Metody Inżynierii Wiedzy
Klasyfikacja - algorytmy KNN, regresja logistyczna - wykład 4

Adam Szmigielski
aszmigie@pjwstk.edu.pl
materiały: $ftp(public) : //aszmigie/MIW$

Zadanie klasyfikacji

Zadanie klasyfikacji polega na przydzielaniu przykładu do jednej z rozłącznych klas.

- *klasyfikacja binarna* - w przypadku gdy wszystkie dane podzielone są na dwie klasy,
- *klasyfikacja wieloklasowa* - w przypadku gdy istnieje więcej niż dwie klasy.

Technika jeden przeciw reszcie OvR (one versus rest)

Jest stosowana stosowana na rozszerzenie klasyfikacji binarnej na problemy wieloklasowe:

- możemy użyć jeden klasyfikator na klasę,
- Klasa traktowana jest jako pozytywna, a próbki z pozostałych klas jako negatywne,
- Do klasyfikacji danych wykorzystuje się n-klasyfikatorów i przydziela się etykietę klas o największej pewności dla danej próbki.

Grupowanie

Zadanie grupowania wymaga wyboru metryki (miary odległości pomiędzy przykładami)

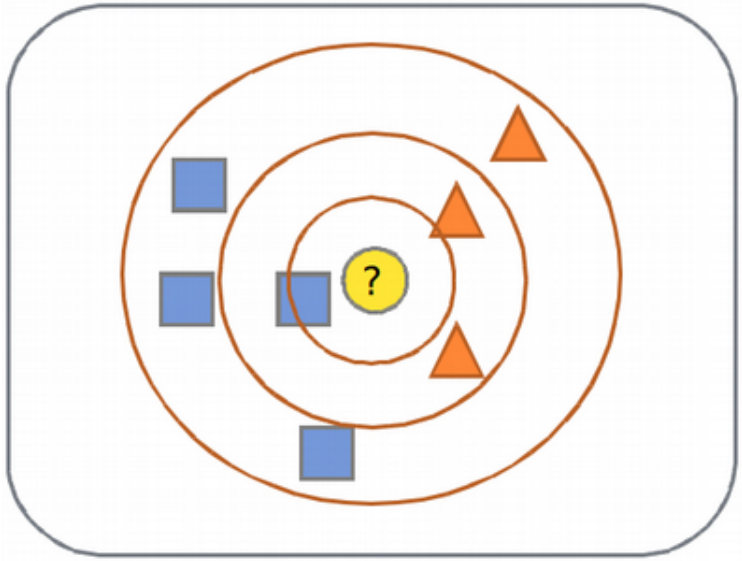
W zadaniach podziału na klasy istotne jest rozwiązanie problemów:

- Problem liczby klas,
- Problem dyskryminacji - podziału danych na klasy.

Wybór liczby klas K

- Właściwy dobór parametru K stanowi podstawę w uzyskaniu równowagi pomiędzy przetrenowaniem i zbyt małym dopasowaniem.
- Wybierana metryka odległości powinna być dopasowana do cech zestawu danych.

Algorytm k-najbliższych sąsiadów — model leniwego uczenia

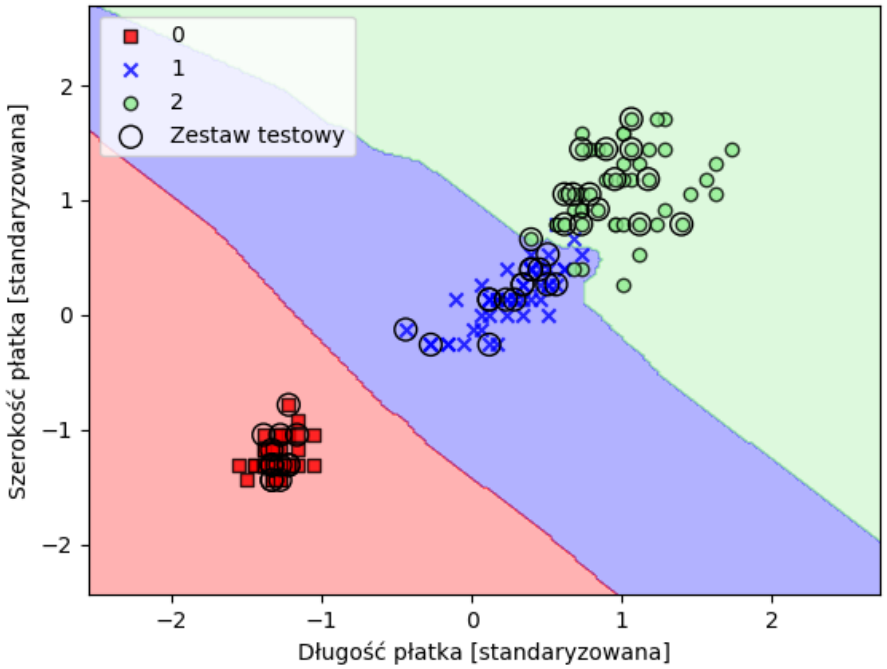


- $k = 1$:
  - Należy do klasy kwadrat
- $k = 3$:
  - Należy do klasy trójkąt
- $k = 7$:
  - Należy do klasy kwadrat

Algorytm KNN:

1. Wybierz jakąś wartość parametru k i metrykę odległości.
2. Znajdź k najbliższych sąsiadów próbki, którą chcesz sklasyfikować.
3. Przydziel etykietę klasy poprzez głosowanie większościowe.

Algorytm KNN - przykład dla K = 5 sąsiadów



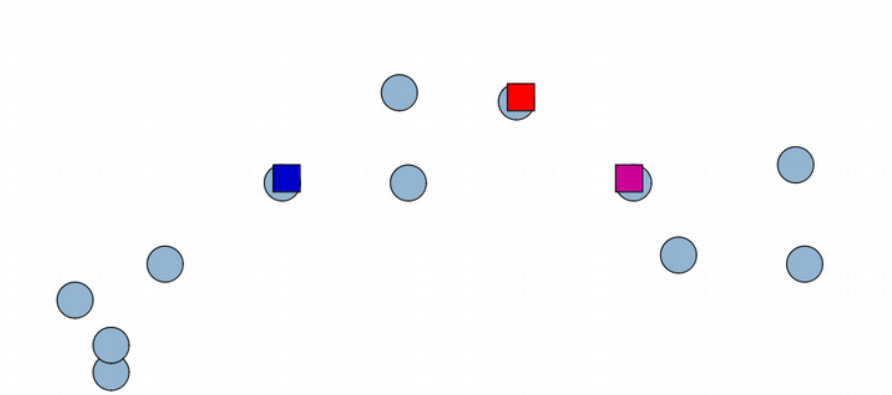
- Algorytm KNN należy do modeli nieparametrycznych,
- Algorytm KNN nie uczy się on funkcji dyskryminacyjnej na podstawie danych uczących, lecz stara się "zapamiętać" cały zbiór próbek.

Algorytm K-means

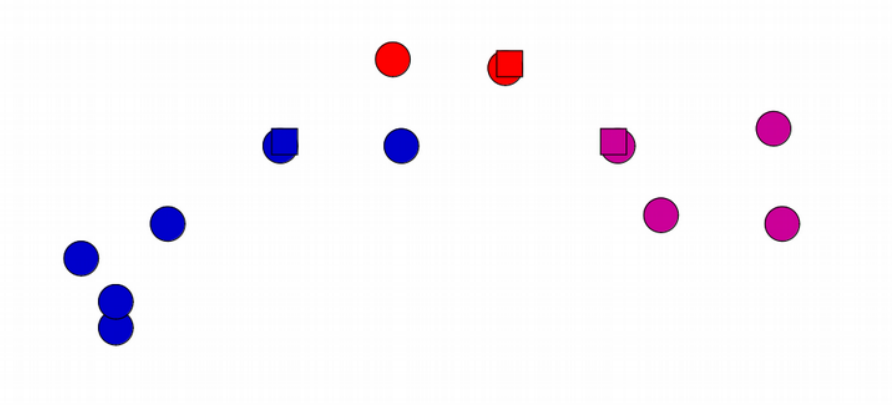
- **Krok 1:** Wybierz liczbę klastrów K,
- **Krok 2** Wybierz dowolnie K punktów, które będą K środkami ciężkości,
- **Krok 3:** Dla Wszystkich punktów oblicz odległości do K środków ciężkości.
- **Krok 4:** Określ przynależność wszystkich punktów do jednej z K grup. Punkt należy do tej grupy do której odległość do środka ciężkości jest najmniejsza. Jeśli żaden z punktów nie zmienił poprzedniej grupy algorytm kończy działanie,
- **Krok 5:** Dla wszystkich K grup punktów oblicz środek ciężkości. Przejdź do *Kroku 3*

Algorytm 3-means przykład

Wybór 3 punktów startowych - kwadraty centroidy

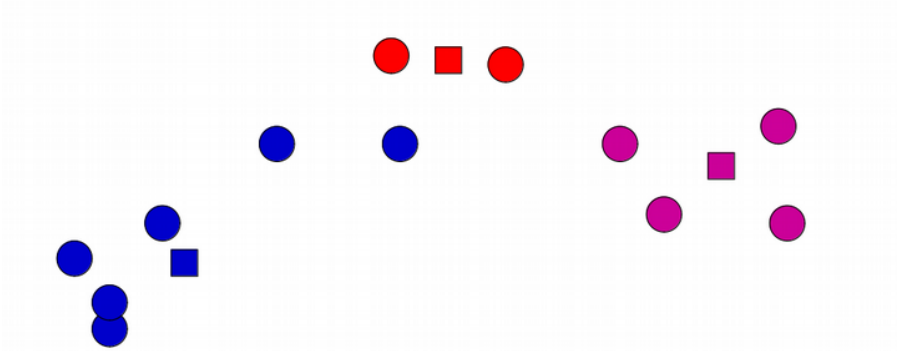


Przydział punktów do klas

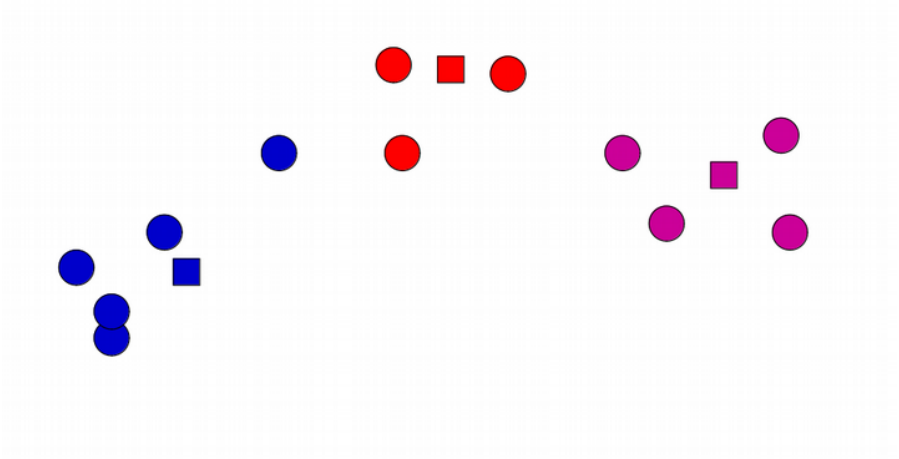


Algorytm 3-means przykład

Wyznaczenie środków ciężkości

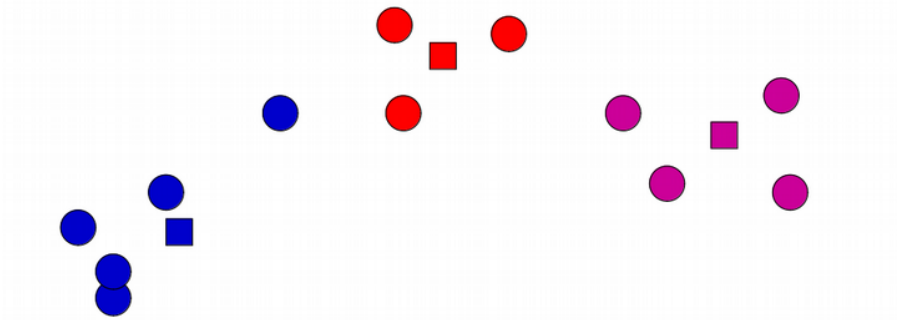


Nowy przydział punktów do klas

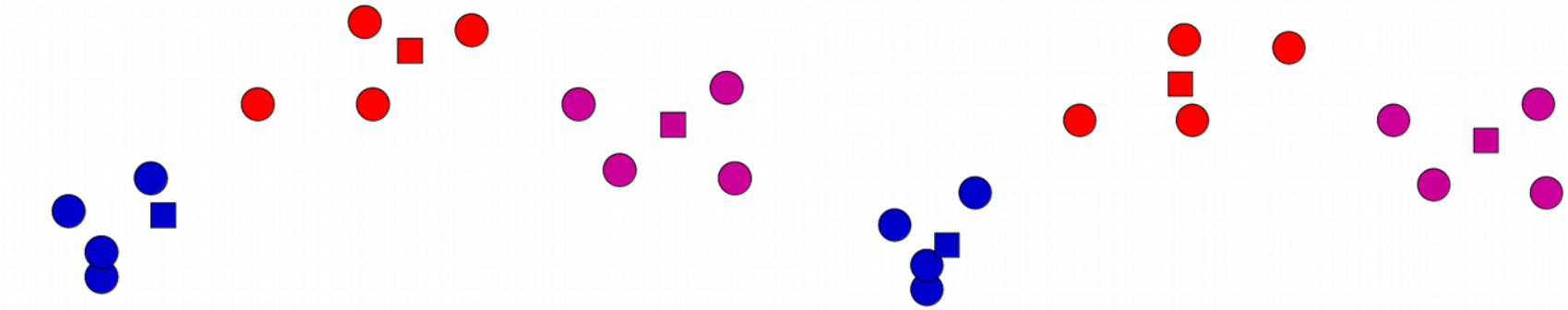


Algorytm 3-means przykład

Wyznaczenie nowych środków ciężkości



Przydział punktów do klas i wyznaczenie środka ciężkości:



KNN @ K-means porównanie

Algorytm KNN

- algorytm klasyfikacji
- Oblicza k najbliższych punktów danych z punktu danych X. Używa tych punktów do określenia, do której klasy X należy,
- Sklasyfikowany punkt nie zmienia klasy,
- Wymaga tylko obliczenia k odległości.

Algorytm K-mean

- algorytm grupowania,
- Wykorzystuje odległość danych do k-centroidów,
- Centroidy niekoniecznie są punktami danych,
- Aktualizuje centroidy po każdej iteracji,
- Musi iterować dane, dopóki punkt centroida się ustabilizuje.

Perceptron jako klasyfikator binarny

- Odnosimy się do dwóch klas: **1 (klasy pozytywnej)** oraz **-1 (klasy negatywnej)**,
- Perceptron oblicza sumę ważoną $z = w_1 x_1 + ... + w_m x_m = w^T \cdot x$ wejść x i wag w, a następnie poddaje je działaniu funkcji aktywacji $\phi(z)$,
- Funkcja aktywacji ϕ jest skokowa tj. daje 1 gdy zostaje przekroczony próg lub -1 w przeciwnym wypadku
- W algorytmie perceptronu funkcja aktywacji jest prostą funkcją skoku jednostkowego,
- Wartość progową Φ możemy zrealizować jako dodatkowe, stałe wejście (bias)

Reguła Rosenblatt nauki perceptronu

1. Wprowadź wagi o wartości 0 lub niewielkich, losowych wartościach.
2. Dla każdej próbki uczącej x^i wykonaj poniższe czynności:
   (a) Oblicz wartość wyjściową $\hat{y}$.
   (b) Zaktualizuj wagi.

W przypadku, gdy pożądana i-ta wartość wyjściowa y^i jest etykietą klasy (tj. -1 lub 1) a neuron odpowiada wartością $\hat{y^i}$ wówczas można obliczyć różnicę odpowiedzi dla wagi w_j.

Wartości wagi należy skorygować o czynnik proporcjonalny do błędu $\bigtriangledown w_j = \eta \cdot (y_i - \hat{y}_i) \cdot x^i_j$, gdzie η jest współczynnikiem uczenia.

nowa wartości wagi w_j po korekcie dla i-tej próbki wynosi:
$w \Leftarrow w + \bigtriangledown w_j$

Sposób nauki perceptronu

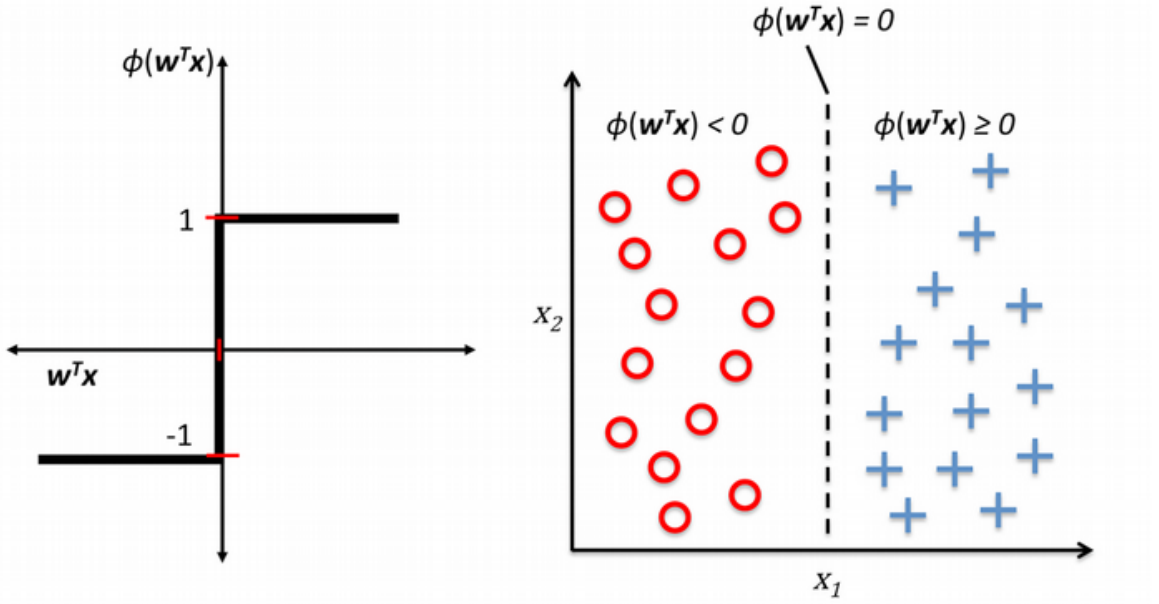
- W przypadku, gdy pożądana i-ta wartość wyjściowa y^i jest etykietą klasy (tj. -1 lub 1) a neuron odpowiada wartością $\hat{y^i}$ wówczas można obliczyć różnicę odpowiedzi dla wagi w_j.
- Wartości wagi należy skorygować o czynnik proporcjonalny do błędu

$$\bigtriangledown w_j = \eta \cdot (y_i - \hat{y}_i) \cdot x_j^i$$

  gdzie η jest współczynnikiem uczenia.
- Nowa wartości wagi w_j po korekcie dla i-tej próbki wynosi:

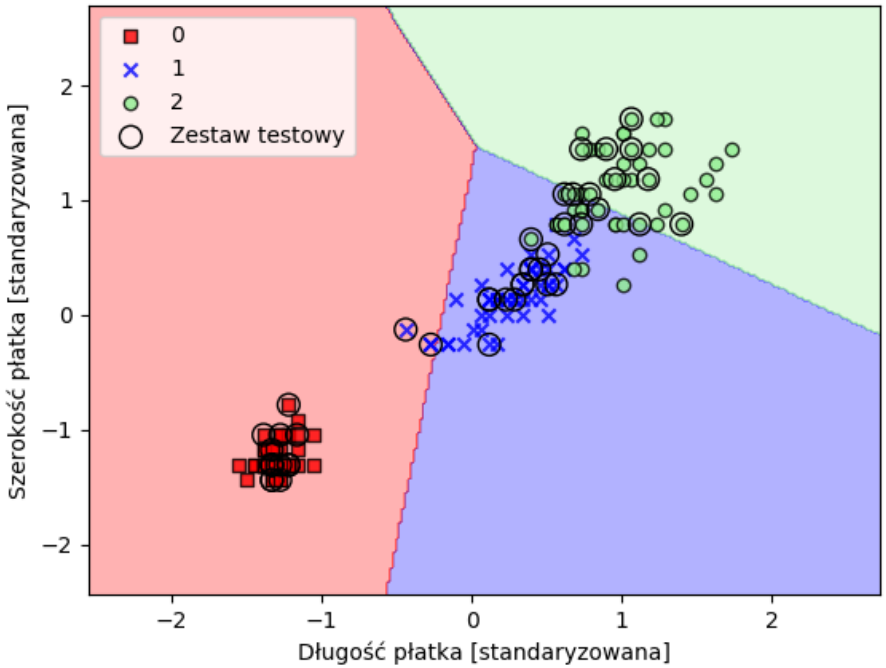
$$w_j \Leftarrow w_j + \bigtriangledown w_j$$

Liniowy podział danych za pomocą perceptronu



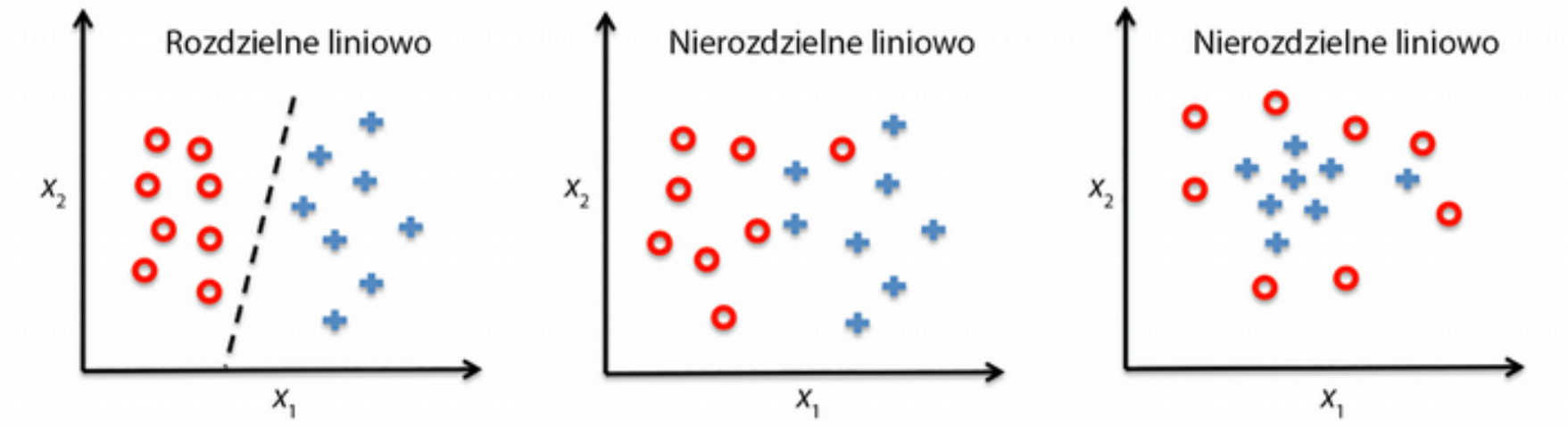
- Zbieżność perceptronu zostaje zapewniona jedynie wtedy, gdy dwie klasy są liniowo rozdzielne,
- Jeżeli nie można oddzielić dwóch klas za pomocą liniowej granicy decyzyjnej, możemy ustalić maksymalną liczbę przebiegów (epok) algorytmu lub próg tolerancji nieprawidłowych klasyfikacji

Klasyfikacja wieloklasowa



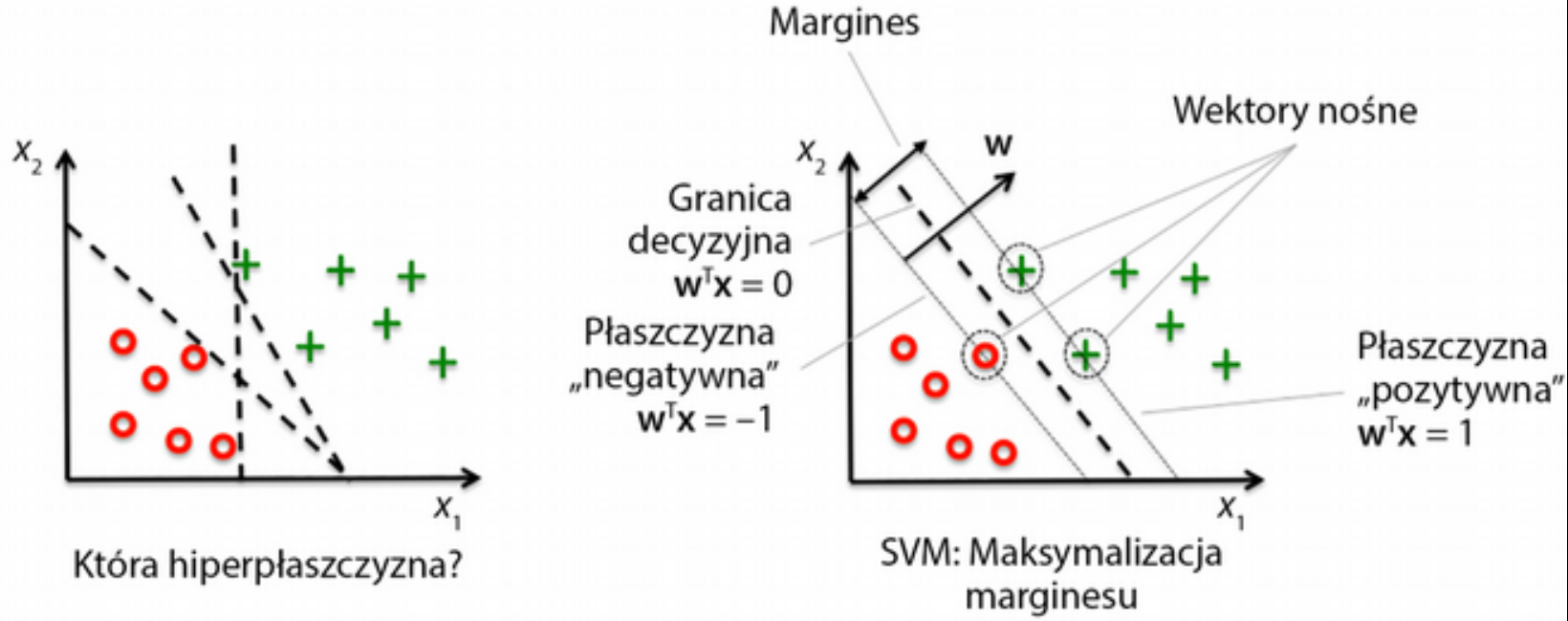
- W przypadku podziału na wiele klas można zastosować wiele klasyfikatorów binarnych,
- Dla każdej z klas można użyć osobny klasyfikator.

Problem liniowej separowalności danych



- Perceptron może jedynie separować dane w sposób liniowy,
- W przypadku, gdy dane nie da się odseparować liniowo należy zastosować inne techniki, jak transformację danych lub klasyfikacja z określonym prawdopodobieństwem.

Maszyna Wektorów Nośnych (*ang. Support Vector Machine*)



- SVM możemy traktować jako rozwinięcie modelu perceptronu.
- Celem optymalizacyjnym modelu SVM jest maksymalizacja marginesu.
- *Margines* definiujemy jako odległość pomiędzy hiperprzestrzenią rozdzielającą (granicą decyzyjną) a najbliższymi próbkami uczącymi (tzw. wektorami nośnymi).

Regularyzacja

- **Wariancja** służy do mierzenia jednorodności modelu prognozowania dla danego wystąpienia próbki w sytuacji wielokrotnego uczenia modelu
- **Obciążenia** stanowi miarę błędu systematycznego niezależnego od losowości.
- Aby regularyzacja mogła zostać właściwie przeprowadzona, musimy sprawić, żeby wszystkie cechy zostały dostosowane do jednolitej skali (np. standaryzacja).
- Najpopularniejszą formą regularyzacji jest tzw. **regularyzacja L2**, zwana także czasami rozpadem wag:

$$\frac{\lambda}{2}||w||^2 = \frac{\lambda}{2}\sum_j w_j^2 \tag{1}$$

Maszyna Wektorów Nośnych - zdefiniowanie problemu

Dla pozytywnej hiperpłaszczyzny mamy $w_0 + w^T \cdot x_{poz} = 1$ i odpowiednio dla negatywnej $w_0 + w^T \cdot x_{neg} = -1$ Po odjęciu stron równań:

$$w^T(x_{poz} - x_{neg}) = 2 \tag{2}$$

Możemy dokonać normalizacji o długość wektora w:

$$||w|| = \sqrt{\sum_j w_j^2}$$

po normalizacji równania (2):

$$\frac{w^T(x_{poz} - x_{neg})}{||w||} = \frac{2}{||w||}$$

Lewą stronę powyższego równania intepretujemy jako odległość pomiędzy hiperpłaszczyzną pozytywną a negatywną, czyli tzw. margines, który chcemy maksymalizować tj:.

$$\frac{w^T(x_{poz} - x_{neg})}{||w||} \Rightarrow max$$

poprzez maksymalizowanie $\frac{2}{||w||} \Rightarrow max$

Zamiast maksymalizować $\frac{2}{||w||}$ można minimalizować $||w||$ lub kwadrat $||w||^2$ oraz "uelastycznić" równania hiperpłaszczyzn, wprowadzając dodatkowe zmienne ζ^i tj.:

$$w^T \cdot x^i \geq 1 - \zeta^i \qquad gdy \qquad y_i = 1$$

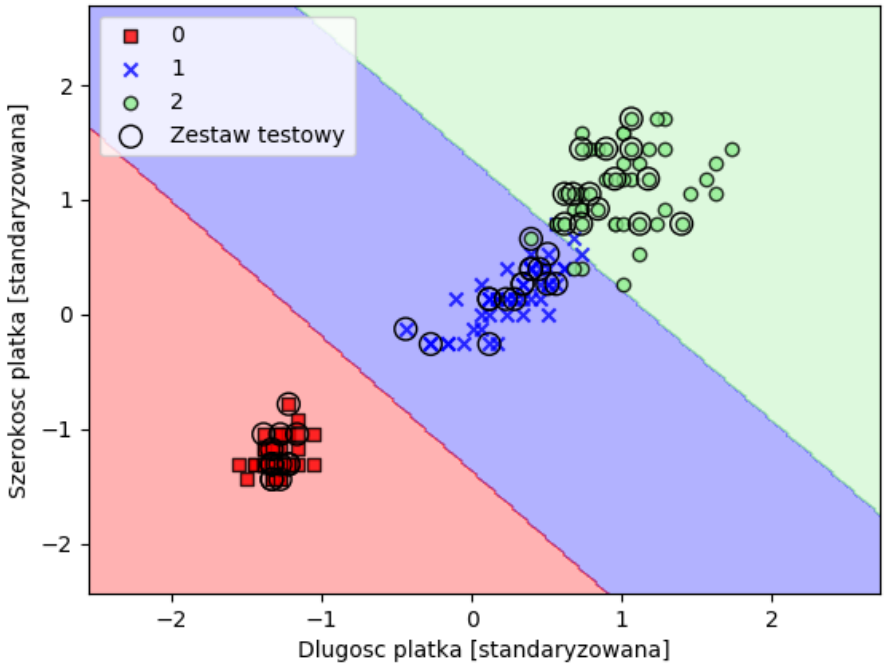
$$w^T \cdot x^i \leq -1 + \zeta^i \qquad gdy \qquad y_i = -1$$

Cel minimalizacji

$$\frac{1}{2}||w||^2 + C\sum_i \zeta^i \Rightarrow min$$

posiada więc dodatkowe ograniczenia, które można dostosowywać, zmieniając parametr C:

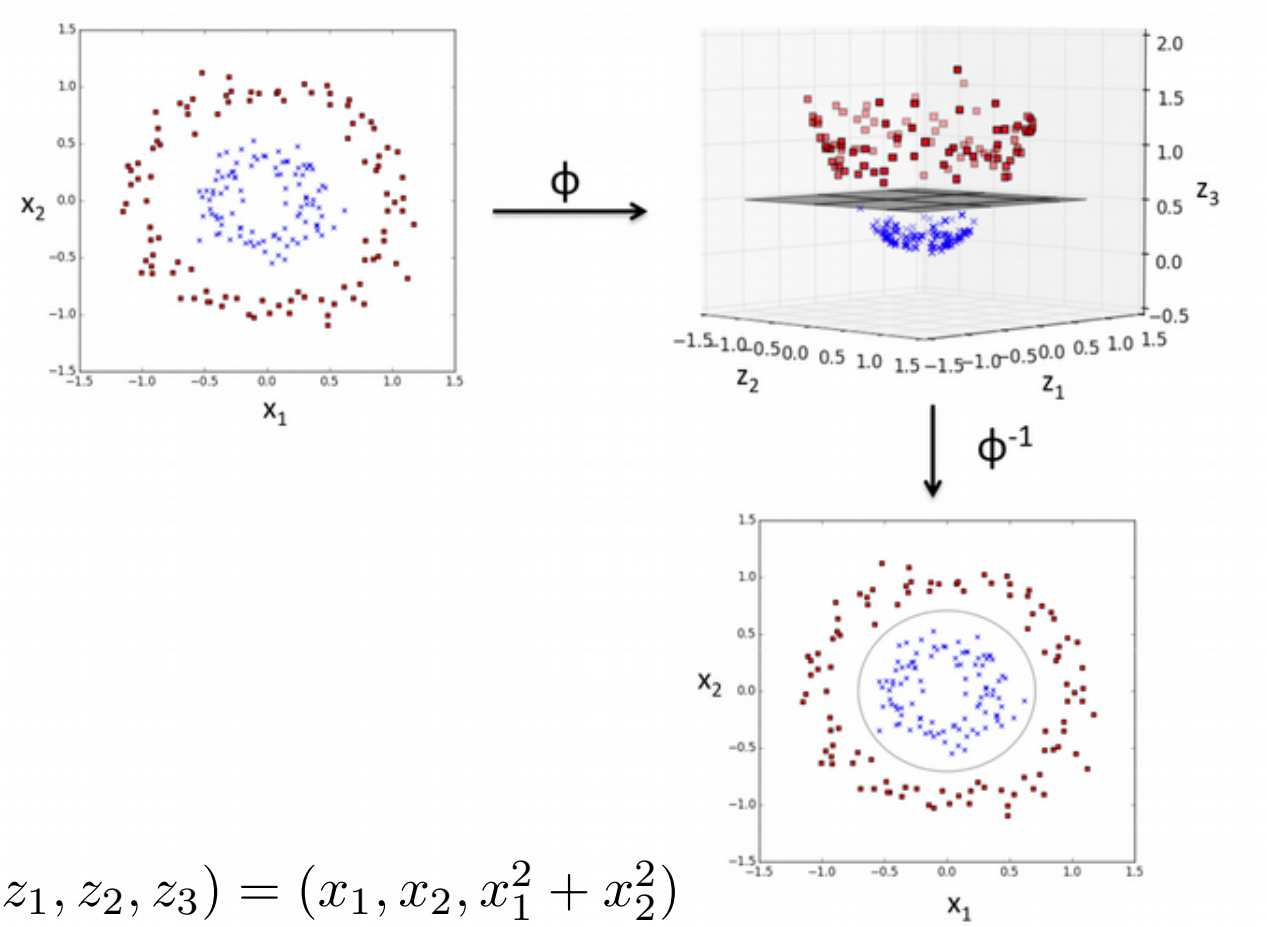
Wynik działania z zastosowaniem SVM



Rozwiązywanie nieliniowych problemów za pomocą jądra SVM

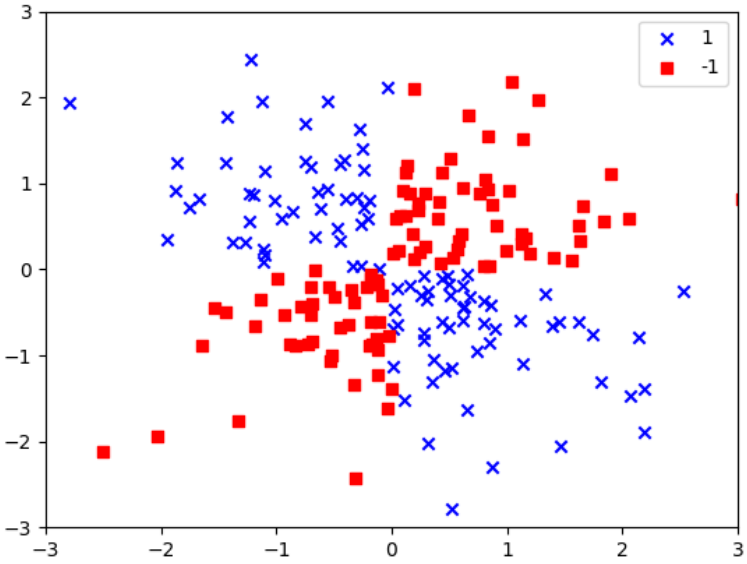
- SVM daje możliwość rozwiązywania nieliniowych problemów klasyfikacji.
- W metodach wykorzystujących funkcje jądrowe podstawową koncepcją radzenia sobie z nierozdzielnymi liniowo danymi jest utworzenie nieliniowych kombinacji pierwotnych cech, które za pomocą funkcji mapowania ϕ będą rzutowane na przestrzeń mającą więcej wymiarów, gdzie staną się liniowo separowalne.

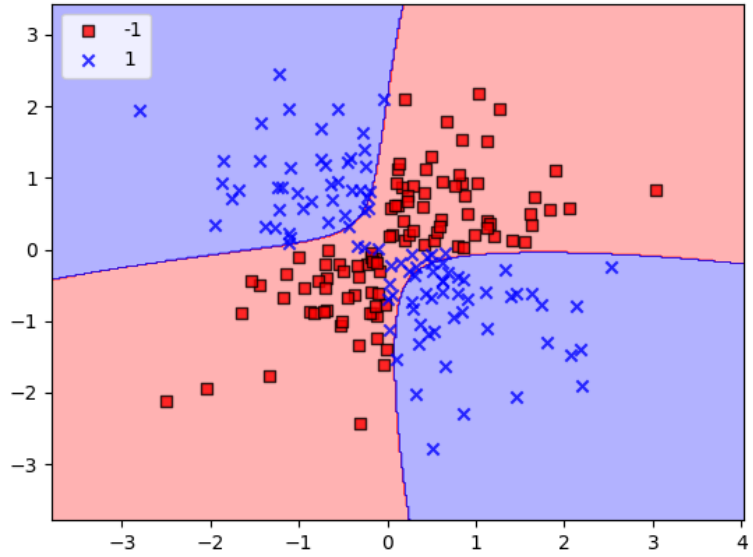
Interpretacja działania funkcji jądrowych SVM



$$\phi(x_1, x_2) = (z_1, z_2, z_3) = (x_1, x_2, x_1^2 + x_2^2)$$

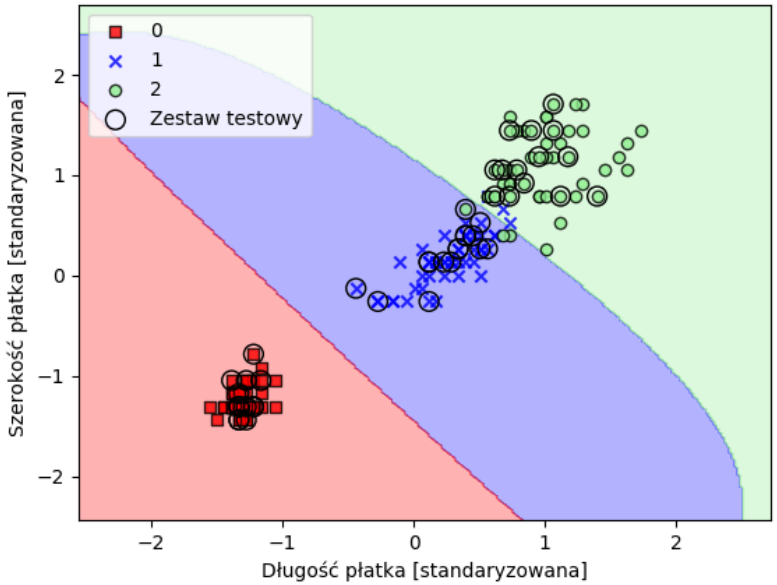
Funkcje jądrowe SVM dla problemu XOR - przykład

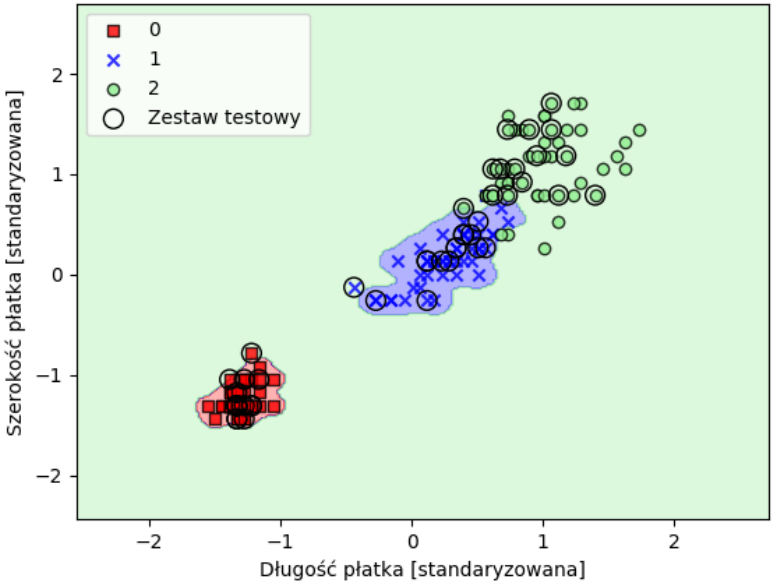




Zestaw danych wygenerowanych za pomocą bramki XOR i granica decyzyjna wygenerowana za pomocą jądra SVM

Jądro Radialnej Funkcji Bazowej dla danych Iris





$$k(x^i, x^j) = exp(-\frac{||x^i - x^j||^2}{2\sigma}) \approx exp(-\gamma||x^i - x^j||^2)$$

Funkcje jądrowe dla różnych wartości parametru $\gamma = \frac{1}{\sigma^2}$, małej i dużej

Klasyfikacja z prawdopodobieństwem - regresja logistyczna

- Perceptron będzie dobrze klasyfikował tylko te dane, które można odseparować liniowo,
- W przypadku przeciwnym nauka perceptronu nigdy się nie zakończy. Można temu zapobiegać wprowadzając ograniczenia na liczbę epok lub dokładność klasyfikacji,
- Regresja logistyczna (ang. logistic regression) jest algorytmem służący do rozwiązywania liniowych i binarnych problemów,
- Regresja logistyczna, pomimo swojej nazwy, jest modelem klasyfikacji, nie regresji,
- Klasyfikacja przy pomocy regresji logistycznej odbywa się z pewnym prawdopodobieństwem.

Funkcja logitowa

- **Ilorazu szans** (ang. odds ratio), czyli szansy wystąpienia danego zdarzenia można wyrazić formułą $\frac{p}{1-p}$ gdzie p oznacza prawdopodobieństwo *pozytywnego zdarzenia*.
- **Funkcja logitowa logit(p)** jest logarytmem ilorazu szans:

$$logit(p) = \log(\frac{p}{1-p})$$

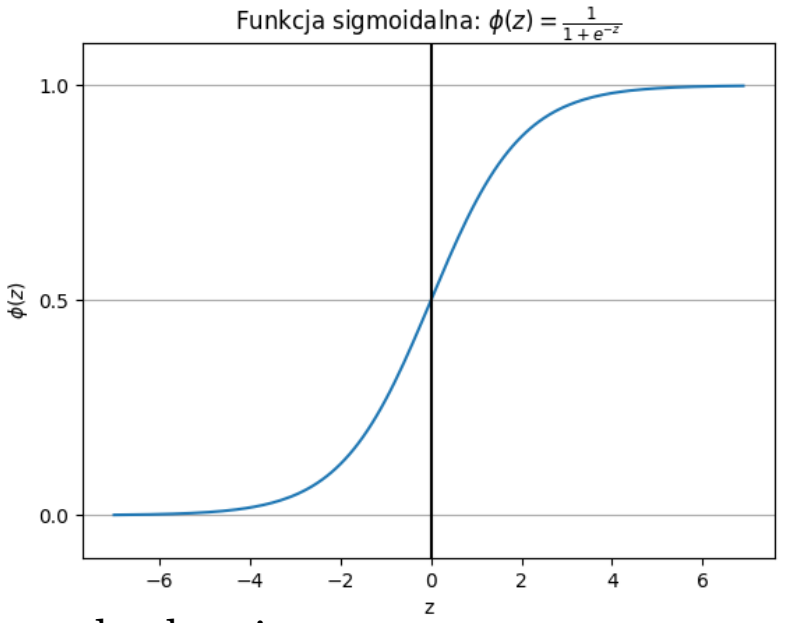
- Funkcja logitowa przyjmuje wartości wejściowe w zakresie od 0 do 1 i przekształca je w wartości z pełnego przedziału liczb rzeczywistych.
- Funkcję logitową możemy wykorzystać do modelowania liniowego związku pomiędzy wartościami cech a zlogarytmowanymi szansami tj.:

$$logit(p(y = 1|x)) = w_0x_0 + w_1x_1 + \ldots w_nx_n = w^T \cdot x$$

gdzie $p(y = 1|x)$ oznacza prawdopodobieństwo warunkowe, zgodnie z którym dana próbka należy do klasy 1 przy znanych cechach x.

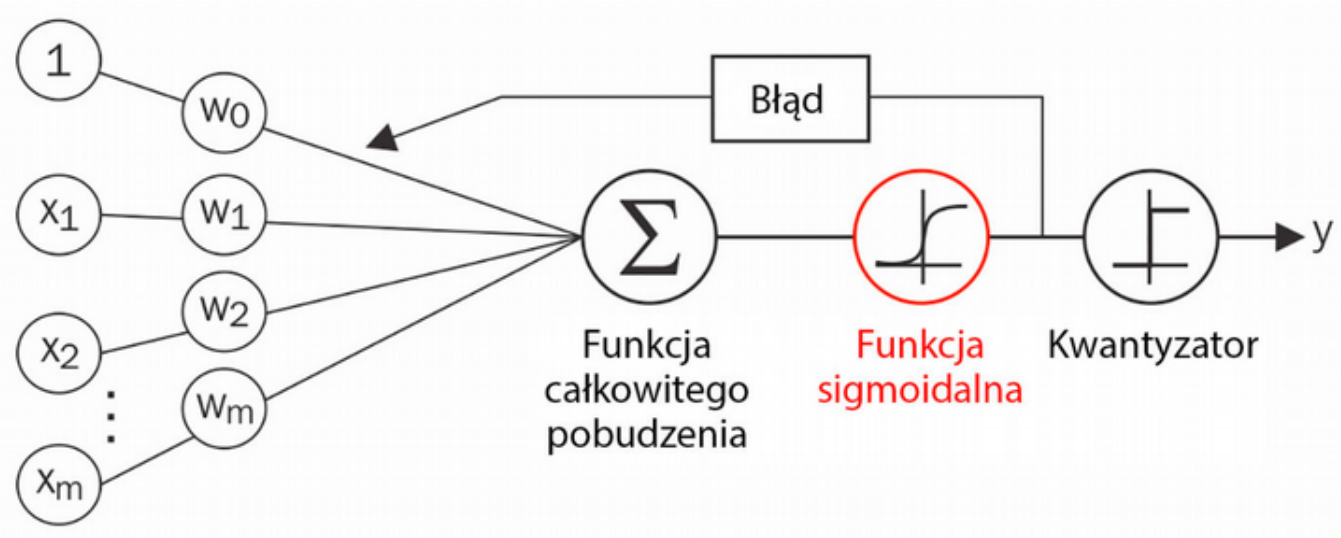
Funkcja logistyczna (sigmoidalna)

- Interesuje nas prognozowanie prawdopodobieństwa przynależności próbki do określonej klasy, co jest odwrotnością funkcji logitowej.
- Mamy tu do czynienia z funkcją logistyczną, zwaną również funkcją sigmoidalną (s-kształtną)



- z określa całkowite pobudzenie $z = w_0x_0 + w_1x_1 + \ldots w_nx_n = w^T \cdot x$,

Schemat modelu regresji logistycznej



- W modelu regresji logistycznej rolę funkcji aktywacji przejmuje funkcja sigmoidalna,
- Wynik funkcji sigmoidalnej jest interpretowany jako prawdopodobieństwo przynależności danej próbki do klasy 1, $\phi(z) = p(y = 1|x, w)$, gdzie x to cechy tej próbki pomnożone przez wagi w.

Funkcja kosztu regresji logistycznej

- W perceptronie funkcją kosztu była suma kwadratów błędów

$$J = \sum_i (\phi(z)^i - y^i)^2$$

- **Wiarygodność L(w)** (dla niezależnych próbek) jest funkcją:

$$L(w) = p(y|x, w) = \prod_i p(y^i|x_i, w) = \prod_i \phi(z^i)^{y^i} \cdot (1 - \phi(z^i))^{1-y^i}$$

- A logarytm wiarygodności wynosi:

$$l(w) = \log(L(w)) = \sum_i [y^i \log(\phi(z^i)) + (1 - y^i) \log(1 - \phi(z^i))]$$

- Regresja logistyczna za funkcję kosztu $J(w)$ przyjmuje $-l(w)$, gdyż sumowanie zmniejsza ryzyko niedomiaru obliczeniowego,

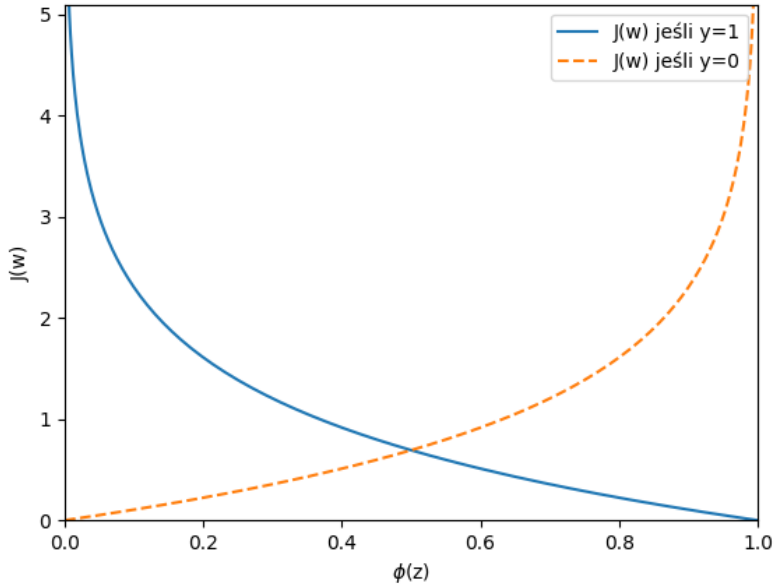
Minimalizacji funkcji kosztu

- Dla i próbek koszt wynosi

$$J(w) = \sum_i [-y^i \log(\phi(z^i)) - (1 - y^i) \log(1 - \phi(z^i))]$$

- Dla pojedynczej próbki y koszt wyniesie

$$J(w) = -y \log(\phi(z)) - (1 - y) \log(1 - \phi(z))]$$



Uczenie modelu regresji logistycznej

Uczenie modelu polegać będzie na minimalizacji funkcji kosztu $J(W)$.

- Pochodna funkcji aktywacji ϕ wynosi:

$$\frac{\delta}{\delta w_j}\phi(z) = \frac{\delta}{\delta w_j}\frac{1}{1+e^{-z}} = \frac{1}{(1+e^{-z})^2}e^{-z} = \phi(z)(1-\phi(z))$$

- Dla wagi w_j gradient funkcji kosztu wyniesie:

$$\frac{\delta}{\delta w_j}J(w) = [y\frac{1}{\phi(z)} - (1-y)\frac{1}{1-\phi(z)}]\frac{\delta}{\delta w_j}\phi(z) =$$

$$= \ldots = (y - \phi(z)) \cdot x_j$$

- Uwzględniając wpływ wszystkich próbek i waga w_j po korekcji wynosi:

$$w_j \Leftarrow w_j + \eta \sum_i (y^i - \phi(z^i)) \cdot x_j^i = w_j + \Delta w_j$$

Implementacja Regresji logistycznej w Pythonie

```
class LogisticRegressionGD(object):
    def __init__(self, eta=0.05, n_iter=100, random_state=1):
        self.eta = eta
        self.n_iter = n_iter
        self.random_state = random_state

    def fit(self, X, y):
        rgen = np.random.RandomState(self.random_state)
        self.w_ = rgen.normal(loc=0.0, scale=0.01, size=1 + X.shape[1])
        self.cost_ = []
        for i in range(self.n_iter):
            net_input = self.net_input(X)
            output = self.activation(net_input)
            errors = (y - output)
            self.w_[1:] += self.eta * X.T.dot(errors)
            self.w_[0] += self.eta * errors.sum()
            cost = (-y.dot(np.log(output)) - ((1 - y).dot(np.log(1 - output))))
            self.cost_.append(cost)
        return self

    def net_input(self, X):
        return np.dot(X, self.w_[1:]) + self.w_[0]

    def activation(self, z):
        return 1. / (1. + np.exp(-np.clip(z, -250, 250)))

    def predict(self, X):
        return np.where(self.net_input(X) >= 0.0, 1, 0)
```

Regularyzacja w regresji logistycznej

- W celu przeprowadzenia regularyzacji wystarczy dodać odpowiedni czynnik funkcji kosztu $J(w)$, który posłuży do zmniejszania wag:

$$J(w) = C \cdot \sum_i [-y^i \log(\phi(z^i)) - (1 - y^i) \log(1 - \phi(z^i))] + \frac{1}{2}||w||^2$$

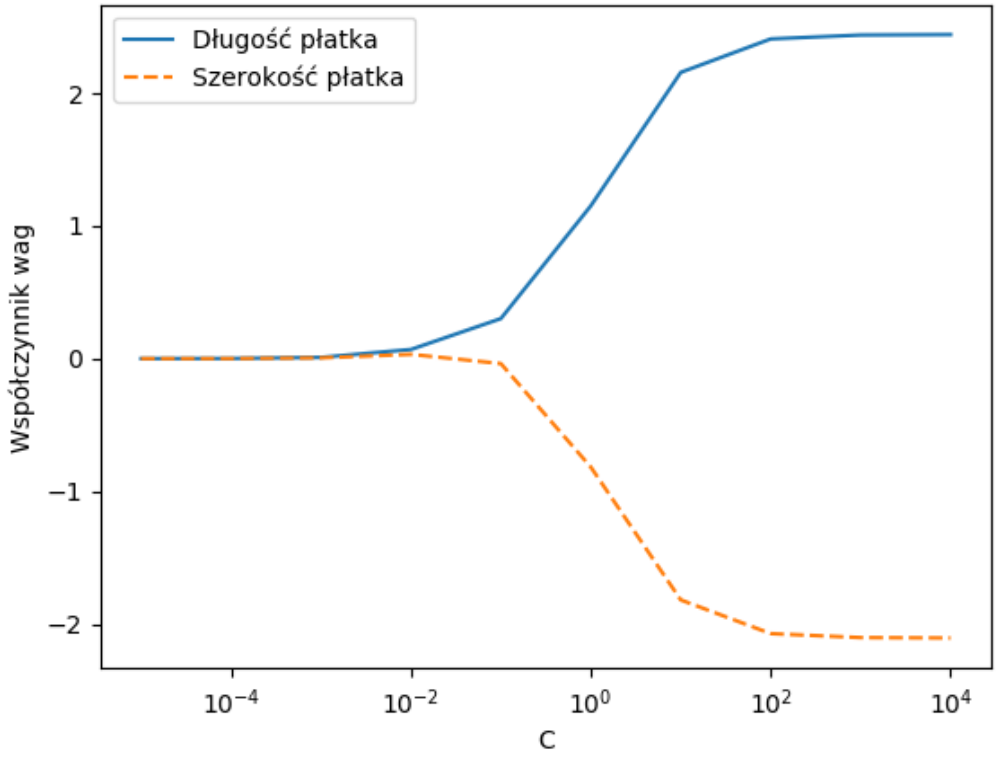
- Parametr C stanowi odwrotność parametru λ we wzorze (1) $C = \frac{1}{\lambda}$
- W bibliotece *sklearn* jest możliwość ustawienia parametru C

```
from sklearn.linear_model import LogisticRegression

    # klasyfikacja w wbudowanego modelu regresji logistycznej
    lr = LogisticRegression(C=1000.0, random_state=1)
    lr.fit(X_train_std, y_train)
```

Kontrola siły regularyzacji

Współczynniki wag maleją w sytuacji zmniejszania wartości parametru C , tzn. w trakcie zwiększania siły regularyzacji.



Zadania klasyfikacji - perceptron, regresja logistyczna

1. Używając perceptronów napisz klasyfikator wielo-klasowy (klasyfikujący 3 lub więcej klas). Dla każdej z klas użyj klasyfikatora binarnego. Przykład klasyfikatora binarnego jest w pliku *perceptron.py*
2. Napisz klasyfikator wielo-klasowy przy użyciu regresji logistycznej. Dla każdej z klas użyj klasyfikatora binarnego. Przykład regresji logistycznej dla przypadku dwuklasowego jest w pliku *reglog.py*
3. Dla regresji logistycznej napisz metodę wypisującą prawdopodobieństwo przynależności próbki (próbek) do danej klasy (klas).

Proszę nie używać dostępnych w bibliotekach klasyfikatorów.