

Implementacja i badanie skuteczności algorytmu k najbliższych sąsiadów (k-NN)

Raport

Kinga Pilch, Hubert Rosiak

14 marca 2018

Krótki opis algorytmu

Algorytm k najbliższych sąsiadów (ang. *k nearest neighbours*) jest prostym algorytmem służącym do rozwiązywania problemu klasyfikacji lub regresji. Idea algorytmu jest następująca: mając zbiór danych uczących w pewnej przestrzeni, dla nowej, nieznanej wcześniej obserwacji prognozujemy wartość będącą średnią arytmetyczną wartości jej k najbliższych (według pewnej metryki, najczęściej euklidesowej) sąsiadów ze zbioru uczącego. Gdy mamy do czynienia z problemem klasyfikacji, nowej obserwacji przypisujemy klasę najczęściej występującą wśród jej k najbliższych sąsiadów.

Krótki opis programu

Nasz program implementuje algorytm k najbliższych sąsiadów do klasyfikacji punktów w dwuwymiarowej przestrzeni Euklidesowej. W programie umożliwiamy zmianę wielkości sąsiedztwa (parametr k) i badamy jej wpływ na działanie algorytmu. Dodatkowo porównujemy zachowanie się podstawowej wersji algorytmu, w której każdy punkt sąsiedztwa ma taki sam wpływ na przewidywane przypisanie do klasy punktu badanego z wersją, w której siła głosu każdego z k sąsiadów jest liniowo zależna od jego odległości od badanego punktu i jest tym większa, im bliżej się on znajduje.

Wybrany język programowania

Projekt stworzyliśmy w języku Python.

Prezentacja wyników eksperymentów

Eksperymenty przeprowadziliśmy dla dwóch rodzajów zbiorów danych:

- *simple* - dane podzielone na dwa zbiorы
- *three_gauss* - dane podzielone na trzy zbiorы

Dla każdego z nich przetestowaliśmy nasz program dla różnych wielkości zbiorów danych, tj. 100, 500, 1000 i 10000 elementów oraz dla różnych wartości parametru k z przedziału [3,30].

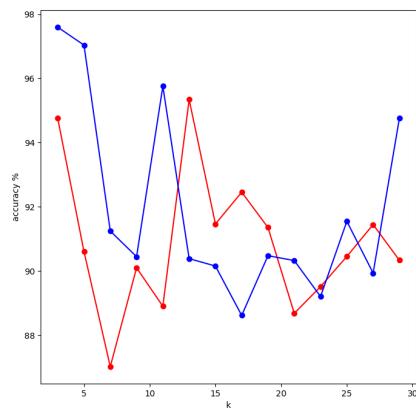
Dokładność

Przeprowadziliśmy pomiary dokładności dla różnych wartości parametru k i porównaliśmy je pomiędzy obiema wersjami algorytmu.

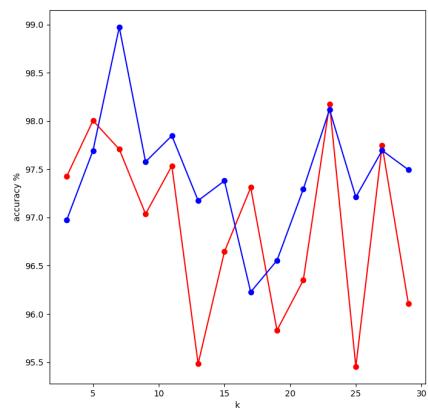
Podział zbioru danych na zbiór treningowy i zbiór testowy odbywał się w sposób losowy, toteż w każdym przypadku obliczenia uruchomiono 3-krotnie i uzyskaną dokładność uśredniono.

Poniższe wykresy przedstawiają porównanie dokładności, czyli liczby elementów zbioru testowego, dla których otrzymano poprawną dokładność, dla obu zbiorów danych i obu wersji algorytmu (podstawowej - linia czerwona, ze średnią ważoną - linia niebieska).

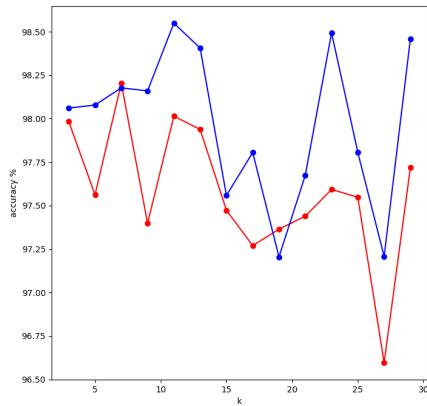
Dane *simple*



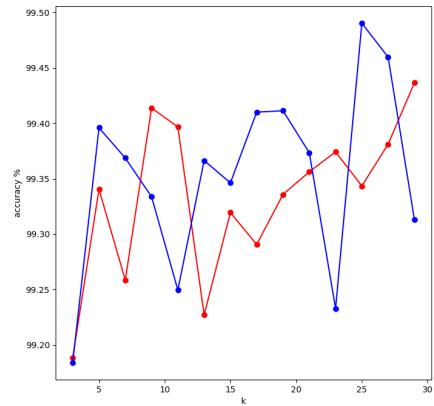
(a) Porównanie dokładności obu algorytmów na zbiorze simple_100



(b) Porównanie dokładności obu algorytmów na zbiorze simple_500

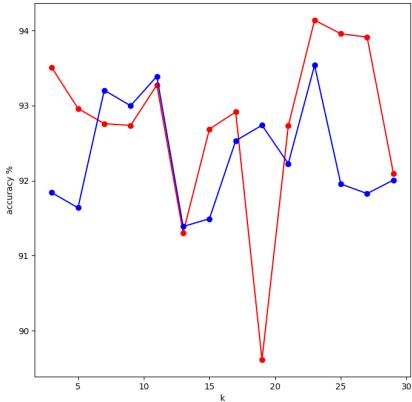


(c) Porównanie dokładności obu algorytmów na zbiorze simple_1000

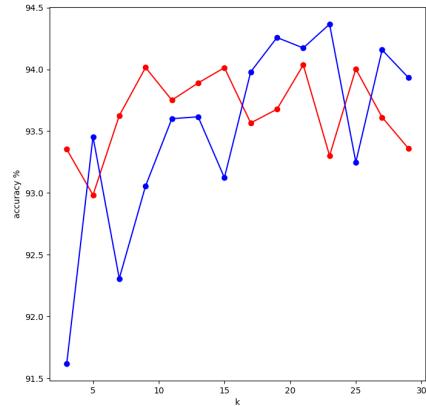


(d) Porównanie dokładności obu algorytmów na zbiorze simple_10000

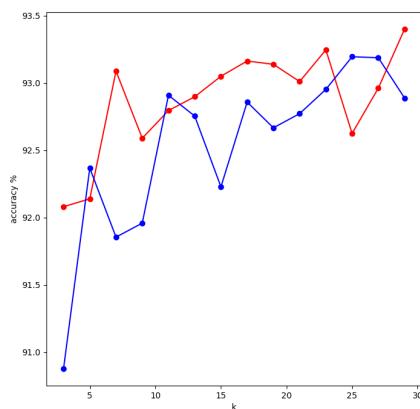
Dane *three_gauss*



(a) Porównanie dokładności obu algorytmów na zbiorze *three_gauss_100*



(b) Porównanie dokładności obu algorytmów na zbiorze *three_gauss_500*



(c) Porównanie dokładności obu algorytmów na zbiorze *three_gauss_1000*

Porównanie dokładności nie wykazało znaczących różnic pomiędzy obiema wersjami algorytmu. W obu przypadkach algorytm sprawdził się dobrze, uzyskując dokładność w granicach 88-94%.

Co zwraca uwagę to fakt, że dla danych simple, im większy zbiór danych (a zatem i zbiór treningowy), tym ogólnie większa dokładność algorytmu (dla 60 danych treningowych i 40 testowych osiągamy dokładność w granicach 87-98%, zaś dla 6000 danych treningowych i 4000 testowych dokładność waha się między 99,2 a 99,8%). Dla tego zestawu danych nie widać jednak wyraźnej zależności wyników od wartości parametru k zaś obie wersje algorytmu zwracają podobne wyniki, nieznacznie lepsze dla algorytmu ze średnią ważoną dla dużego rozmiaru sąsiedztwa.

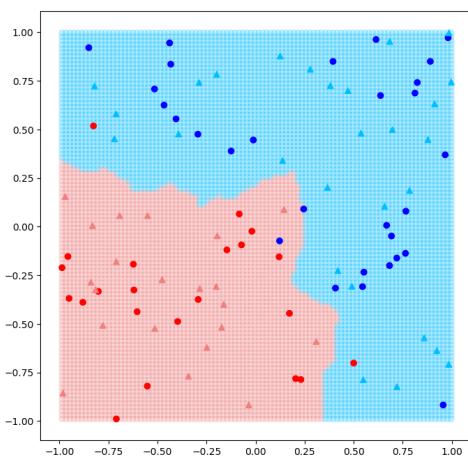
Dla danych *three_gauss* dokładność jest nieco gorsza niż dla danych simple i nie zależy istotnie od rozmiaru danych - w każdym przypadku mieści się w granicach 91-94%. W tej konfiguracji można jednak zauważać poprawę dokładności algorytmu dla większych wartości parametru k i trend ten jest bardziej widoczny dla większych rozmiarów danych - wyraźnie widać go już przy 300 danych testowych i 200 danych treningowych.

Klasyfikacja

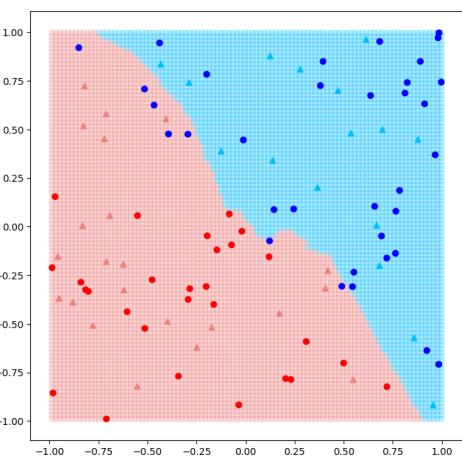
Poniżej zobrazowano wyniki klasyfikacji dla różnych zbiorów danych i obu wersji algorytmu. W każdym przypadku dane podzielono w sposób losowy na zbiór treningowy i testowy w stosunku 6:4, po czym uruchomiono algorytm na danych testowych, za sasiadów, na podstawie których wyznaczana jest klasyfikacja, biorąc k najbliższych punktów ze zbioru treningowego. Dla lepszej prezentacji działania algorytmu, po przeprowadzeniu testów, dokonano również podziału przestrzeni na regularną siatkę 10000 punktów i również dla nich dokonano klasyfikacji, traktując tym razem cały początkowy zbiór jako zbiór testowy.

Kółka w ciemnym kolorze oznaczają zbiór treningowy, jaśniejsze trójkąty - zbiór testowy.

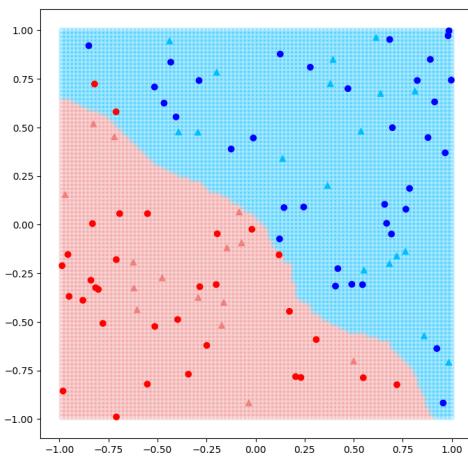
Dane simple_100



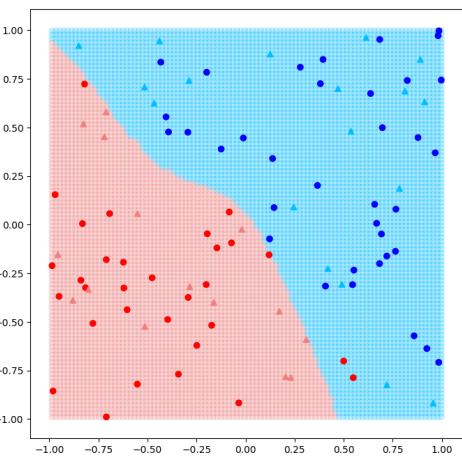
(a) zbiór simple_100, k = 9, wersja podstawowa



(b) zbiór simple_100, k = 29, wersja podstawowa

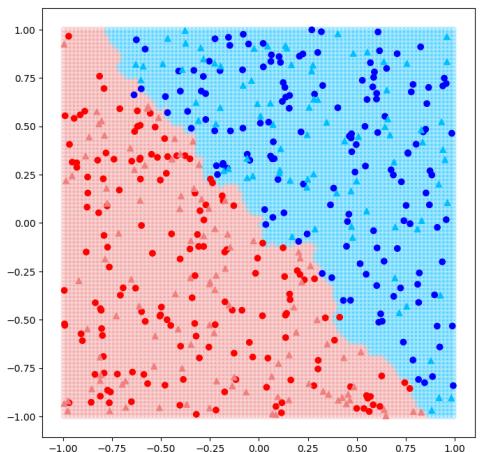


(c) zbiór simple_100, k = 9, wersja ze średnią ważoną

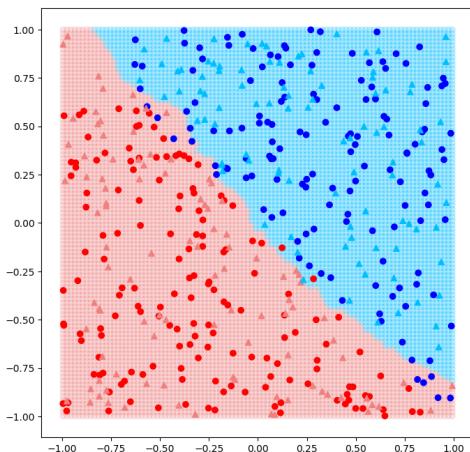


(d) zbiór simple_100, k = 29, wersja ze średnią ważoną

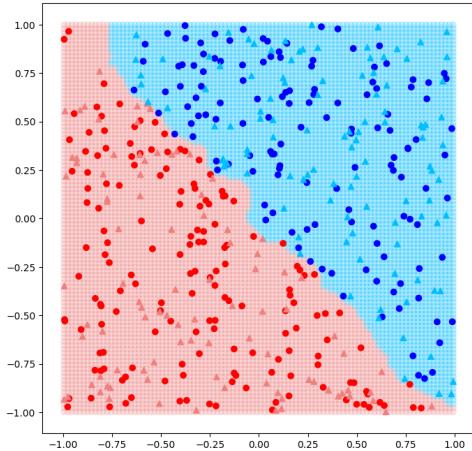
Dane simple _500



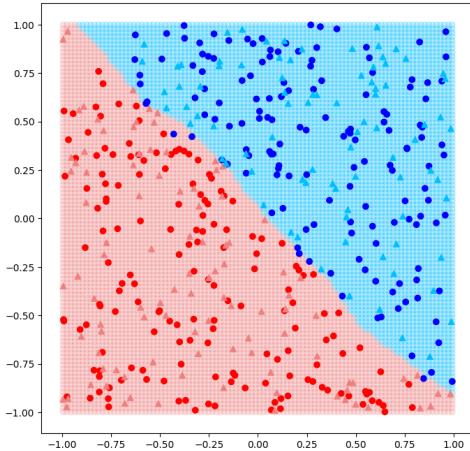
(a) zbiór simple_500, $k = 9$, wersja podstawowa



(b) zbiór simple_500, $k = 29$, wersja podstawowa

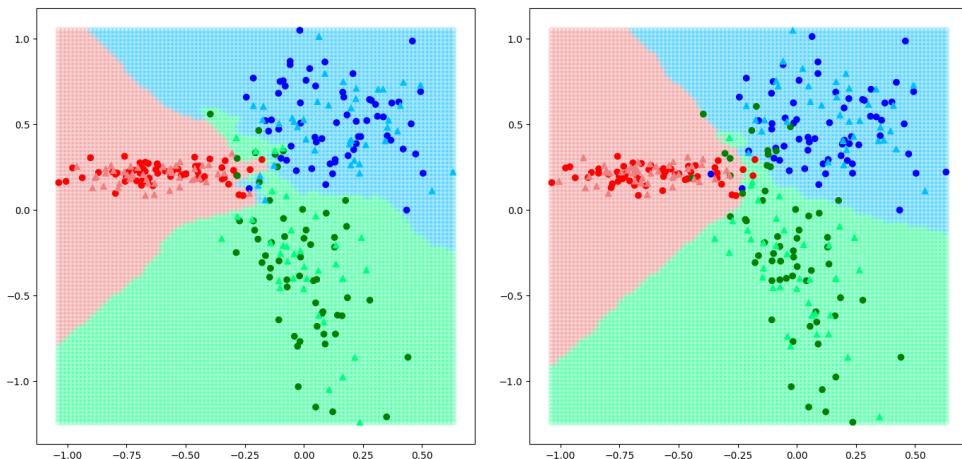


(c) zbiór simple_500, $k = 9$, wersja ze średnią ważoną

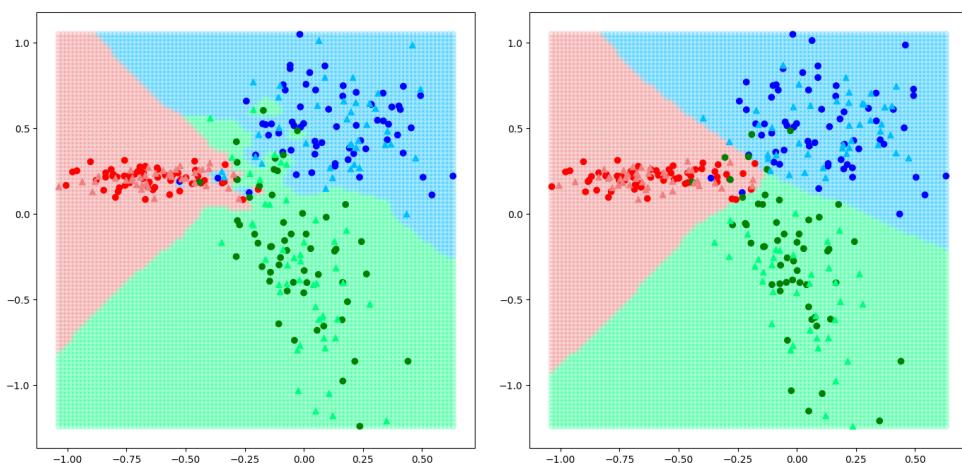


(d) zbiór simple_500, $k = 29$, wersja ze średnią ważoną

Dane three_gauss_100

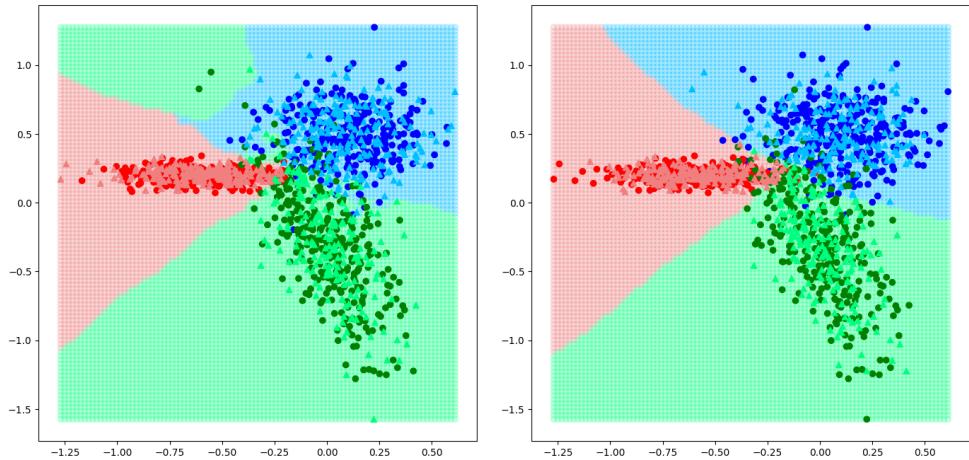


(a) zbiór three_gauss_100, $k = 3$, wersja podstawowa (b) zbiór three_gauss_100, $k = 29$, wersja podstawowa

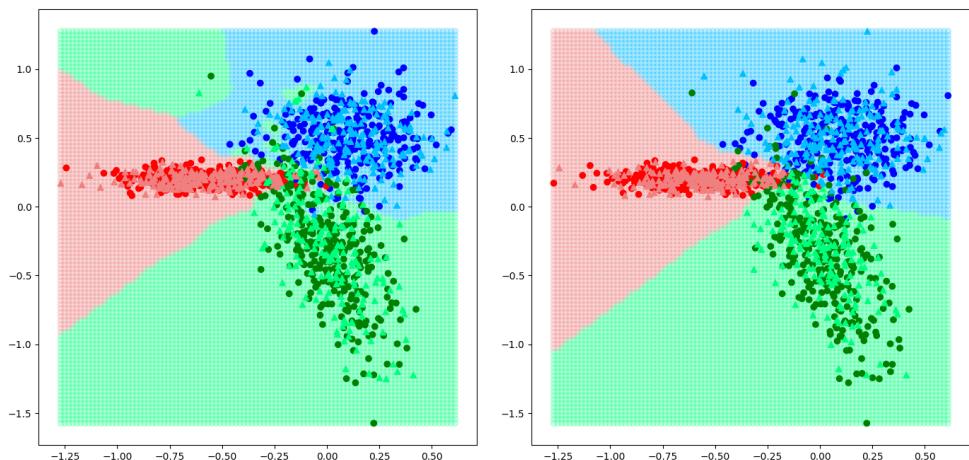


(c) zbiór three_gauss_100, $k = 3$, wersja ze średnią ważoną (d) zbiór three_gauss_100, $k = 29$, wersja ze średnią ważoną

Dane three_gauss_500



(a) zbiór three_gauss_500, $k = 3$, wersja podstawowa (b) zbiór three_gauss_500, $k = 29$, wersja podstawowa



(c) zbiór three_gauss_500, $k = 3$, wersja ze średnią ważoną (d) zbiór three_gauss_500, $k = 29$, wersja ze średnią ważoną

Dla obu rodzajów danych wyraźnie widać wpływ wielkości sąsiedztwa na sposób klasyfikacji. Dla małych k algorytm kieruje się tylko wpływem najbliższego otoczenia i przez to podatny jest na losowe zaburzenia wynikające z losowego wyboru punktów treningowych, które nie są idealnie równomiernie rozłożone w przestrzeni. Dla $k=3$ granice pomiędzy zbiorami mogą być poszarpane i nieregularne, zaś dla $k=29$ przyjmują kształt zbliżony do linii prostych, co spowodowane jest tym, że przy klasyfikacji każdego z punktów testowych mniejsze znaczenie ma wtedy przynależność najbliższych sąsiadów, a bardziej klasyfikacja szerszego otoczenia. Widać to zarówno w przypadku danych *simple*, gdzie dla $k=29$ linia podziału bliższa jest przekątnej, zaś dla danych *three_gauss* większa wartość k powoduje znikanie „wysp” i „półwyspów”, czyli niepożądanych nieregularności w klasyfikacji. Podobny wpływ jak zwiększenie sąsiedztwa ma zwiększenie zbioru danych, dzięki czemu punkty zbioru treningowego rozłożone są bardziej równomiernie i szczelniej wypełniają przestrzeń.

Wykresy te obrazują również niewielką przewagę algorytmu biorącego pod uwagę średnią ważoną odległości przy decydowaniu o klasyfikacji - zwłaszcza dla zbioru danych *simple* oraz *zbioru three_gauss_100* dla $k=29$ widać, że tworzone przez niego granice zbiorów są bardziej regularne, a więc odporne na lokalne zaburzenia.

Literatura

- [1] Sahibsingh A. Dudani, *The Distance-Weighted k-Nearest-Neighbour Rule*, IEEE Transactions on Systems, Man, and Cybernetics (Vol: SMC-6, Issue:4, April 1976)