Maszyny wektorów nośnych. Teoria i zastosowania. część 2: Zastosowania

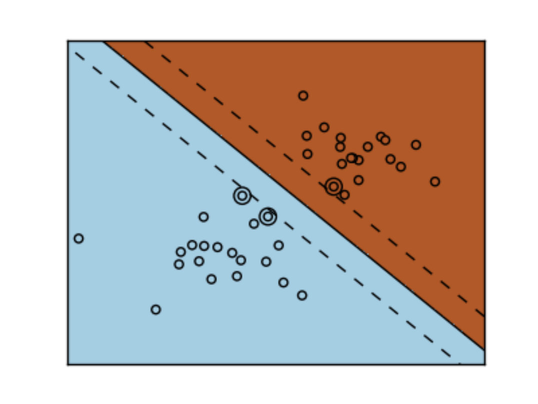
mgr Agnieszka Pocha, mgr Maciej Brzeski

Wydział Matematyki i Informatyki UJ,

Katedra Metod Uczenia Maszynowego

Kraków, 19 stycznia 2016

Przypomnienie



$$w^T x + b = 0$$

$$✵\ sgn(w^T x_{new} + b) = \begin{cases} +1 \\ -1 \end{cases}$$

☞ $w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$

α_i ➞ wyliczenie ze wzorku wyprowadzonego na poprzednim seminarium:

$\mathscr{L}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{j=1,k=1}^{m} y^{(j)} y^{(k)} \alpha_j \alpha_k (x^{(j)})^T x(k)$

będziemy szukać $\max_{\alpha} \mathscr{L}(a)$ przy ograniczeniach:

- $\alpha_i \geqslant 0$
- $\sum_i \alpha_i y^{(i)} = 0$

Używamy w tym celu algorytmu SMO.

A można prościej!

Podstawmy ☞ do ✺ i otrzymamy: $sgn((\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)})^T x + b)$

x - to tutaj nowy punkt, który chcielibyśmy sklasyfikować, $x^{(i)}$ - kolejne punkty ze zbioru uczącego

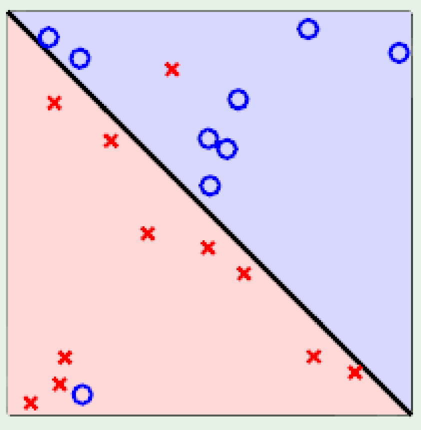
zatem pytamy jaki znak miałoby poniższe wyrażenie:

✫ $\sum_{i} \alpha_i y^{(i)} < x^{(i)}, x > +b$

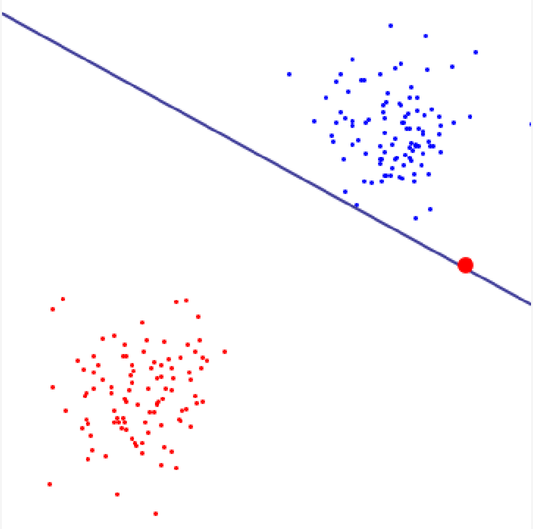
zauważmy, że w tym wzorze nie ma w ogóle w, zatem nie musimy go wyliczać!

$x^{(i)}$ - to są właśnie tytułowe wektory nośne

Dane liniowo nieseparowalne.



Overfitting.



Funkcja kosztu.

$$\min_{w,b,\xi_i} \frac{1}{2}\|w\|^2 + c\sum_{i=1}^{m} \xi_i,$$

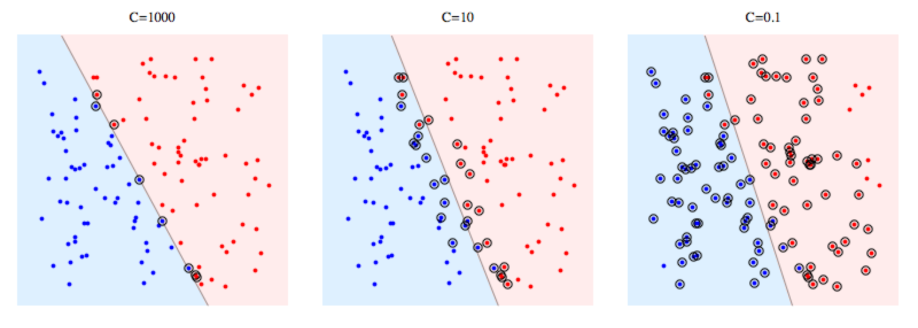
pod warunkami:

- $y^{(i)}[w^T x^{(i)} + b] \geqslant 1 - \xi_i$
- $\xi_i \geqslant 0$

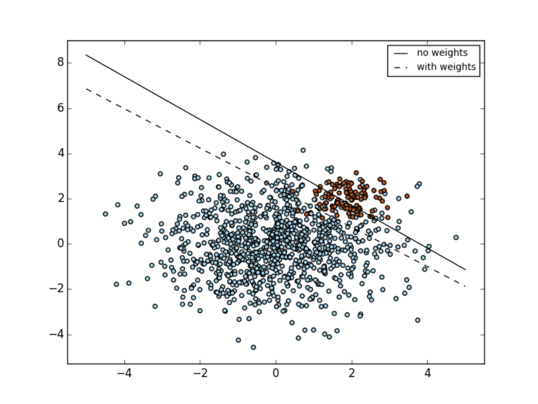
gdzie:

- ξ_i - koszt przydzielenia i-temu punktowi wartości $y^{(i)}$. Dla dobrze sklasyfikowanych punktów leżących poza marginesem wynosi 0, dla pozostałych jest równa odległości od marginesu.
- $c > 0$ - stała regularyzacji, przypisując jej konkretną wartość określamy, jak bardzo chcemy karać klasyfikator za błędną klasyfikację.

Parametr c. Ilustracja



Niezbalansowane zbiory.



R. Batuwita, V. Palade, **Imbalanced Learning: Foundations, Algorithms, and Applications**, Haibo He and Yungian Ma (Eds.), Wiley, 2013, rozdział 6

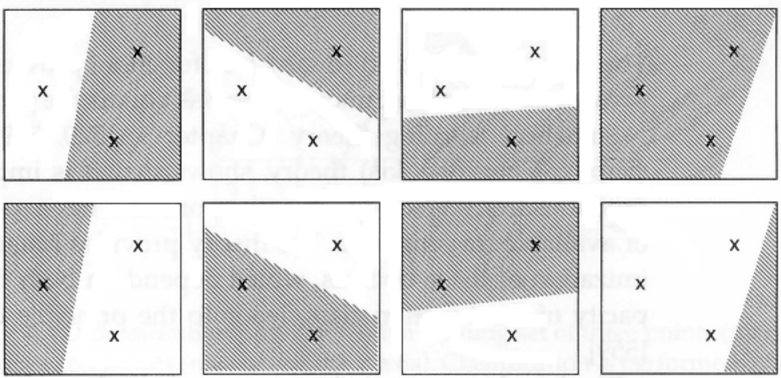
Wymiar Vapnika-Chervonenkisa

Wyobraźmy sobie taką grę:

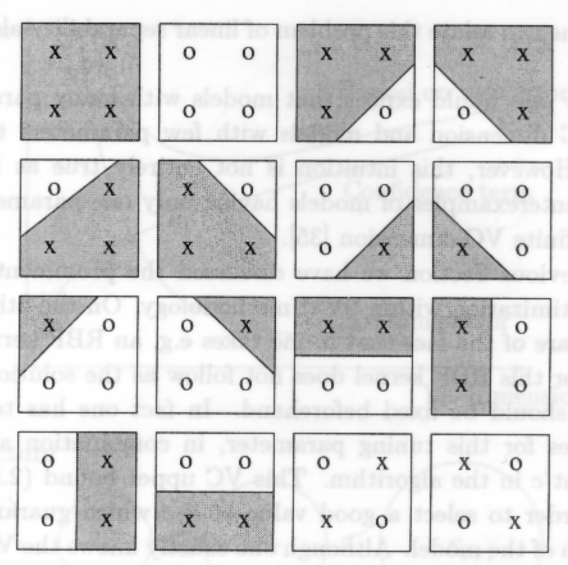
1. dwaj gracze ustalają liczbę punktów oraz sposób rozdzielania ich
2. gracz A ustala położenie punktów
3. gracz B nadaje im etykiety (w miarę możliwości złośliwie)
4. gracz A usiłuje rozdzielić punkty tak, by te o jednej etykiecie były po jednej stronie, a te o drugiej etykiecie po drugiej

Wygrywa gracz A jeśli uda mu się rozdzielić punkty. Wygrywa gracz B, jeśli graczowi A się nie powiedzie.
Wymiar V-C to maksymalna liczba punktów, taka że A wygra niezależnie od poetykietowania zadanego przez B.

Wymiar Vapnika-Chervonenkisa. Przykład sukcesu.



Wymiar Vapnika-Chervonenkisa. Przykład porażki.



Wymiar VC. Do czego się przydaje?

$$P(test\_error \leqslant training\_error + \sqrt{\frac{h(\log(\frac{2N}{h})+1)-\log\frac{\eta}{4}}{N}}) \geqslant 1-\eta$$

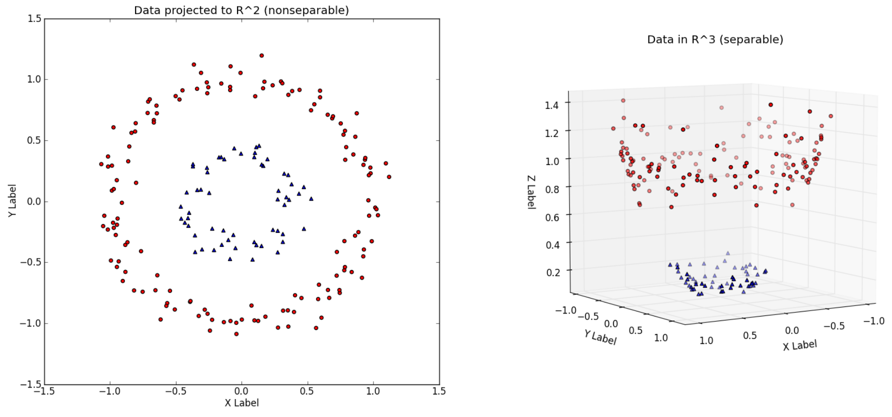
- *test_error* - błąd na zbiorze testowym
- *training_error* - błąd na zbiorze trenującym
- h - wymiar VC
- $0 < \eta < 1$
- N - wielkość zbioru trenującego

Ważne:

- $h << N$.
- przykłady w zbiorze trenującym i testującym są wybierane niezależnie od siebie i z tego samego rozkładu

Kernel trick. Motywacja

$$x \rightarrow \phi(x)$$



`http://rvlasveld.github.io/images/oc-svm/visualization.gif`

Kernel trick w SVM

Podstawmy zatem zamiast $< x^{(i)}, x >$ we wzorze $\star$ $< \phi(x^{(i)}), \phi(x) >$ a otrzymamy:

$$\sum_i \alpha_i y^{(i)} < \phi(x^{(i)}), \phi(x) > +b$$

Wyliczenie $\phi(x)$ może być trudne/czasochłonne/niemożliwe, ale okazuje się, że policzenie $< \phi(x^{(i)}), \phi(x) >$, jest znacznie łatwiejsze/szybsze/w ogóle możliwe.

Kernel trick. Jak to robimy?

Wiemy, że $< \phi(x), \phi(y) >$, to jest po prostu $\phi(x)^T\phi(y)$. Definiujemy funkcję (nazywamy ją jądrem (ang. kernel)):

$$K(x, y) := \phi(x)^T\phi(y)$$

✭ ma teraz postać:

$$\sum_i \alpha_i y^{(i)} K(x^{(i)}, x) + b$$

Kernel trick. Magia.

Pomysł jest taki, aby w ogóle nie definiować $\phi(x)$, tylko zdefiniować od razu $K(x, y)$. Powstaje pytanie: jak powinna wyglądać funkcja K? Chcielibyśmy, by K było pewną miarą "podobieństwa", tzn. jeżeli $\phi(x)$ i $\phi(y)$ leżą blisko siebie (są podobne), to $K(x, y)$ powinno być duże, natomiast jeżeli $\phi(x)$ i $\phi(y)$ leżą daleko od siebie (są niepodobne), to $K(x, y)$ powinno być bliskie 0.
Tak jest np. gdy mamy do czynienia z jądrem gaussowskim:

$$K(x, y) = exp(-\frac{\|x-y\|^2}{2\sigma^2})$$

Kernel trick. Twierdzenie Mercera.

Niech X - zbiór jakichś punktów oraz $x^{(i)}$ to i-ty punkt w X.
Zdefiniujmy macierz $\mathscr{K}$ w taki sposób:

$$\forall i, j \; \mathscr{K}_{ij} = K(x^{(i)}, x^{(j)})$$

Jeżeli $\mathscr{K}$ jest nieujemnie określona oraz symetryczna, to K jest jądrem (tzn. istnieje $\phi(x)$, takie że $\forall x, y \; < \phi(x), \phi(y) >= K(x, y)$).

To the batmobile!

Dlaczego SVMy są fajne?

- M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim; **Do We Need Hundreds of Classifiers to Solve Real World Classification Problems?**; J. Mach. Learn. Res.; January 2014;
- szybko się uczą. naprawdę szybko.
- "można" je douczać (online learning), są to tzw. incremental SVMs
- mają (stosunkowo) mało parametrów
- są dosyć odporne na overfitting i nie przeraża je klątwa wielowymiarowości
- niski wymiar Vapnika-Chervonenkisa
- mało hyperparamaterów
- dobrze się zachowują na rzadkich danych
- dobrze sobie radzą, jeżeli klasy są niezbalansowane

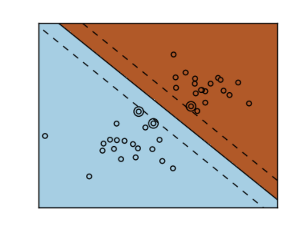
Jakie problemy można rozwiązywać za pomocą SVMów?

- klasyfikacja
- regresja: SV Regression, można poczytać w: A.J. Smola, B. Schölkopf, **A Tutorial on Support Vector Regression**, 1998
- novelty detection: one-class SVM, można poczytać w: B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, J. C. Platt, **Support Vector Method for Novelty Detection**, 1999, NIPS

Przykłady w pythonie.

1. sam proces uczenia - jak zmieniają się wektory nośne/decision boundary
2. jakie są różnice, gdy stosujemy różne kernele
3. multi-class classification
4. rysowanie największego marginesu
5. niezbalansowanie zbiory
6. przykład dla weighted examples
7. regression
8. novelty detection

Podsumowanie



- $w^T x + b = 0$
- małe
- szybkie
- zwinne
- dobrze radzą sobie z większością problemów
- istnieje wiele wariacji SVMów (SVR, on-class SVM, różne kernele)

Bibliografia

1. no wikipedie, bo zawsze warto od nich zacząć
2. C. Bishop, **Pattern Recognition and Machine Learning**, 2006, Springer, rozdziały 6 (Kernels) i 7 (Kernel Models)
3. T. Hastie, R. Tibshirani, J. Friedman, **The Elements of Statistical Learning**, Springer, 2008, rozdział 12
4. Andrew Ng, CS229, Stanford, `http://cs229.stanford.edu/notes/cs229-notes3.pdf` + jest także nagranie z wykładu
5. M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim; **Do We Need Hundreds of Classifiers to Solve Real World Classification Problems?**; J. Mach. Learn. Res.; January 2014;
6. A.J. Smola, B. Schölkopf, **A Tutorial on Support Vector Regression**, 1998

7. B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, J. C. Platt, **Support Vector Method for Novelty Detection**,
8. R. Batuwita, V. Palade, **Imbalanced Learning: Foundations, Algorithms, and Applications**, Haibo He and Yungian Ma (Eds.), Wiley, 2013, rozdział 6
9. `http://rvlasveld.github.io/blog/2013/07/12/introduction-to-one-class-support-vector-machines/`
10. przykłady w pythonie i pożyteczne intuicje są na scikit-learn
11. `http://stackoverflow.com/a/4630731`

Obrazki

1. http://scikit-learn.org/stable/_images/plot_svm_margin_0011.png
2. http://www.blaenkdenum.com/images/notes/machine-learning/kernel-methods/slightly-non-separable.png
3. http://yaroslavvb.com/upload/save/so-svm.png
4. http://yaroslavvb.com/upload/save/so-libsvm.png
5. http://scikit-learn.org/stable/_images/plot_separating_hyperplane_unbalanced_001.png
6. http://www.svms.org/vc-dimension/ScholkopfSmola2002_1-4.png
7. http://www.svms.org/vc-dimension/Suykens-etal2002_2-9.png
8. http://www.eric-kim.net/eric-kim-net/posts/1/imgs/data_2d_to_3d.png

Dziękuję za uwagę.
Slajdy i kody można znaleźć pod
`https://github.com/Lamiane/presentations/tree/master/2016-01-SVMs_sem_matematyka_stosowana`