Metody Analizy Danych
Support Vector Machine

dr inż. Marcin Luckner
mluckner@mini.pw.edu.pl

Wersja 1.1
22 kwietnia 2022

Metoda wektorów nośnych

- Support Vector Machine (SVM) w literaturze polskiej:
  - metoda wektorów nośnych [Krzyśko et al., 2008],
  - metoda wektorów podpierających [Koronacki and Ćwik, 2006],
  - metoda wektorów wspierających [Jankowski, 2003],
  - metoda wektorów podtrzymujących [Osowski, 2006].

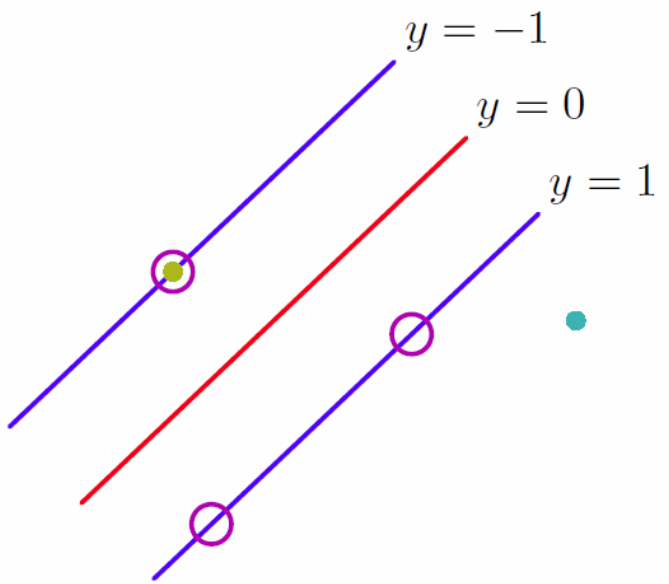
Założenia

- SVM jest binarną metodą dyskryminacji uczenia maszynowego.
- Zbiór uczący składa się z par obserwacja, etykieta

$$\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$$

  y_i jest etykietą klasy przypisaną do wektora cech $\boldsymbol{x}_i$.
- Etykiety należą do zbioru $\{-1, 1\}$

Zadanie



Rysunek 1: Optymalna hiperpłaszczyzna [Bishop, 2006]

- Należy stworzyć hiperpłaszczyznę dzielącą w sposób optymalny, czyli zachowując maksymalny margines między przedstawicielami odmiennych klas, przestrzeń danych.

Model liniowy

- W klasyfikacji wykorzystuje się funkcję f:

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}^T \boldsymbol{x}_i + b,$$

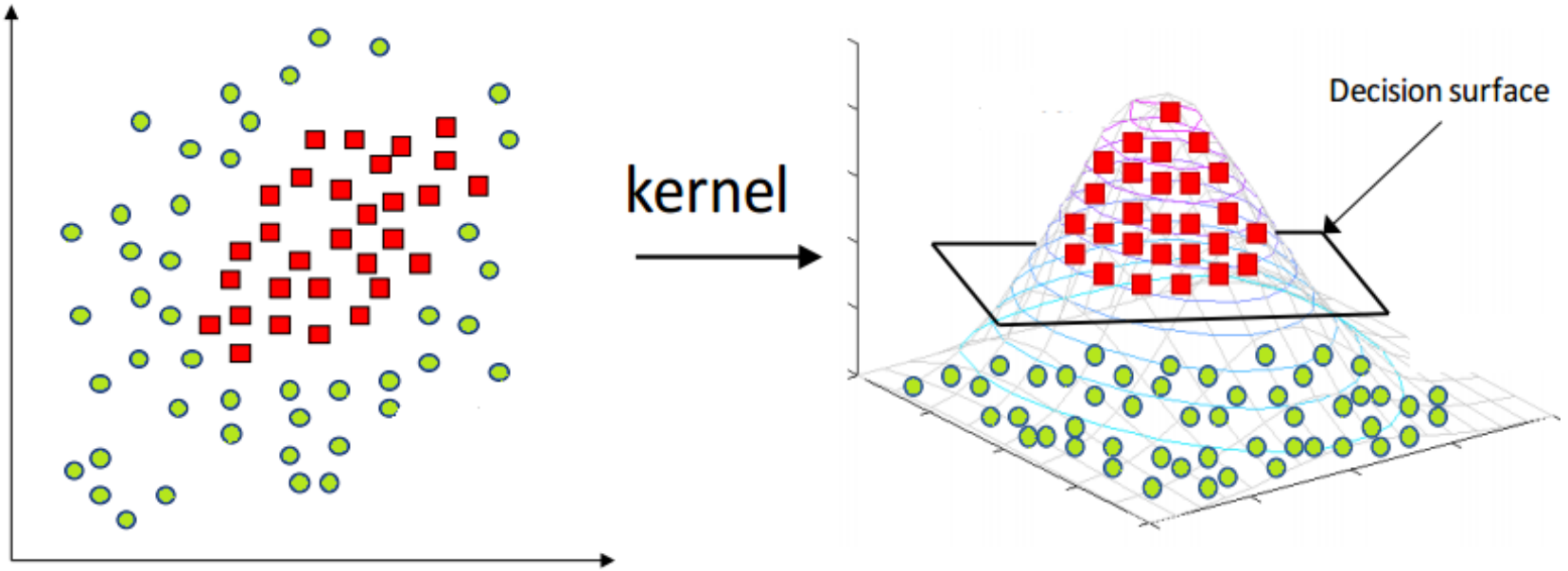
  N jest licznością zbioru uczącego, a parametry α_i i b są wyliczane w procesie uczenia.
- Funkcja f zwraca odległość punktu x od granicy podziału i decyzję o zaklasyfikowaniu przykładu podejmuje się na podstawie jej znaku $\text{sgn}(f(x))$.

Model jądrowy

- Zadanie można przeformułować tak, aby podział dotyczył innej przestrzeni, dla której odnalezienie podziału liniowego będzie łatwiejsze.
- W tym celu należy zastąpić iloczyn skalarny dotyczący przestrzeni podstawowej, iloczynem skalarnym funkcji bazowych nowej przestrzeni, czyli funkcją jądrową K:

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i K(\boldsymbol{x_i}, \boldsymbol{x}) + b.$$

Sztuczka jądrowa



Rysunek 2: Sztuczka jądrowa [Sharma, 2019]

- Nieseparowane dane stają się separowane gdy zwiększymy wymiar przestrzeni danych

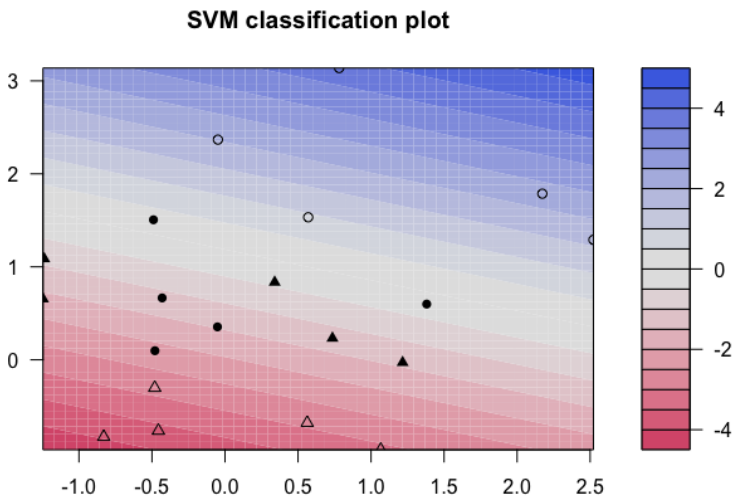
Funkcje jądrowe

- Popularnymi funkcjami jądrowymi dla których $K(\boldsymbol{x_i}, \boldsymbol{x}) = K(\boldsymbol{x_i})K(\boldsymbol{x})$ są:

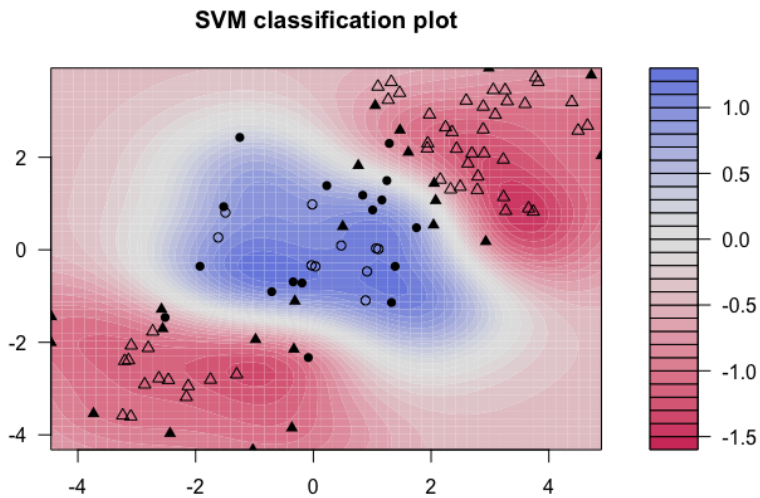
$$\begin{aligned}
\text{Liniowa:} &\quad \boldsymbol{x}^T \boldsymbol{x}_i, \\
\text{Wielomianowa:} &\quad \left(\gamma \boldsymbol{x}^T \boldsymbol{x}_i + c\right)^n, \\
\text{RBF:} &\quad \exp\left(-\gamma \|\boldsymbol{x} - \boldsymbol{x}_i\|^2\right), \\
\text{Sigmoidalna:} &\quad \operatorname{tgh}\left(\gamma \boldsymbol{x}^T \boldsymbol{x}_i + c\right).
\end{aligned}$$

- Należy zauważyć, że oprócz liniowej wszystkie funkcje jądra wprowadzają dodatkowe parametry.

Wpływ jądra na podział



Rysunek 3: Jądro liniowe



Rysunek 4: Jądro radialne

[UC Business Analytics R Programming Guide, 2000]

Wyliczanie rozdzielającej hiperpłaszczyzny

- Hiperpłaszczyznę charakteryzują wektor współczynników zmiennych ψ i wyraz wolny b.
- Dla ich wyznaczenia należy zmaksymalizować wyrażenie:

$$L(\boldsymbol{\psi}, \boldsymbol{b}, \alpha) = \frac{1}{2}(\boldsymbol{\psi} * \boldsymbol{\psi}) - \sum_{i=1}^{n} \alpha_i (y_i((\boldsymbol{x}_i * \boldsymbol{\psi}) + \boldsymbol{b}) - 1),$$

  gdzie $\alpha_i \geqslant 0$ są mnożnikami Lagrange'a.
- W tym celu należy rozwiązać zadanie optymalizacji programowania kwadratowego.

Warunki punktu siodłowego

- Warunki Karusha–Kuhna–Tuckera, które muszą być spełnione dla rozwiązania optymalnego, dają zależności

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \text{ i } \boldsymbol{\psi}^0 = \sum_{i=1}^{n} y_i \alpha_i \boldsymbol{x}_i$$

- Po podstawieniu uzyskujemy funkcję

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i^T \boldsymbol{x}_j)$$

która minimalizujemy przy warunkach

$$(\forall i) \alpha_i \geqslant 0$$

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

Interpretacja wektorów nośnych

- Dla rozwiązania optymalnego

$$\boldsymbol{\psi}^0 = \sum_{i=1}^{n} y_i \alpha_i^0 \boldsymbol{x}_i$$

- Jednak dla wektorów innych niż nośne $\alpha^0 = 0$ stąd optymalna hiperpłaszczyzna to

$$\sum_{\text{wektory nośne}} y_i \alpha_i^0 \boldsymbol{x}_i \boldsymbol{x} + \boldsymbol{b}^0 = 0$$

Wyznaczanie b_0

- Współczynnik b, przesunięcie hiperpłaszczyzny, łatwo wyliczyć na podstawie płaszczyzn wyznaczanych przez wektory nośne.

$$b^0 = \frac{1}{2}[((\psi^0)^T x^1) + ((\psi^0)^T x^{-1})]$$

gdzie x^{-1} i x^1 są dowolnymi wektorami nośnymi klas -1 i 1.

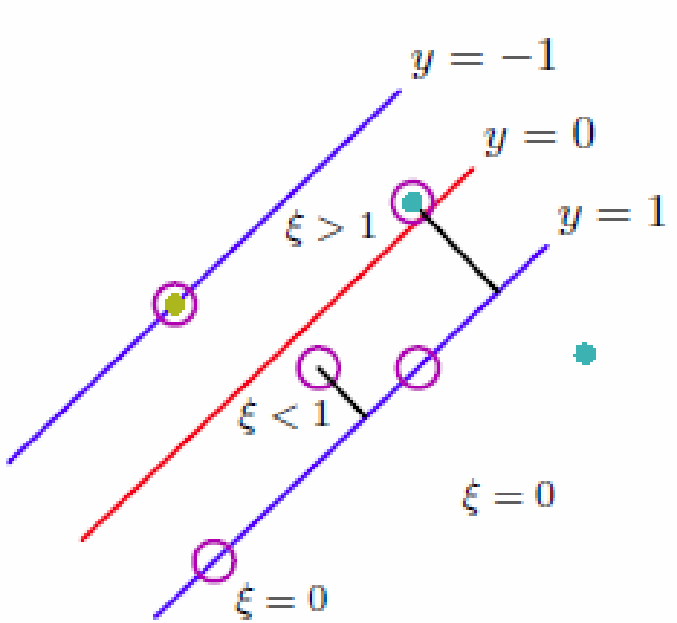
Rozluźnienie warunków

- Nawet po wprowadzeniu funkcji jądrowych rozwiązanie zadania może nie istnieć, ze względu na brak możliwości liniowego odseparowania klas.
- Przekształcona postać zadania umożliwia znalezienie rozwiązania przybliżonego

$$L(\boldsymbol{\psi}, \boldsymbol{b}, \alpha, \mu) = \frac{1}{2}(\boldsymbol{\psi} * \boldsymbol{\psi}) + C \sum_{i=1}^{N} \xi_i + \\ - \sum_{i=1}^{N} \alpha_i (y_i((\boldsymbol{x}_i * \boldsymbol{\psi}) + b) - 1 + \xi_i) - \sum_{i=1}^{N} \mu_i \xi_i.$$

- Współczynniki ξ_i rozluźniają nierówności ograniczające dla wybranych wektorów x_i, a μ_i są mnożnikami Lagrange.

Przykład

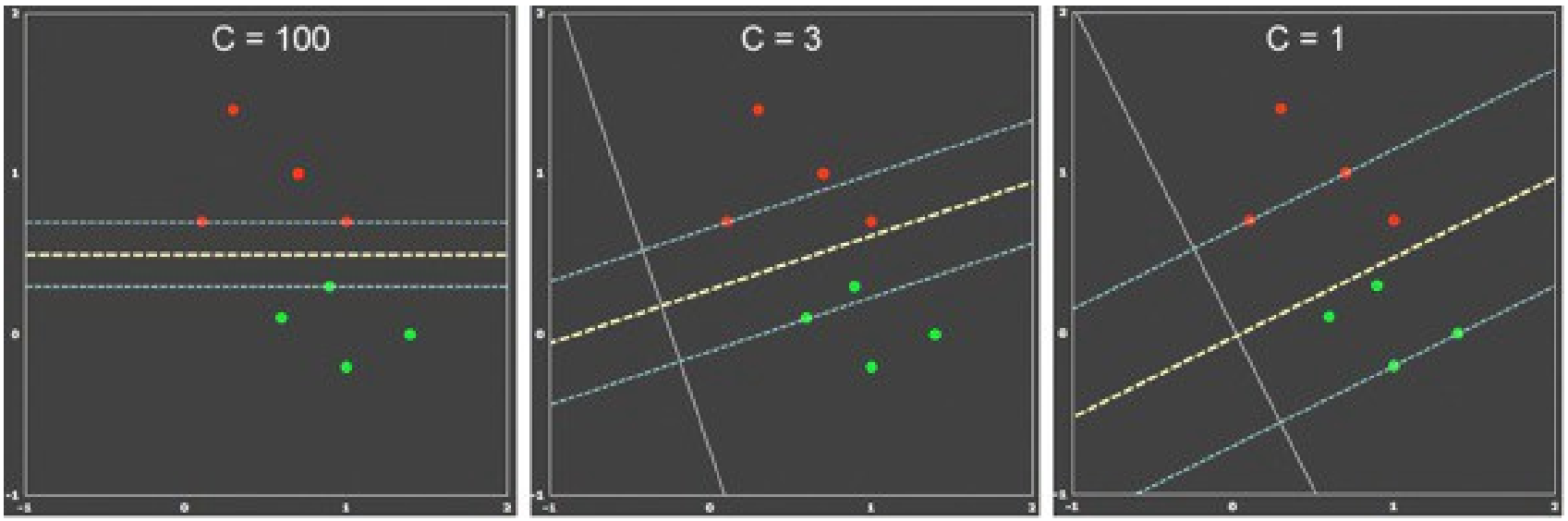


Rysunek 5: Przykłady różnych ξ_i [Bishop, 2006]

Współczynnik C

- Współczynnik C określa wymiar kary za rozluźnienie warunków rozwiązania optymalnego.
- Pozwala sterować stosunkiem szerokości marginesu między klasami, czyli zdolnością do generalizacji, a liczbą błędów w zbiorze uczącym.
- Ponieważ jego dobór nie jest intuicyjny pojawiły się rozwiązania które próbują zastąpić go innymi parametrami lub całkowicie wyeliminować.

Wpływ współczynnika C



Rysunek 6: Wpływ współczynnika C na margines [Mandot, 2017]

Zadania wieloklasowe

- Metoda SVM daje bardzo dobre rezultaty klasyfikacji, ale jest dedykowana do zadań binarnych (klasyfikacja dwóch klas).
- Istnieją modyfikacje zadania optymalizacji budujące podział dla wielu klas, ale nie nadają się do praktycznego zastosowania.
- W praktyce tworzy się zespół klasyfikatorów binarnych, które tworzą układ do rozwiązywania zadań wieloklasowych.

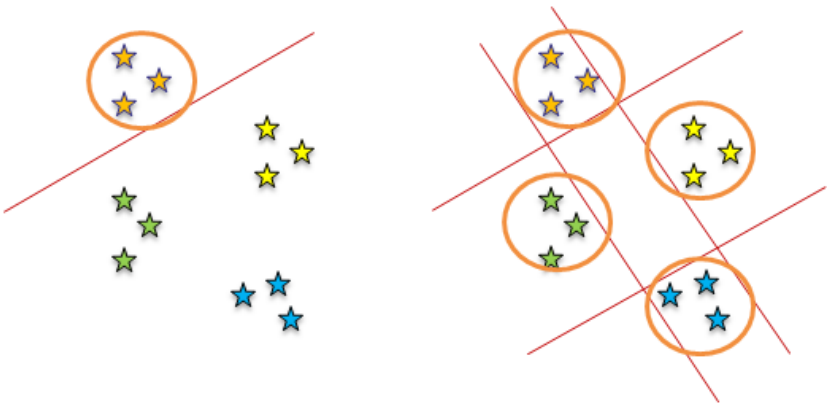
Jeden kontra wszyscy

- Zadanie z $K > 2$ klasami rozkładamy na K zagadnień binarnych.
- W każdym zagadnieniu wybieramy jedną klasę (próbie uczącej dla tej klasy przypisujemy etykietę 1), a pozostałe $K - 1$ klas tworzy super klasę (obserwacjom z tych klas przypisujemy 0).

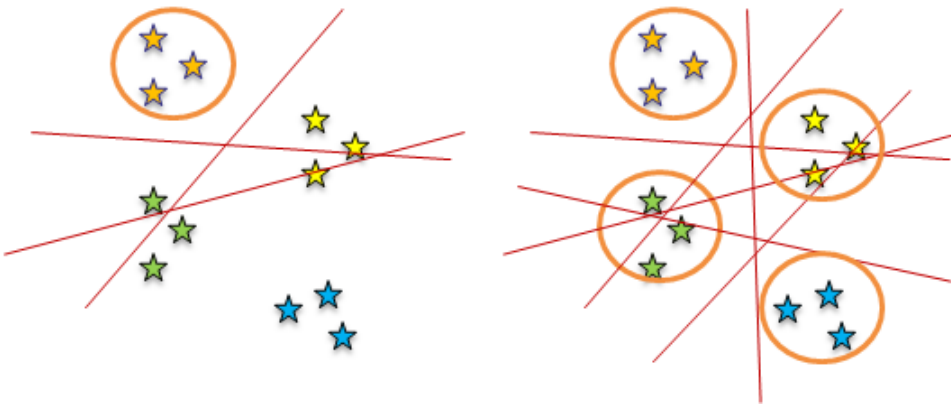
Jeden kontra jeden

- Zadanie z $K > 2$ klasami rozkładamy na $L = \frac{K(K-1)}{2}$ zagadnień binarnych.
- W każdym zagadnieniu wybieramy pomiędzy jedną z dwóch klas, jedna otrzymuje etykietę 1, a druga 0.
- Rozpatrujemy wszystkie możliwe pary.

Porównanie działania metod JkJ i JkW



Rysunek 7: Metoda JkW, krok dla pierwszej klasy i końcowy podział



Rysunek 8: Metoda JkJ, krok dla pierwszej klasy i końcowy podział

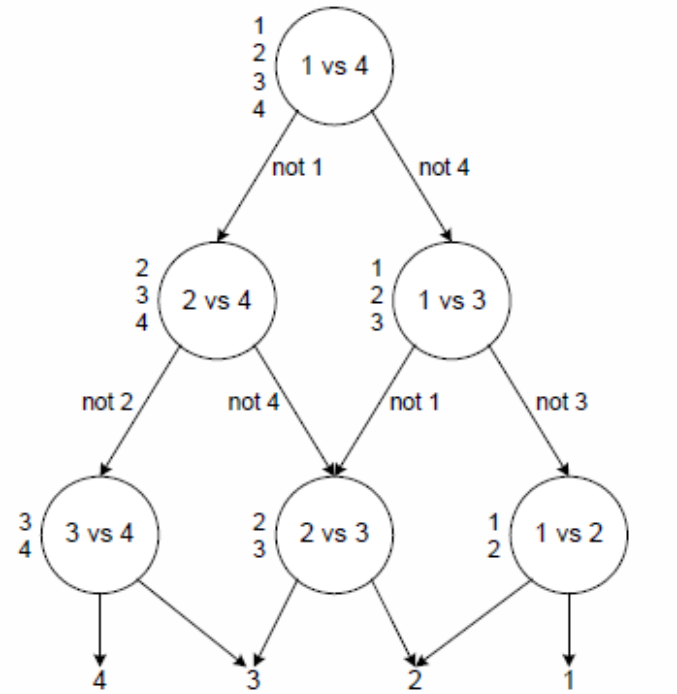
Przykłady ograniczeń metod

Rysunek 9: Metoda jeden kontra wszyscy tworzy zbyt mało podziałów, aby przeprowadzić liniową separację. Metoda jeden kontra jeden buduje nadmiarowe klasyfikatory.

Algorytm DAGSVM

- Algorytm DAGSVM łączy klasyfikatory SVM w zespół za pomocą niecyklicznego grafu skierowanego (ang. *Directed Acyclic Graph*).
- Pierwszy klasyfikator dokonuje wyboru jednej z dwóch klas. W węzłach będącymi jego potomkami wygrany rywalizuje z następnym kandydatem z listy. Kolejność w jakiej porównywane są klasy jest losowa.

przykład DAGSVM



Rysunek 10: Platt (2000)

Glass Identification Database

- Zbiór ten służy identyfikacji szkła. Motywacją jego powstania było śledztwo kryminalne. Szkło pozostawione na miejscu zbrodni może zostać użyte jako dowód, oczywiście, jeśli zostanie poprawnie rozpoznane.
- Dane te powstały z analizy 214 próbek szkła. Każda próbka w zbiorze jest reprezentowana przez 10 różnych atrybutów + atrybut klasy.
- Wyróżniono 7 rodzajów szkła, jednak jeden z nich nie występuje we zbiorze.

Wyniki dla różnych parametrów

Rysunek 11: $C = 10, \gamma = 10$ (30/66)

	1	2	3	5	6	7
1	18	21	4	5	4	5
2	1	10	1	0	0	0
3	0	0	1	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	0	0	0	0	0	1

Rysunek 12: $C = 10, \gamma = 0.001$ (28/66)

	1	2	3	5	6	7
1	19	27	6	2	3	1
2	0	4	0	3	0	0
3	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	0	0	0	0	1	5

Wyniki dla różnych funkcji jądra

Rysunek 13: RBF $C = 10, \gamma = 10$ (30/66)

	1	2	3	5	6	7
1	18	21	4	5	4	5
2	1	10	1	0	0	0
3	0	0	1	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	0	0	0	0	0	1

Rysunek 14: liniowe $C = 10$ (48/66)

	1	2	3	5	6	7
1	15	6	4	0	0	0
2	4	20	2	1	0	1
3	0	0	0	0	0	0
5	0	2	0	4	0	0
6	0	2	0	0	4	0
7	0	1	0	0	0	5

Porady

- SVM jest bardzo wrażliwy na dobór parametrów.
- W praktyce nie ma innej metody jak przeszukanie przestrzeni parametrów w poszukiwaniu najlepszego rozwiązania.
- Problem klasyfikacyjny zacznij rozwiązywać minimalizując liczbę parametrów. Zacznij od modelu liniowego, później RBF, przy braku dobrych wyników użyj innych funkcji jądra.
- Przeskalowanie danych wpływa istotnie na wynik. Funkcja svm w R domyślnie normalizuje dane.
- Czas trenowania SVM zależy od rozmiaru zbioru uczącego. Lepiej dobrać reprezentatywny podzbiór niż uczyć na zbliżonych obserwacjach.

Dobór parametrów

Zalecana procedura doboru parametrów dla modelu z jądrem RBF wygląda następująco.

1. Badaj wyniki dla różnych wartości C z przedziału $[1, 1000]$ (np. $\{2^0, 2^1, \ldots . 2^{12}\}$).
2. Wynik klasyfikatora weryfikuj poprzez kroswalidację.
3. Dla ustalonego najlepszego C wykonaj testy na kilku wartościach γ (np. $\{2^{-12}, 2^{-11}, \ldots 2^4\}$).

Lepszym, choć kosztowniejszym, rozwiązaniem jest przeszukanie przestrzeni wszystkich par.

Wyniki po optymalizacji parametrów

Rysunek 15: RBF $C = 2, \gamma = 0.0625$ (50/66)

	1	2	3	5	6	7
1	18	6	2	0	1	0
2	1	22	4	2	0	1
3	0	0	0	0	0	0
5	0	3	0	3	0	0
6	0	0	0	0	2	0
7	0	0	0	0	1	5

Rysunek 16: liniowe $C = 10$ (48/66)

	1	2	3	5	6	7
1	15	6	4	0	0	0
2	4	20	2	1	0	1
3	0	0	0	0	0	0
5	0	2	0	4	0	0
6	0	2	0	0	4	0
7	0	1	0	0	0	5

Podsumowanie

- SVM jest bardzo dobrym klasyfikatorem binarnym, który potrafi znajdować rozwiązania dyskryminacyjne dla przypadków liniowo nieseparowalnych.
- W przypadku rozluźnienia parametrów zadania optymalizującego znajduje wyniki przybliżone w rozsądnym czasie.
- Dla wyników kluczowy jest dobór parametru C i parametrów jądra.
- W wypadku rozwiązywania zadań wieloklasowych należy skorzystać z zestawu klasyfikatorów binarnych.

Bibliografia I

[Bishop, 2006] Bishop, C. M. (2006).

Pattern recognition and machine learning.

Springer.

[Jankowski, 2003] Jankowski, N. (2003).

Ontogeniczne sieci neuronowe: o sieciach zmieniających swoją strukturę.

Akademicka Oficyna Wydawnicza EXIT.

[Koronacki and Ćwik, 2006] Koronacki, J. and Ćwik, J. (2006).

Statystyczne systemy uczące się.

Akademicka Oficyna Wydawnicza EXIT.

[Krzyśko et al., 2008] Krzyśko, M., Wołyński, W., Górecki, T., and Skorzybut, M. (2008).

Systemy uczące się. Rozpoznawanie wzorców analiza skupień i redukcja wymiarowości.

WNT.

Bibliografia II

[Mandot, 2017] Mandot, P. (2017).

What is the significance of c value in support vector machine?

[Osowski, 2006] Osowski, S. (2006).

Sieci neuronowe do przetwarzania informacji.

Oficyna Wydawnicza Politechniki Warszawskiej.

[Sharma, 2019] Sharma, S. (2019).

Kernel trick in svm.

[UC Business Analytics R Programming Guide, 2000] UC Business Analytics R Programming Guide (2000).

Support vector machine.