Modelul bag-of-words. Kernel Ridge Regression

1. Modelul bag-of-words

➔ este o metodă de reprezentare a datelor de tip text, bazată pe frecvența de apariție a cuvintelor în cadrul documentelor

➔ algoritmul este alcătuit din 2 pași:

1. definirea unui vocabular prin atribuirea unui id unic fiecărui cuvânt regăsit în setul de date (setul de antrenare)

2. reprezentarea fiecărui document ca un vector de dimensiune egală cu lungimea vocabularului, definit astfel:

$$features(word\_idx) = numarul\ de\ aparitii\ al\ cuvantului\ cu\ id-ul\ word\_idx$$

Automated Essay Scoring[1]

În cadrul laboratorului vom folosi un subset din setul de date propus la competiția "The Hewlett Foundation: Automated Essay Scoring", care conține eseuri aparținând unor studenți și punctajele asociate acestora, din intervalul [2, 12].

Scopul este să învățăm un model de regresie care primește reprezentarea unui eseu în formatul bag-of-words și prezice nota corespunzătoare.

Setul de date a fost împărțit în 80% exemple de antrenare (1426) și 20% de testare (357). Acestea se regăsesc în fișierele "train_data.csv", respectiv "test_data.csv". Exemplele sunt stocate pe linii, pe prima coloană regăsindu-se textul corespunzător eseului, iar pe cea de-a doua punctajul, ca în imaginea de mai jos:

essay	score
Dear local newspaper, I think effects computers have on people are great learning	8
Dear @CAPS1 @CAPS2, I believe that using computers will benefit us in many wa	9
Dear, @CAPS1 @CAPS2 @CAPS3 More and more people use computers, but not	7
Dear Local Newspaper, @CAPS1 I have found that many experts say that comput	10

Codul care citește datele este prezentat în continuare:

```
import csv
import re


def read_data(file_path):
    data = []
    scores = []
```

[1] https://www.kaggle.com/c/asap-aes

```
    with open(file_path, mode='r') as csv_file:
        csv_reader = csv.DictReader(csv_file)
        for row in csv_reader:
            data.append(re.sub("[-.,;:!?\"\'\/()_*=`]", "", row["essay"].lower()).split())
            scores.append(int(row["score"]))
    return data, scores


train_data, train_scores = read_data("Data/train_data.csv")
test_data, test_scores = read_data("Data/test_data.csv")
```

Cuvintele au fost transformate astfel încât să conțină numai litere mici, iar caracterele speciale au fost eliminate. Datele sunt ținute sub forma:

$$train\_data[i] = lista\ continand\ cuvintele\ din\ eseul\ cu\ indicele\ i$$
$$train\_scores[i] = scorul\ corespunzator\ eseului\ cu\ indicele\ i$$

2. Kernel Ridge Regression

- ➔ Algoritmul combină regresia ridge cu funcțiile nucleu
- ➔ Acesta minimizează funcția:

$$|| y_{hat} - y||^2 + alpha * ||w||_2^2 \quad \text{, unde}$$

$$y_{hat} = \sum_{i=1}^{n} \alpha_i k(x_i, x'),$$

$$\alpha = (K + \lambda I_n)^{-1} y$$

n - numărul de exemple din setul de antrenare

y - etichetele corecte

y_{hat} - etichetele prezise

k - funcție nucleu

$$\mathbf{K} = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \vdots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix}$$

Importarea modelului:

```
from sklearn.kernel_ridge import KernelRidge
```

Detalii de implementare:

1. **Definirea modelului:**

```
class sklearn.kernel_ridge.KernelRidge(alpha, kernel, gamma)
```

Parametri:

alpha (float)

- ponderea termenului de regularizare al modelului
  - alpha mare => regularizare puternică
  - alpha = 0 => nu se face regularizare

kernel (string, default = ‘rbf’)

- tipul de kernel folosit: în cadrul laboratorului vom lucra cu ‘linear’ și ‘rbf’

Kernel linear:

$$K(u, v) = u^T v$$

Kernel RBF:

$$K(u, v) = exp(-gamma * ||u - v||^2)$$

gamma (float, default =None)

- coeficient pentru kernelul ‘rbf’

2. **Antrenarea:**

```
krr_model.fit(train_data, train_labels)
```

Parametri:

train_data

- setul de antrenare având exemplele stocate pe linii => dimensiune ($num\_samples$ x $num\_features$)

train_labels

- etichetele corespunzătoare fiecărui exemplu de antrenare

3. **Predicția**

```
krr_model.predict(test_data)
```

Parametri:

test_data

- setul de test având exemplele stocate pe linii => dimensiune ($num\_test\_samples$ x $num\_features$)

Funcția întoarce un vector cu $num_test_samples$elemente de tip *float*, fiecare reprezentând valoarea prezisă de model pentru respectivul exemplu.

Exerciții

1. Încărcați setul de date “Automated Essay Scoring”.

2. Definiți clasa **Bag_of_Words** în al cărui constructor se inițializează vocabularul (un dicționar gol). În cadrul ei implementați metoda **build_vocabulary(self, data)** care primește ca parametru o listă de eseuri(listă de liste de strings) și construiește vocabularul pe baza acesteia. Cheile dicționarului sunt reprezentate de cuvintele din eseuri, iar valorile de id-urile unice atribuite cuvintelor.
   Afișati vocabularul construit (14555).

   OBS. Vocabularul va fi construit doar pe baza datelor din setul de antrenare.

3. Definiți metoda **get_features(self, data)** care primește ca parametru o listă de eseuro de dimensiune $num\_samples$(listă de liste de strings) și returnează o matrice de dimensiune ($num\_samples$ x $dictionary\_length$) definită astfel:

$$features(sample\_idx, word\_idx) = numarul\ de\ aparitii\ al\ cuvantului\ cu\ id-ul\ word_{idx}\ in\ documentul\ sample\_idx$$

4. Normalizați vectorii de caracteristici obținuți folosind norma “L2”.

5. Antrenați un model KRR folosind kernel-ul ‘rbf’ și $alpha = 10^{-4.25}$. Calculați eroarea medie pătratică (MSE), respectiv eroarea medie în modul (MAE) atât pe datele de antrenare cât și pe cele de test.

$$MSE = \frac{1}{num\_samples} \sum_{i=1}^{num\_samples} (y_i - y_{i\,hat})^2$$

$$MAE = \frac{1}{num\_samples} \sum_{1}^{num\_samples} |y_i - y_{i\,hat}|$$

```
from sklearn.metrics import mean_squared_error, mean_absolute_error

print("MSE train: ", mean_squared_error(train_scores, predicted_scores_train))  # 0.643
print("MSE test: ", mean_squared_error(test_scores, predicted_scores_test))     # 1.080

print("MAE train: ", mean_absolute_error(train_scores, predicted_scores_train)) # 0.630
print("MAE test: ", mean_absolute_error(test_scores, predicted_scores_test))    # 0.832
```