \right)^{y_v} \right\}$$
$$k_v = \frac{(9.4 + 0.02 MW) T^{1.5}}{209 + 19 MW + T}$$
$$x_v = 3.5 + \frac{986}{T} + 0.01 MW$$
$$y_v = 2.4 - 0.2 x_v$$

2, Правила смешения (для смесей)

Уравнение Аррениуса для вязкости жидкой смеси

$$\ln \eta_{lmix} = \sum_{i=1}^N x_i \ln \eta_{li}$$

Основные классы методов для расчета свойств веществ и смесей

3, Принцип соответствующих состояний (CSP)

Основан на наблюдении Ван-дер-Ваальса, что различные газы имеют одинаковый коэффициент сжимаемости и мольный объем (значение делят на критическое) при приведенных давлении и температуре, Так и вязкость вещества,

$$\eta_a(P_a, T_a) = \frac{\eta_{ca}}{\eta_{cz}} \cdot \eta_z(P_z, T_z)$$

где z – вещество с известной вязкостью при температуре $P_z = \frac{P_a P_{cz}}{P_{ca}}, T_z = \frac{T_a T_{cz}}{T_{ca}}$

4, Метод групповых вкладов (GC)

Рассчитывает свойства на основе структурных элементов молекулы

Метод Джобака (Joback)

Метод групповых вкладов для предсказания 11-и свойств веществ.

Нормальная температура кипения (н,т,к,)

$$T_b[\text{K}] = 198.2 + \sum T_{b,i}.$$

Температура замерзания

$$T_m[\text{K}] = 122.5 + \sum T_{m,i}.$$

Критическая температура

$$T_c[\text{K}] = T_b \left[0.584 + 0.965 \sum T_{c,i} - \left(\sum T_{c,i} \right)^2 \right]^{-1}.$$

Критическое давление

$$P_c[\text{bar}] = \left[0.113 + 0.0032 N_a - \sum P_{c,i} \right]^{-2},$$

Критический объем

$$V_c[\text{cm}^3/\text{mol}] = 17.5 + \sum V_{c,i}.$$

Теплота образования (298 К, идеальный газ) $H_{\text{formation}}[\text{kJ/mol}] = 68.29 + \sum H_{\text{form},i}.$

Метод Джобака (Joback)

Метод групповых вкладов для предсказания 11-и свойств веществ,

Энергия Гиббса образования

$$G_{\text{formation}} [\text{kJ/mol}] = 53.88 + \sum G_{\text{form},i}.$$

Теплоемкость

$$C_P [\text{J}/(\text{mol} \cdot \text{K})] = \sum a_i - 37.93 + \left[\sum b_i + 0.210 \right] T + \left[\sum c_i - 3.91 \cdot 10^{-4} \right] T^2 + \left[\sum d_i + 2.06 \cdot 10^{-7} \right] T^3.$$

Теплота испарения при н,т,к,

$$\Delta H_{\text{vap}} [\text{kJ/mol}] = 15.30 + \sum H_{\text{vap},i}.$$

Теплота плавления

$$\Delta H_{\text{fus}} [\text{kJ/mol}] = -0.88 + \sum H_{\text{fus},i}.$$

Динамическая вязкость жидкости

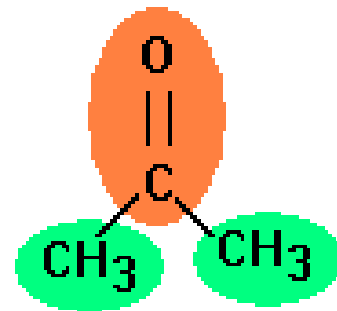
$$\eta_L [\text{Pa} \cdot \text{s}] = M_w \exp \left[\left(\sum \eta_a - 597.82 \right) / T + \sum \eta_b - 11.202 \right],$$

Область применимости: температура плавления -- $T_r < 0,7$

Метод Джобака (Joback)

Вклады групп: таблицы есть в Википедии по адресу
(https://en.wikipedia.org/wiki/Joback_method)

Пример: ацетон



	-CH ₃ (метил)		>C=O (кетон)					
Свойство	Число групп	Вклад	Число групп	Вклад	$\sum G_i$	Значение свойства	Эксперимент	Единицы
T_c	2	0,0141	1	0,0380	0,0662	500,559	508	К
P_c	2	-1,20e-3	1	3,10e-3	7,00e-4	48,025	48	бар

ML-модели

Machine learning (машинное обучение) – методы, которые используют обучение на данных для решения сходных задач.

Три основных вида:

1) обучение с учителем (supervised learning);

На вход модели поступают размеченные данные (датасет) – входы и желаемые выходы. Модель сравнивает свой выход с желаемым и меняет свои параметры для улучшения рассогласования (метрика ошибки).

Основная решаемая задача – предсказание (*регрессия*) и классификация (в заданное число классов).

2) обучение без учителя (unsupervised learning);

Классы не заданы. Желаемого выхода нет.

Основная решаемая задача – кластеризация (группировка данных по заранее не заданным кластерам).

ML-модели

Machine learning (машинное обучение) – методы, которые используют обучение на данных для решения сходных задач.

Три основных вида:

3) обучение с подкреплением (reinforcement learning)

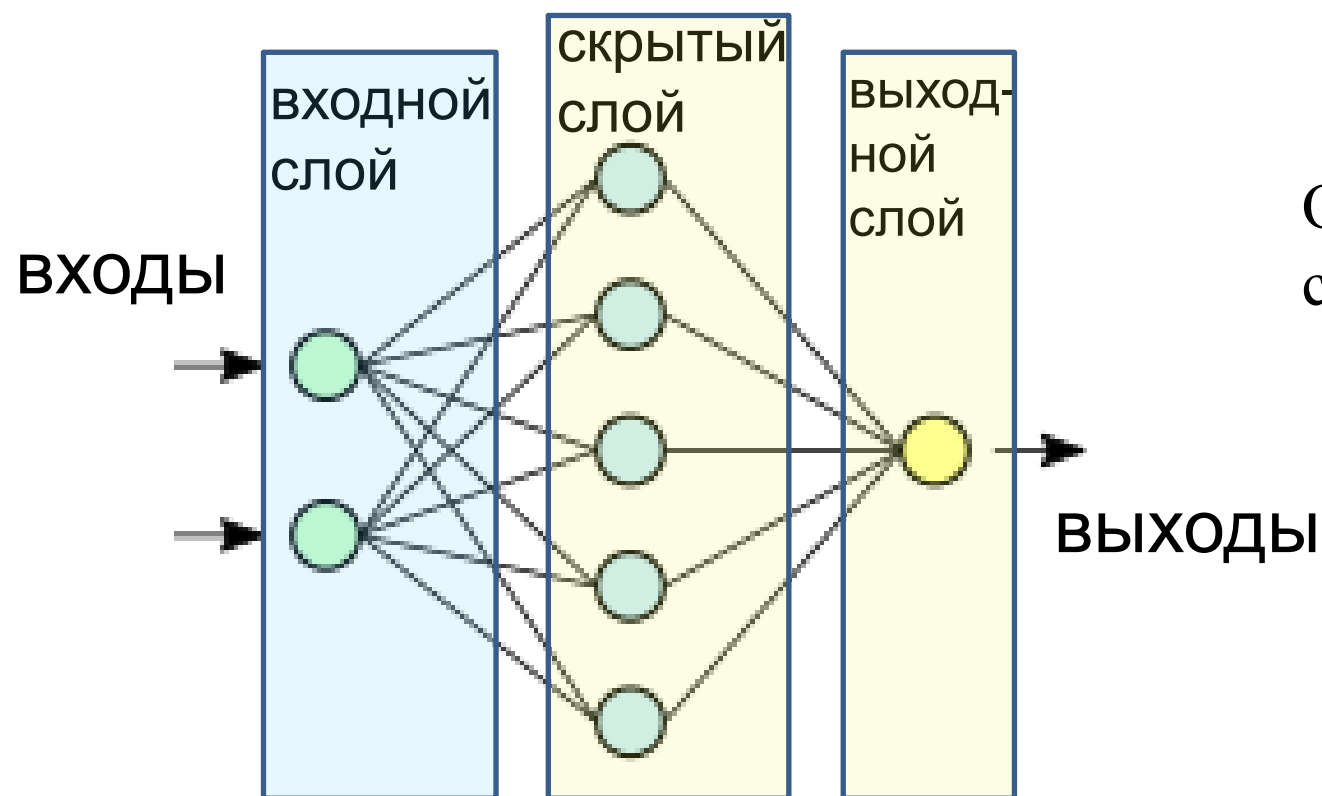
Есть критерий оценки работы в целом (без отдельных примеров). Алгоритм изменяется так, чтобы улучшить критерий.

Пример: автоматическое управление транспортом

Искусственная нейронная сеть

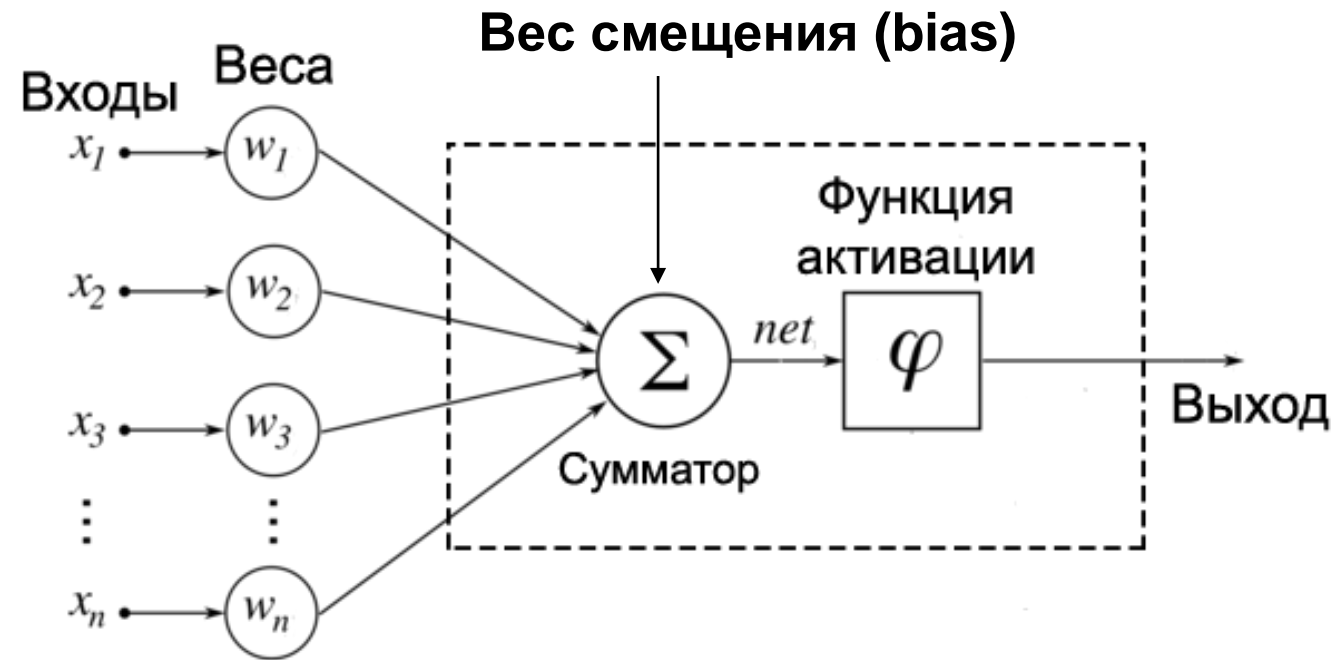
Для решения задач ML часто используют ИНС – набор искусственных нейронов, соединенных синаптическими связями. Имитируют работу нервной системы (передачу сигналов).

Существуют множество типов ИНС. Самый простой для решения задачи предсказания – многослойный перцептрон. Нейроны обозначают кругами, которые группируются в слои в направлении прохождения сигналов, а связи – линиями. Стандартный мн. перц. – полносвязанный между соседними слоями.



Обучение нейросети – подбор коэффициентов синаптических связей

Схема работы отдельного нейрона



Этапы решения задачи с помощью ИНС

1. Сбор датасета.
2. Очистка и нормализация данных.
3. Выбор типа сети.
4. Обучение сети (на обучающей и тестовой выборке).
5. Валидация ИНС (на валидационной выборке).

Правила:

1) число связей ИНС должно быть $<$ числа примеров в обучающей выборке из датасета. Традиционно, в 10 раз.

2) число нейронов скрытого слоя


$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$$

Иногда работает формула $N_h = N_i + N_o$

i - входной слой, o - выходной, $\alpha = 2$ для начала.

Нормализация данных

После подготовки данные должны выглядеть как строки: набор входов и набор выходов.



Входы

Выходы

Temperature,Pressure,Density,GL		
134.9	6.70E-06	735,0
150	8.60E-05	720.9,0
200	0.0194	674,0
220	0.0781	654.8,0
240	0.241	635.1,0

Нормализация - приведение данных к единому масштабу. Часто может улучшить качество модели.

Например, масштабирование к диапазону [0.1 1].

В случае данных, меняющихся на порядки, возможен переход к логарифмической шкале (логарифмирование).

Обучение нейросети

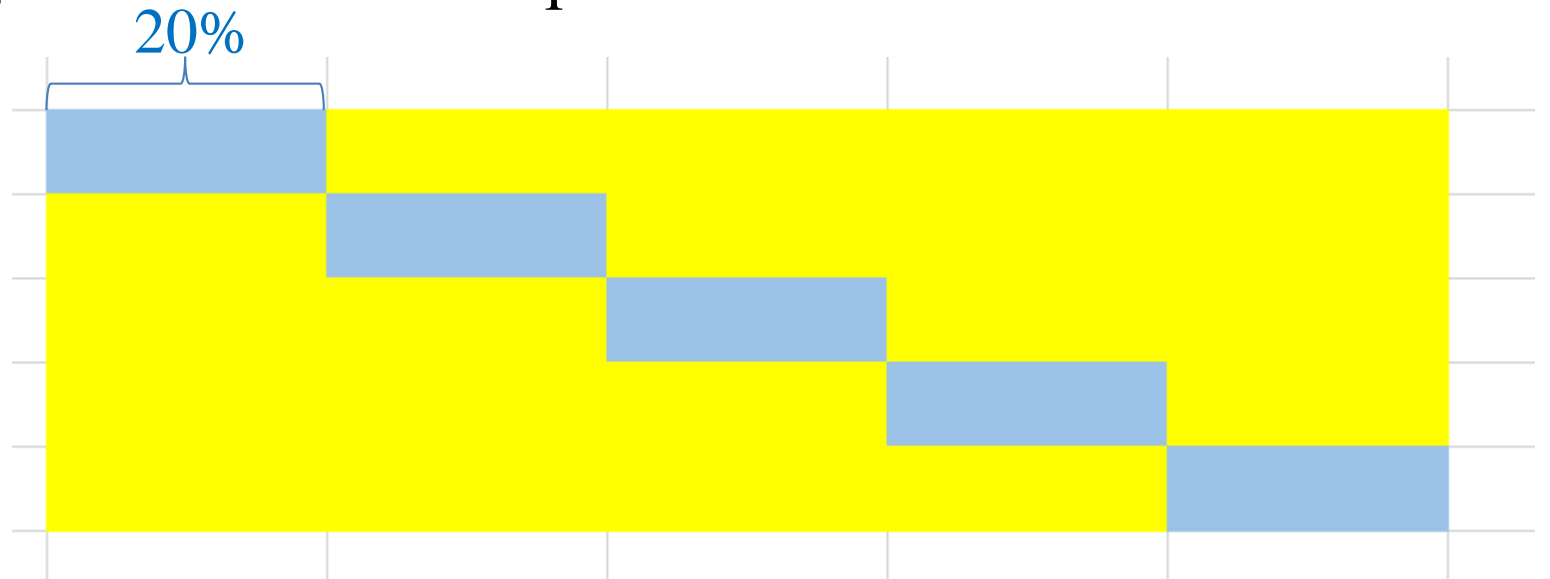
Датасет (строки) делится на две части:

- Обучающая выборка (80%)
 - Тестовая выборка (20%)
- (иногда — три — еще валидационная)

Обучение состоит из фиксированного числа эпох (например, 1000). На каждой эпохе все данные обучающей выборки по одному образцу поступают на вход ИНС.

Корректируются весовые коэффициенты. Потом метрики ИНС вычисляются на тестовой выборке.

Кроссвалидация — делим выборку по-разному. 5-кратная кроссвалидация — 5 раз обучаем полностью ИНС с нуля, причем 20% данных для тестовой выборки берем все 5 раз по-разному.



ML-библиотеки с ИНС

1. Keras



2. Scikit-learn



PyTorch, TensorFlow, Theano

Код для обучения многослойного перцептрона



Получить датасет

```
def get_dataset():  
    # X: samples * inputs  
    # y: samples * outputs
```

```
data = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/data_butane.csv")  
data=data[data["GL"]==0]  
X=data[["Temperature","Pressure"]].to_numpy()  
y=data[["Density"]].to_numpy() #,"GL"  
return X, y
```

Нормализовать каждый столбец [0.1 0.9]

```
def normalize_data(X):  
    nX=X.copy();  
    minsX=[]  
    maxsX=[]  
    for j in range(0,X.shape[1]):  
        minsX.append(min(X[:,j]))  
        maxsX.append(max(X[:,j]))  
        for i in range(0,X.shape[0]):  
            nX[i,j]=(X[i,j]-minsX[j])/(maxsX[j]-  
minsX[j])*0.9+0.1  
    return nX,minsX,maxsX
```

Код для обучения многослойного перцептрона

Построить модель

```
def get_model(n_inputs, n_outputs):  
    model = Sequential()  
    model.add(Dense(2, input_dim=n_inputs, activation='sigmoid'))  
    model.add(Dense(n_outputs, activation='linear'))  
    opt1=optimizers.Adam(learning_rate=0.005)  
    model.compile(loss='mae', metrics = ['mape'], optimizer=opt1)  
    model.summary()  
    return model
```

Обучить модель 5 раз. 1) разбить данные на 20% и 80%

```
def evaluate_model(X, y):  
    n_inputs, n_outputs = X.shape[1], y.shape[1]  
    cv = RepeatedKFold(n_splits=5, n_repeats=1, random_state=22527)  
    i=0  
    MAPE=300  
    ##K-fold  
    for train_ix, test_ix in cv.split(X):  
        # prepare data  
        i=i+1  
        ##for K-fold  
        X_train, X_test = X[train_ix], X[test_ix]  
        y_train, y_test = y[train_ix], y[test_ix]
```

Код для обучения многослойного перцептрона

Обучить модель 5 раз. 2) обучение (model.fit) и 3)
оценка качества работы модели (model.evaluate)

```
model = get_model(n_inputs, n_outputs)
# fit model
history = model.fit(X_train, y_train, verbose=0, epochs=1000)
...
[mae_train, mape_train] = model.evaluate(X_train, y_train)
[mae_test, mape_test] = model.evaluate(X_test, y_test)
[mae, mape] = model.evaluate(X, y)
if (mape < MAPE):
    MAPE = mape
    model2 = model
print('fold: %d' % i)
print('> MAE train: %.3f' % mae_train)
print('> MAE test: %.3f' % mae_test)
print('> MAPE train: %.3f' % mape_train)
print('> MAPE test: %.3f' % mape_test)
print('> MAE total: %.3f' % mae)
print('> MAPE total: %.3f' % mape)
return mae, mape, model2
```

Предсказать, денормализовать результаты

```
new_y = model.predict(X)
dnX = denormalize_data(X, minsX, maxsX)
dny = denormalize_data(y, minsy, maxsy)
new_y = denormalize_data(new_y, minsy, maxsy)
```