Algorytmy Statystyki Praktycznej 18.03.2020

Materiał 1, Zadania

Paweł Lorek

1 Wstęp - zadania

Poniżej opisane są m.in. dwa zadania. Są one "rozgrzewką" do projektu nr 1. Wystarczy dostarczyć działające pliku Python'owe – ew. uwagi wystarczy wpisać jako komentarze wewnątrz pliku. (Późniejszy projekt będzie już wymagał szczegółowego raportu w .pdf). Zadań tych nie będę szczegółówo oceniał, jest to bardziej zero jedynkowe (trzeba zrobić).

DEADLINE: 11.04.2020 (pliki należy umieścić w Moodle).

2 Lasso, ADM, proximal gradient, Nesterov

Rozważmy model liniowy

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

gdzie $\mathbf{Y} \in \mathbb{R}^n$ (kolumna), $\mathbf{X} \in \mathbb{R}^{n\times d}, \boldsymbol{\beta} \in \mathbb{R}^d$ oraz $\boldsymbol{\varepsilon} \in \mathbb{R}^n$. Oznaczmy

$$\mathbf{Y} = (y_1, , \ldots, y_n)^T,\ \boldsymbol{\beta} = (\beta_1, \ldots, \beta_d)^T,\ \boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_d)^T,$$

wyrazy macierzy $\mathbf{X}$ to $x_{ij}, i = 1, \ldots, n, j = 1, \ldots, d$.

Jest to model z wyrazem wolnym – pierwsza kolumna macierzy $\mathbf{X}$ to same jedynki. W sytuacji, gdy $d > n$ estymacja $\boldsymbol{\beta}$ wymaga dodania regularyzacji/kary. Jedną z najpopularniejszych metod jest LASSO, czyli estymator $\hat{\boldsymbol{\beta}}$ zdefiniowany jest następująco ($\lambda > 0$ jest parametrem)

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{ ||\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}||^2 + \lambda \sum_{i=2}^{d} |\beta_i| \right\}. \tag{1}$$

(dla przypomnienia, np: metoda najmniejszych kwadratów nie ma drugiego wyrazu z λ)

Zadanie 1 Zaimplementuj następujące 3 algorytmy do powyższego problemu Lasso podanego w (1).

- Proximal gradient
- Przyspieszenie Nesterova dla proximal gradient
- Alternating Direction Method of Multipliers (ADMM)

Szczegóły wszystkich tych algorytmów były podane na wykładzie.

Uwaga 1 *Projekt nr 1 będzie związany z powyższymi algorytmami. "I tak" trzeba będzie je zaimplementować. Następne zadanie to poćwiczenie/zastosowanie powyższych algorytmów.*

3 Python

Można zacząć od przetestowania algorytmów w dostępnych bibliotekach. Trochę inne sformułowanie problemu jest w bibliotece `sklearn`. Warto się zapoznać z:
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html

Więcej szczeółów jest w udostępnionym w Moodle pliku `example_linear_regression_lasso.py` (tam jest i LASSO i regresja liniowa, dodatkowo robione są rysunki), tutaj prosty przyklad uzycia:

```
import numpy as np
from sklearn import linear_model

normal1_mean=[4, 6];
normal1_cov = [[2,4], [4,11]];

#symulujemy 100 punktow z rozkladu normalnego dwuwymiarowego:
x1,y1 = np.random.multivariate_normal(normal1_mean , normal1_cov, 100).T
# tworzymy model liniowy Lasso (alpha to "nasza" lambda)
model = linear_model.Lasso(alpha=1)
# dopasowywujemy go do punktow x1, y1
model.fit (x1.reshape(-1,1),y1)
# 'wydobywamy' wspolczynniki beta1 i beta2 (y=beta1+ beta2*x)
beta1=model.coef_[0]
beta2=model.intercept_
```

4 `kc_house_data.csv`

W Moodle udostępniony jest plik `kc_house_data.csv.zip`. Dane te pochodzą z Kaggle:
https://www.kaggle.com/shivachandel/kc-house-data

Plik ten (po rozpakowaniu) ma taką strukturę:

- Pierwszy wiersz: wymienione nazwy pól po przecinku
- Kolejne wiersze = dane

Pierwsze 3 linijki

```
  id,date,price,bedrooms,bathrooms,sqft_living,sqft_lot,floors,waterfront,view,condition,grade,sqft_above,sqft_basement,yr_built,yr_renovated,zipcode,lat,long,sqft_
"7129300520","20141013T000000",221900,3,1,1180,5650,"1",0,0,3,7,1180,0,1955,0,"98178",47.5112,-122.257,1340,5650
"6414100192","20141209T000000",538000,3,2.25,2570,7242,"2",0,0,3,7,2170,400,1951,1991,"98125",47.721,-122.319,1690,7639
```

Takie pliki .csv (w szczególności, gdy są bardzo duże) najwygodniej wczytać używając biblioteki `pandas` (nie musimy znać tej biblioteki szczegółowo, wystarczy poniższy przykład – ostatecznie macierz `Xnp` jest już typu `numpy`)

```
1 import numpy as np
2 import pandas as pd
3
4 dane = pd.read_csv(kc_house_data_csv)
5 print(type(dane)) # jest to tzw. typ DataFrame
6 print(dane.head())
7 X=dane[['price','bedrooms']]
8 Xnp=X.values
```

Krótkie wyjaśnienie:

- Linia 4: wczytanie danych , w linii 6 widzimy, iz typem jest `DataFrame`
- Linia 5: wyswietla skrotowo kilka pierwszych linii (dodaje "naglowki" oraz id wierszy (pierwsza kolumna)
- Linia 7: wydobywamy kolumny "price" oraz "bedrooms"
- Linia 8: Zamiana `X` typu `DataFrame` na macierz Numpy

Oczywiście powyższy przykład wybiera tylko 'price' i 'bedrooms' – należy wydobyć wszystkie możliwe kolumny (tzw. cechy), a także wstawić pierwszą kolumnę `Xnp` złożoną z samych jedynek.

Zadanie 2 Na podstawie powyższego przykładu, wczytaj całą macierz $\mathbf{X}$ (powyżej nazywała się `Xnp`) oraz $\mathbf{Y}$ (wartości domów). Zastosuj wszystkie 3 algorytmy z Zadania 1, sprawdź który się sprawdzi najlepiej (oczywiście to zależy również od tego jakie parametry dobierzesz).

<u>Dodatkowo</u> możesz zrobic *przewidywanie cen domów*. Podziel zbiór $\mathbf{X}$ (i $\mathbf{Y}$) na zbiór `treningowy` (ok. 80% danych) oraz `testowy` (pozostale 20% danych). Algorytmy wyuczamy (= estymujemy $\boldsymbol{\beta}$) tylko na danych `treningowych`, natomiast używając wyuczonych bet estymujemy ceny domów ze zbioru `testowego`, ostatecznie wyliczamy `mean_squared_error` (znamy prawdziwe ceny, możemy to zatem zrobić).