Sprawozdanie - Maciej Groszyk 289761

Zadanie 4:

Zaimplementuj algorytm SVM oraz zbadaj działanie algorytmu w zastosowaniu do zbioru danych Wine Quality Data Set. W celu dostosowania zbioru danych do problemu klasyfikacji binarnej dyskretyzacja zmienną objaśnianą. Pamiętaj, aby podzielić zbiór danych na zbiór trenujący oraz uczący.
Zbadaj wpływ hiper parametrów na działanie zaimplementowanego algorytmu. W badaniach rozważ dwie różne funkcje jądrowe poznane na wykładzie.

1. Wstęp

Formalizacja problemu klasyfikacji:

W przestrzeni danych omega znajdują się znajdują się wektory danych x stanowiące próbkę uczącą D, należące do dwóch klas

$$D = \left\{ \left(\mathbf{x}_i, c_i \right) \middle| \, x_i \in R^p, c_i \in \{1, -1\} \right\}_{i=1}^{N}$$

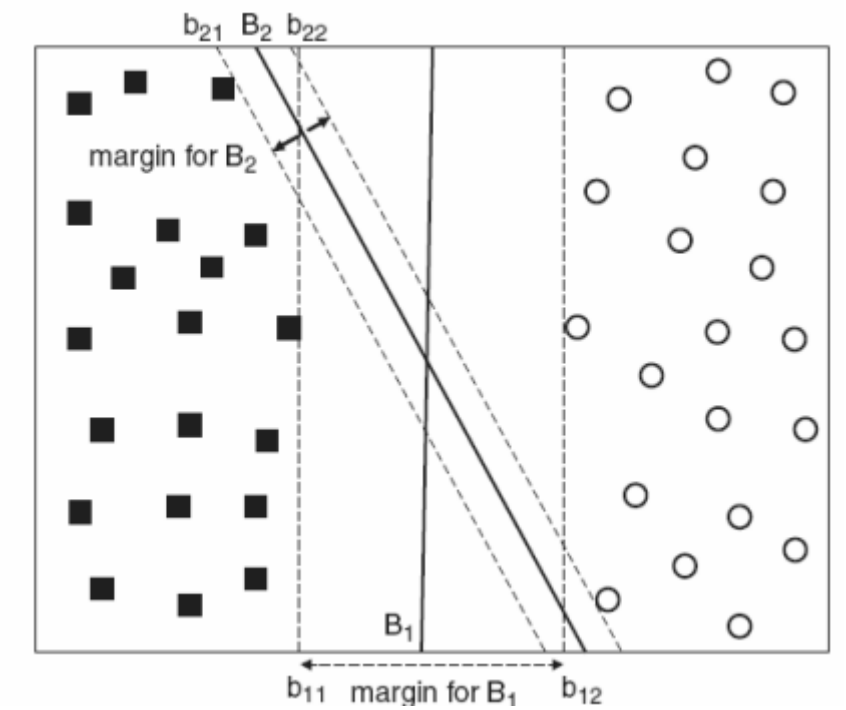
Szukamy klasyfikatora, który pozwoli na podział tej przestrzeni na dwie części odpowiadające klasom 1 i -1 oraz pozwoli na jak najlepsze przypisanie obiektów do tych klas.

Dwie klasy są liniowo separowalne pod warunkiem że istnieje hiperpłaszczyzna H w postaci g(x):

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

która przyjmuje wartości:

$$\begin{cases} g(\mathbf{x}_i) > 0 & \mathbf{x}_i \in 1 \\ g(\mathbf{x}_i) < 0 & \mathbf{x}_i \in -1 \end{cases}$$



Hiperpłaszczyzna B tworzy margines o odległości równej odległości B do najbliższych punktów obu klas. Należy pamiętać aby wybierać najszerszy margines ponieważ posiada on lepsze własności generalizacji, a także ma mniejsza podatność na przeuczenie.

Należy wyznaczyć parametry w ten sposób aby aby maksymalne marginesy były miejscem geometrycznym punktów x:

$$b_{i1} \quad \mathbf{w} \cdot \mathbf{x} + \mathbf{b} = 1$$

$$b_{i2} \quad \mathbf{w} \cdot \mathbf{x} + \mathbf{b} = -1$$

Po przekształceniach algebraicznych celem końcowym jest zmaksymalizowanie

$$\frac{2}{\|\mathbf{w}\|}$$

co można przekształcić w minimalizacje

$$\frac{\|\mathbf{w}\|^2}{2}$$

Jest to problem optymalizacji kwadratowej z liniowymi ograniczeniami -> uogólnione zadanie optymalizacji rozwiązywany metodą mnożników Lagrange'a.

$$L(w,b,\alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i (y_i(\mathbf{w}\mathbf{x}_i + b) - 1)$$

Problem też można przedstawić w postaci dualnej, która była optymalizowana w mojej implementacji:

Poszukujemy maksimum tej funkcji:

$$\sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

Klasyfikacja odbywa się za pomocą funkcji signum

$$f(x) = sign(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b)$$

Która przyjmuje parametry alpha z wyniku maksymalizacji funkcji w postaci dualnej.

W momencie gdy dane nie są w pełni separowalne, należy zastosować funkcję jądrowe które pozwalają na obliczenie wartości iloczynu skalarnego w rozszerzonej przestrzeni (która jest wymagana ze wzgledu na nie separowalność danych)

W implementacji przedstawiłem działania funkcji jądrowej normalnej (gaussowskiej) i wielomianowej.

2. Przygotowanie danych

Aby móc skorzystać z algorytmu svm należy najpierw przygotować dane wejściowe. W tym celu wykorzystałem dane Wine Quality Data Set. Najpierw dane zostały wczytane do ramki danych, a następnie została im zmieniona wartość quality na -1 bądź 1 w zależności od ustawionego progu.

Następnie zbiór danych został podzielony na wektory danych X które zawierają wszystkie kolumny oprócz kolumny "Quality" która tworzy wektor danych Y.

Przed podziałem zbioru na podzbiór trenujący i testujący, został znormalizowany wektor danych X za pomocą funkcji normalize z modułu sklearn.

Zbiór X i Y został podzielony w stosunku 80% zestaw treningowy i 20% zestaw testowy. W ten sposób przygotowane dane są już gotowe do użycia w SVM.

3. Algorytm SVM

W programie została zaimplementowana klasa, która przyjmuje w konstruktorze następujące parametry:

- Zestaw treningowy X
- Zestaw treningowy Y
- Parametr funkcji jądrowej
- funkcję jądrową
- Parametr ograniczający C

Do celu optymalizacji funkcji dualnej został wykorzystany solver cvxopt. (https://courses.csail.mit.edu/6.867/wiki/images/a/a7/Qp-cvxopt.pdf)

dodatkowo został zaimplementowany parametr ograniczający C który pozwala na skuteczne znalezienie optymalnego rozwiązania.

Następnie obliczamy wartość b i podstawiamy ją do funkcji klasyfikującej, która przydziela wektor do odpowiedniej klasy

4. Wyniki

Założenia:

Dobre wino miało parametr quality > 5
Parametr ograniczający C przyjmuje wartość 100

A. Dla funkcji jądrowej normalnej (rbf):

wartość sigmy = 0.1

```
Optimal solution found.
good predictions 218/320  -> 68.125%
number of good wines 181/320 there is 161 good wines
number of bad 139/320 there is 159 bad wines
```

wartość sigmy = 0.8

```
Optimal solution found.
good predictions 234/320  -> 73.125%
number of good wines 167/320 there is 161 good wines
number of bad 153/320 there is 159 bad wines
```

Wartość sigmy = 2

```
Optimal solution found.
good predictions 230/320  -> 71.875%
number of good wines 175/320 there is 161 good wines
number of bad 145/320 there is 159 bad wines
```

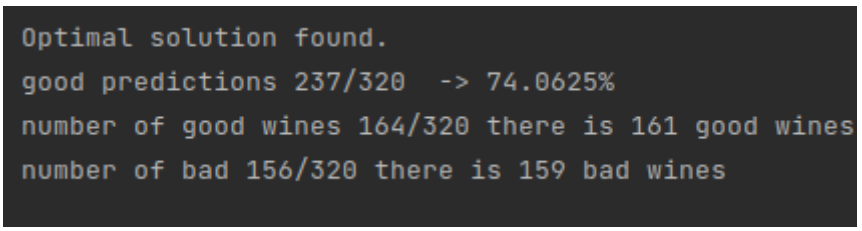
Skuteczność algorytmu z wykorzystaniem funkcji jądrowej rbf jest równa 71.875%
Algorytm poprawnie sklasyfikował 230 wektory. Dobrych win sklasyfikował więcej niż w rzeczywistości istniały.

Wartość sigmy = 5

```
Optimal solution found.
good predictions 219/320  -> 68.4375%
number of good wines 164/320 there is 161 good wines
number of bad 156/320 there is 159 bad wines
```

Podczas zwiększenia sigmy skuteczność predykcji zmalała o ponad 3 punkty procentowe.

Wartość sigmy = 1



Skuteczność przy sigmie = 1 jest najwyższa. Przy C = 100 w momencie zwiększania sigmy zmniejsza się skuteczność predykcji. Przy zbyt małej wartości sigmy również predykcja się zmniejsza.

B. Dla funkcji jądrowej wielomianowej (poly):

Stopień funkcji jądrowej: 2

```
Optimal solution found.
good predictions 230/320  -> 71.875%
number of good wines 177/320 there is 161 good wines
number of bad 143/320 there is 159 bad wines
```

Stopień funkcji jądrowej: 5

```
Optimal solution found.
good predictions 230/320  -> 71.875%
number of good wines 171/320 there is 161 good wines
number of bad 149/320 there is 159 bad wines
```

Stopień funkcji jądrowej 10:

```
Optimal solution found.
good predictions 235/320  -> 73.4375%
number of good wines 168/320 there is 161 good wines
number of bad 152/320 there is 159 bad wines
```

Stopień funkcji jądrowej 100:

```
Optimal solution found.
good predictions 236/320  -> 73.75%
number of good wines 165/320 there is 161 good wines
number of bad 155/320 there is 159 bad wines
```

Podczas zwiększania stopnia funkcji wielomianowej zwiększa się skuteczność działania programu.

Przy parametrze ograniczającym C = 10:
sigma = 0.6

```
Optimal solution found.
good predictions 231/320  -> 72.1875%
number of good wines 174/320 there is 161 good wines
number of bad 146/320 there is 159 bad wines
```

sigma = 0.8

```
Optimal solution found.
good predictions 228/320  -> 71.25%
number of good wines 177/320 there is 161 good wines
number of bad 143/320 there is 159 bad wines
```

sigma = 1

```
Optimal solution found.
good predictions 227/320  -> 70.9375%
number of good wines 174/320 there is 161 good wines
number of bad 146/320 there is 159 bad wines
```

sigma = 2

```
Optimal solution found.
good predictions 221/320  -> 69.0625%
number of good wines 164/320 there is 161 good wines
number of bad 156/320 there is 159 bad wines
```

W przypadku C = 10 najskuteczniejsza sigma jest w okolicach wartości 0.8.

Dla funkcji jądrowej wielomianowej przy stopniu = 2 i C = 10:

```
Optimal solution found.
good predictions 213/320  -> 66.5625%
number of good wines 156/320 there is 161 good wines
number of bad 164/320 there is 159 bad wines
```

Przy stopniu 10:

```
Optimal solution found.
good predictions 226/320  -> 70.625%
number of good wines 169/320 there is 161 good wines
number of bad 151/320 there is 159 bad wines
```

przy stopniu 100:

```
Optimal solution found.
good predictions 237/320  -> 74.0625%
number of good wines 176/320 there is 161 good wines
number of bad 144/320 there is 159 bad wines
```

Najwyższa wartość ponownie jest przy stopniu równym 100. Przy przykładowym zwiększeniu do 400, skuteczność nieznacznie się zmniejszy.

Przy C = 1 dla sigmy = 1 w funkcji rbf:

```
good predictions 213/320  -> 66.5625%
number of good wines 140/320 there is 161 good wines
number of bad 180/320 there is 159 bad wines
```

Przy C = 1 dla sigmy = 0.1

```
Optimal solution found.
good predictions 229/320  -> 71.5625%
number of good wines 168/320 there is 161 good wines
number of bad 152/320 there is 159 bad wines
```

w funkcji wielomianowej dla C =1, dla stopnia = 1:

```
Optimal solution found.
good predictions 180/320  -> 56.25%
number of good wines 285/320 there is 161 good wines
number of bad 35/320 there is 159 bad wines
```

dla stopnia = 5

```
Optimal solution found.
good predictions 195/320  -> 60.9375%
number of good wines 192/320 there is 161 good wines
number of bad 128/320 there is 159 bad wines
```

dla stopnia = 100

```
Optimal solution found.
good predictions 201/320  -> 62.81250000000001%
number of good wines 170/320 there is 161 good wines
number of bad 150/320 there is 159 bad wines
```

5. Wnioski:

Algorytm SVM jest algorytmem wrażliwym na parametry. Nawet nie właściwy dobór jednego z nich może skończyć się stratą kilkunastu punktów procentowych skuteczności predykcji. Najlepsza skuteczność udało mi się uzyskać przy funkcji jądrowej rbf z wartością sigmy 1 i parametru C = 100. Ten sam wynik udało się uzyskać również za pomocą wielomianowej funkcji jądrowej przy parametrze C = 10 i stopniu wielomianu równym 100.