Wprowadzenie do sztucznej inteligencji - Laboratorium 4

Ireneusz Okniński 310228

Grudzień 2022

1 Treść zadania

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 zdyskretyzuj zmienną objaśnianą. Pamiętaj, aby podzielić zbiór danych na zbiór trenujący oraz uczący.Zbadaj wpływ hiperparametrów na działanie implementowanego algorytmu. W badaniach rozważ dwie różne funkcje jądrowe poznane na wykładzie.

2 Opis algorytmu

2.1 Liniowa separacja

Hiperpłaszczyznę wyznacza funkcja:

$$f(x) = w^T x + b$$

gdzie w jest wektorem normalnym do hiperpłaszczyzny, a $\frac{b}{||w||}$ jest odległością płaszczyzny od początku układu współrzędnych. Rozwiązaniem problemu klasyfikacji jest w tym przypadku znalezienie wartości w, b takich, że:

$$\forall i \; y_i(x_i \cdot w + b) - 1 \geqslant 0$$

Szerokość marginesu wynosi $\frac{1}{||w||}$ zatem znalezienie najszerszego marginesu czyli $max(\frac{1}{||w||}$ jest równoznaczne z minimalizacją $||w||$. Minimalizacja $||w||$ jest równoznaczna z minimalizacją $\frac{1}{2}||w||^2$. Umożliwi to użycie programowania kwadratowego do rozwiązania problemu, który sprowadza się do:

$$min\frac{1}{2}||w||^2 \quad s.t \quad \forall i \; y_i(x_i \cdot w + b) - 1 \geqslant 0$$

Aby uwzględnić ograniczenia optymalizacji musimy wprowadzić mnożniki Lagrange'a:

$$L(w, b, \alpha) \equiv \frac{1}{2}||w||^2 - \alpha[y_i((x_i \cdot w + b) - 1)\forall i]$$

$$\equiv \frac{1}{2}||w||^2 - \sum_{i=1}^{L} \alpha_i y_i((x_i \cdot w + b) - 1)$$

$$\equiv \frac{1}{2}||w||^2 - \sum_{i=1}^{L} \alpha_i y_i(x_i \cdot w + b) + \sum_{i=1}^{L} \alpha_i$$

Aby wyznaczyć w i b które minimalizują oraz α które maksymalizuje powyższą funkcję przy założeniu $\alpha_i \geqslant 0 \forall i$ musimy policzyć pochodne funkcji po w i b:

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{L} \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{L} \alpha_i y_i = 0$$

Podstawiając otrzymane równania do problemu otrzymujemy:

$$L(w, b, \alpha) \equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j \;\; s.t. \;\; \alpha_i \geqslant 0 \; \forall i, \;\; \sum_{i=1}^{L} \alpha_i y_i = 0$$

Przyjmując $H_{i,j} = y_i y_j x_i x_j$ otrzymujemy:

$$L(w, b, \alpha) \equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i H_{ij} \alpha_j$$

$$\equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \alpha^T H \alpha \;\; s.t. \;\; \alpha_i \geqslant 0 \; \forall i, \;\; \sum_{i=1}^{L} \alpha_i y_i = 0$$

Rozwiązaniem problemu jest teraz wartość α dla której powyższa funkcja ma największą wartość. Korzystając z otrzymanego wyniku możemy obliczyć:

$$w = \sum_{i=1}^{L} \alpha_i y_i x_i$$

Punkty spełniające równanie $\sum_{i=1}^{L} \alpha_i y_i = 0$ są wektorami nośnymi X_s spełniają równanie:

$$y_s(x_s \cdot w + b) = 1$$

$$y_s(\sum_{m \in S} \alpha_m y_m x_m \cdot x_s + b) = 1$$

gdzie S jest zbiorem indeksów wektorów nośnych. Należą do niego indeksy dla których $\alpha_i > 0$. Możemy pomnozyć obie strony równania przez y_s, $y_s^2 = 1$. Otrzymamy wtedy:

$$y_s^2(\sum_{m \in S} \alpha_m y_m x_m \cdot x_s + b) = y_s$$

$$b = y_s - \sum_{m \in S} \alpha_m y_m x_m \cdot x_s$$

Możemy też użyć średniej wektorów nośnych:

$$b = \frac{1}{N_s} \sum_{s \in S} \left(y_s - \sum_{m \in S} \alpha_m y_m x_m \cdot x_s \right)$$

2.2 Brak liniowej separacji

Gdy dane nie są w pełni liniowo separowalne dopuszczamy błędnie sklasyfikowane punkty. Wprowadzamy zmienną ξ mówiącą o tym jak luźne ograniczenie przyjmujemy. Ograniczenia problemu maksymalizacji marginesu mają teraz postać:

$$y_i(x_i \cdot w + b) - 1 + \xi_i \geqslant 0 \;\; \xi_i \geqslant 0 \;\; \forall i$$

Podobnie jak w przypadku liniowej separacji problem sprowadza się do rozwiązania:

$$min \frac{1}{2}||w||^2 + C \sum_{i=1}^{L} \xi_i \quad s.t \quad \forall i \; y_i(x_i \cdot w + b) - 1 + \xi_i \geqslant 0$$

W tym przypadku dodajemy karę za przekroczenie marginesu. Parametr C jest współczynnikiem kary. Kontroluje wpływ odległości od marginesu na wielkość kary. Analogicznie wprowadzamy mnożniki Lagrange'a:

$$L(w, b, \alpha, \xi) = \frac{1}{2}||w||^2 + C \sum_{i=1}^{L} \xi_i - \sum_{i=1}^{L} \alpha_i [y_i(x_i \cdot w + b) - 1 + \xi_i] - \sum_{i=1}^{L} \mu_i \xi_i$$

Liczymy pochodne po w, b,ξ_i, ($\alpha_i \geqslant 0$, $\mu_i \geqslant 0 \; \forall i$) i przyrównujemy je do zera:

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{L} \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{L} \alpha_i y_i = 0$$

$$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow C = \alpha_i + \mu_i$$

Z warunku $\mu_i \geqslant 0 \; \forall i$ wynika, że $\alpha \leqslant C$. Analogicznie jak wcześniej podstawiamy równania do problemu i otrzymujemy problem maksymalizacji:

$$\sum_{i=1}^{L} \alpha_i - \frac{1}{2} \alpha^T H \alpha \;\; s.t. \;\; 0 \leqslant \alpha_i \leqslant C \; \forall i, \;\; \sum_{i=1}^{L} \alpha_i y_i = 0$$

$$H_{i,j} = y_i y_j x_i x_j$$

Wartości w oraz b wyliczamy podobnie jak w poprzednim przypadku. Jedyną różnicą jest warunek na przynależność indeksu do zbioru indeksów wektorów nośnych. W tym przypadku jest to $0 < \alpha_i \leqslant C$.

2.3 Nieliniowość

W przypadku kiedy danych nie da się podzielić przy użyciu hiperpłaszczyzny możemy rzutować problem do przestrzeni o większej liczbie wymiarów. Można tego dokonać przy użyciu Kernel trick. Wprowadzamy wtedy funkcje jądrowe, dzięki którym będzie można zwiększyć liczbę wymiarów problemu bez jawnego przekształcania przestrzeni, co byłoby kosztowne. Funkcje te mają postać $K(x_i, x_j)$ i ich wartość jest wartością pewnego $\phi(x)$. Po wprowadzeniu funkcji jądrowej macierz H ma postać:

$$H_{i,j} = y_i y_j K(x_i, x_j)$$

W mojej implementacji porównam dwie funkcje jądrowe:

- liniowa $K(x_i, x_j) = x_i^T x_j$
- RFB $K(x_i, x_j) = \exp\left(-\frac{||x_j - x_i||^2}{2\sigma^2}\right)$

Po obliczeniu wartości H problem sprowadza się do jednego z dwóch poprzednich problemów.

3 Implementacja

3.1 Trenowanie modelu

Wprowadzam zmienną objaśnianą: -1 jeżeli ocena jest mniejsza od 5 oraz 1 w przeciwnym przypadku. Algorytm SVM ma na celu znalezienie hiperpłaszczyzny separującej te dwie klasy jak największym marginesem. W celu poprawienia skuteczności algorytmu przeprowadzam normalizację danych (z użyciem sklearn.preprocessing). Otrzymane zbiory x oraz y dzielę na zbiory treningowe i testowe. Zbiór treningowy stanowi 80% całości. Zaimplementowany algorytm SVM zakłada nieliniowość problemu oraz brak liniowej separowalności po przekształceniu do nowej przestrzeni. Problem trenowania modelu sprowadza się do rozwiązania problemu maksymalizacji:

$$\sum_{i=1}^{L} \alpha_i - \frac{1}{2}\alpha^T H \alpha \;\; s.t. \;\; 0 \leqslant \alpha_i \leqslant C \;\forall i, \;\; \sum_{i=1}^{L} \alpha_i y_i = 0$$

$$H_{i,j} = y_i y_j K(x_i, x_j)$$

W celu optymalizacji problemu programowania kwadratowego skorzystam z solvera qp z biblioteki cvxopt. Wymaga on podania problemu w postaci:

$$min \; \frac{1}{2} x^T P x + q^T x$$

$$s.t.$$

$$Gx \leqslant h$$

$$Ax = b$$

gdzie $P = M(n,n)$, $q = M(n,1)$, $G = M(m,n)$, $h = M(m,1)$, $A = M(p,n)$, $b = M(p,1)$ Aby doprowadzić nasz problem do postaci akceptowalnej przez solver mnożę funkcję przez -1:

$$min\frac{1}{2}\alpha^T H\alpha - \sum_{i=1}^{L}\alpha_i$$

$$s.t.$$

$$0 \leqslant \alpha_i \leqslant C$$

$$\sum_{i=1}^{L}\alpha_i y_i = 0$$

Wyprowadzenie P i q: Porównując funkcje:

$$\frac{1}{2}\alpha^T H\alpha - \sum_{i=1}^{L}\alpha_i$$

oraz

$$\frac{1}{2}x^T Px + q^T x$$

można zauważyć, że $P = H$ oraz:

$$q^T\alpha = -\sum_{i=1}^{L}\alpha_i$$

$$q_i\alpha_i = -\alpha_i$$

$$q_i = -1$$

Zatem q ma wymiary $(n,1)$postać:

$$\begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix}$$

Wyprowadzenie G i h:

$$\forall\alpha_i \in \alpha \;\; 0 \leqslant \alpha_i \leqslant C$$

$$\forall\alpha_i \in \alpha \;\; \alpha_i \geqslant 0 \;\; \alpha_i \leqslant C$$

$$\forall\alpha_i \in \alpha \;\; -\alpha_i \leqslant 0 \;\; \alpha_i \leqslant C$$

Założenie takie jest spełnione, jeżeli wynik mnożenia Gx ma postać:

$$\begin{bmatrix} -\alpha_1 \\ \vdots \\ -\alpha_n \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}$$

Aby otrzymać taką postać należy pomnożyć macierz o wymiarze $(2n, n)$ przez macierz α, która ma wymiar $(n, 1)$. Macierz G ma postać:

$$\begin{bmatrix} -1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 0 & & \\ 0 & 0 & -1 & & \vdots \\ \vdots & & & \ddots & \\ 0 & & \dots & & -1 \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & & \\ 0 & 0 & 1 & & \vdots \\ \vdots & & & \ddots & \\ 0 & & \dots & & 1 \end{bmatrix}$$

Wyprowadzenie A i b:

$$\sum_{i=1}^{L} \alpha_i y_i = 0 \Rightarrow y \cdot \alpha = 0$$

Wynika z tego , że b jest macierzą zerową o wymiarze $(1, 1)$, $A = y$ i ma wymiar $(1, n)$.

Wszystkie macierze parametrów solvera wyglądają następująco:

$$P = H, \;\; H_{i,j} = y_i y_j K(x_i, x_j)$$

$$q = \begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix}$$

$$G = \begin{bmatrix} -1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 0 & & \\ 0 & 0 & -1 & & \vdots \\ \vdots & & & \ddots & \\ 0 & & \dots & & -1 \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & & \\ 0 & 0 & 1 & & \vdots \\ \vdots & & & \ddots & \\ 0 & & \dots & & 1 \end{bmatrix}$$

$$h = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ C \\ \vdots \\ C \end{bmatrix}$$

$$A = \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix}$$

$$b = \begin{bmatrix} 0 \end{bmatrix}$$

Po otrzymaniu wyniku z solvera obliczam indeksy, dla których otrzymana $\alpha > \epsilon$, gdzie ϵ jest minimalną wartością mnożnika Lagrange'a powyżej której możemy uznać wektory znajdujące się pod tym samym indeksem co mnoznik za wspierające. Następnie zapisuję zbiory mnożników, wektorów wspierających, oraz ich etykiet znajdujących się pod otzrymanymi wcześniej indeksami. Kolejnym krokiem jest obliczenie wartości:

$$b = \frac{1}{N_s} \sum_{s \in S} \left(y_s - \sum_{m \in S} \alpha_m y_m x_m \cdot x_s \right)$$

3.2 Klasyfikacja

Dla każdego wektora zbioru x_{test} obliczamy:

$$p = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x_{test})$$

gdzie α, x, y są odpowiednio: zbiorami mnożników, wektorów wspierających, etykiet wektorów wspierajacych. Przewidywana etykieta danego wektora to:

$$sgn(p + b)$$

gdzie b jest obliczoną podczas trenowania wartością. Otrzymany zbiór etykiet porównuję ze zbiorem poprawnych etykiet. Na podstawie tego obliczam skuteczność klasyfikacji. Dane zapisuję do plików json, dzięki czemu mogę później wygenerować z nich wykresy.

4 Badania

Do badania skuteczności algorytmu SVM w problemie klasyfikacji wykorzystam pliki: *winequality-white.csv*, oraz *winequality-red.csv*.

4.1 Porównanie funkcji jądrowych

W moich badaniach użyję dwóch funkcji jądrowych:

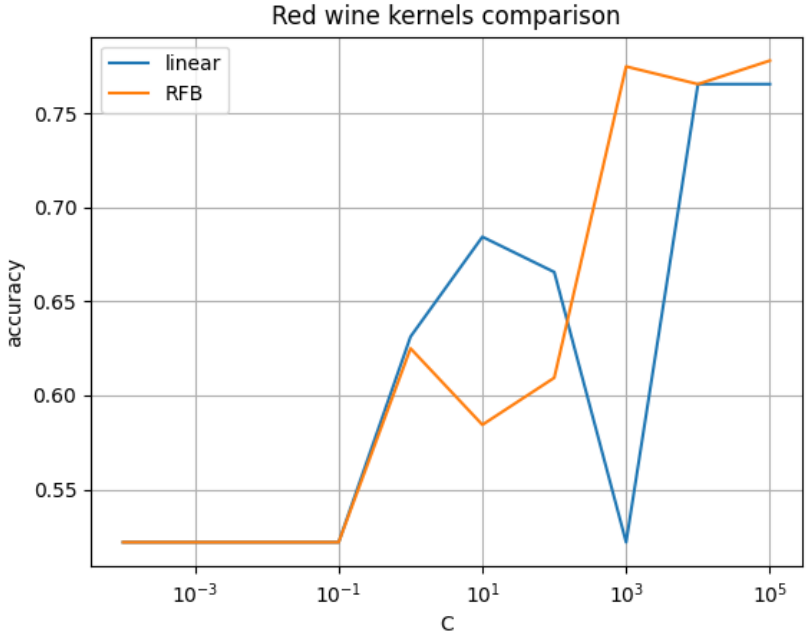
$$(K(x_i, x_j) = x_i^T x_j$$

$$K(x_i, x_j, \sigma) = \exp\left(-\frac{||x_j - x_i||^2}{2\sigma^2}\right)$$

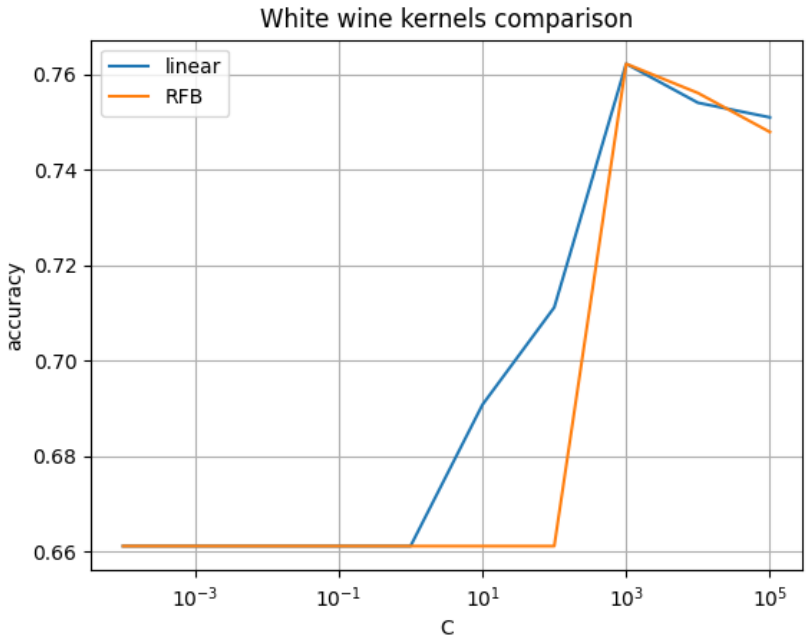
Porównam osiągane przez nie rezultaty dla wartości:

$$C \in \{10^{-4}10^{-3}, ..., 10^4, 10^5\}$$

$$\sigma = 1.0$$



Rysunek 1: Porównanie skuteczności dla zbioru win czerwonych



Rysunek 2: Porównanie skuteczności dla zbioru win białych

W obu zbiorach funkcja RFB osiąga najlepsze rezultaty dla $C = 10^3$. Z wykresu wynika, że skuteczności obu funkcji jądrowych są podobne. Maksymalna dokładność klasyfikacji wynosi tutaj ponad 75%.

4.2 Wpływ parametrów na działanie

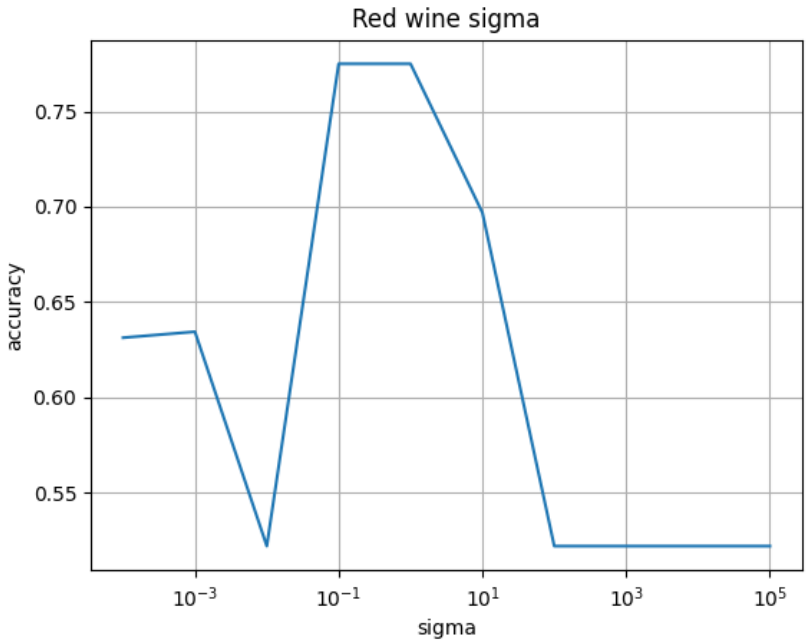
4.2.1 Jądro liniowe

Algorytm osiąga najlepsze rezultaty dla $C = 10^4$.

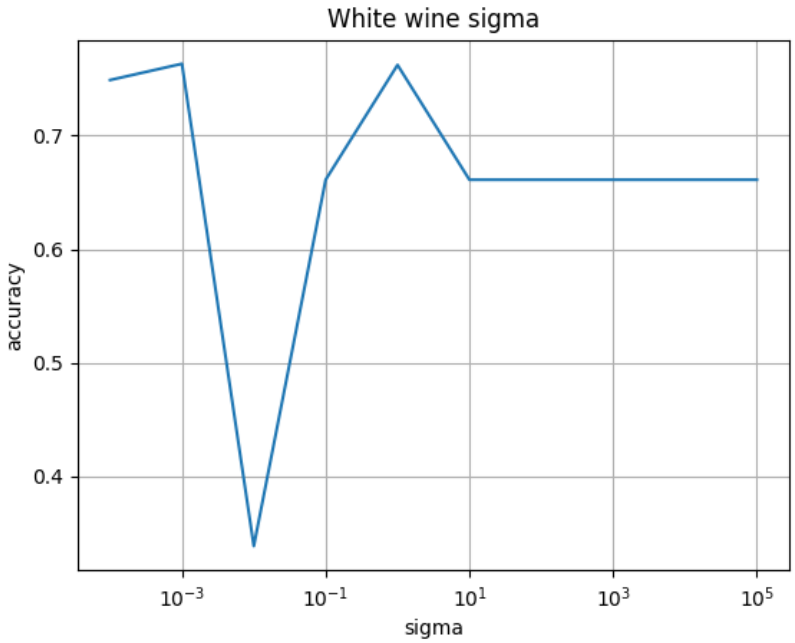
4.2.2 Jądro RBF

Przy stałym σ algorytm osiąga najlepsze rezultaty dla $C = 10^3$. Następnie sprawdzam dla tej wartości wpływ parametru σ.

$$\sigma \in \{10^{-4}, 10^{-3}, ..., 10^4, 10^5\}$$



Rysunek 3: Wpływ parametru σ na skuteczność dla zbioru win czerwonych



Rysunek 4: Wpływ parametru σ na skuteczność dla zbioru win białych

Przy stałym $C = 10^3$ algorytm osiąga najlepsze rezultaty dla $\sigma = 10^0 = 1$.

5 Wnioski

Zastosowanie algorytmu SVM do problemu klasyfikacji binarnej przynosi zadowalające rezultaty (ponad 75% skuteczności) nawet gdy dane nie są liniowo separowalne. Algorytm ten wymaga odpowiedniego doboru hiperparametrów, aby jego skuteczność była jak największa. Każdy model wymaga innego doboru parametrów. Przeprowadzenie większej liczby eksperymentów i bardziej dokładny dobór parametrów mogłoby jeszcze zwiększyć dokładność SVM. Podczas przeprowadzania doświadczeń zauważyłem również, że złożoność czasowa algorytmu zależy od wielkości zbioru danych oraz użytej funkcji jądrowej. Dla mniejszego zbioru tj. wina czerwone algorytm działał szybciej. Jądro liniowe było prostsze do obliczenia niż jądro RFB, co również miało wpływ na czas wykonywania algorytmu.

Literatura

[1] Tristan Fletcher (2009) *Support vector machines explained.*