**Inteligencja Obliczeniowa**

Praca domowa nr 4 – Przetwarzanie obrazu

Krzysztof Kulewski, 238149, grupa 1, 19.01.2019

## Opis zadania

Celem zadania było dokonanie klasyfikacji obrazów przy użyciu wybranych klasyfikatorów,  
a następnie zestawienie otrzymanych wyników i ich analiza.

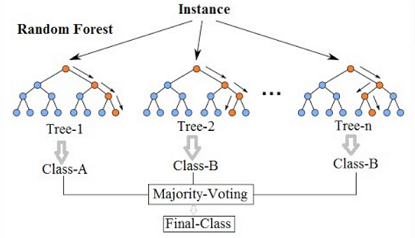
## Wybrany zbiór danych i obróbka

W projekcie postanowiłem posłużyć zbiorem MNIST, czyli zestawem cyfr od 0 do 9 pisanych odręcznie. W jego skład wchodzi 60 000 obrazów treningowych oraz 10 000 obrazów testowych, które zostały użyte do ewaluacji. Każdy z obrazów ma wielkość 28 x 28 pikseli, natomiast każdy piksel jest reprezentowany przez liczbę w zakresie 0 – 255, która opisuje nasycenie kanału alfa.  
Zbiór MNIST zdobył ogromną popularność i stał się wręcz kanoniczny dla zagadnienia klasyfikacji obrazów.   
  
Już na samym początku projektu okazało się, że praca na obrazach w formie plików jest bardzo niewygodna, stąd decyzja, by zamiast plików graficznych, użyć pliku CSV, w którego skład wchodzi 785 kolumn – pierwsza to klasa (cyfry od 0 do 9), a 784 pozostałe reprezentują wartość kanału alfa dla kolejnych pikseli obrazu. Wielkość pliku treningowego to około 110 MB, testowego – 10 MB.  
  
Dane zostały dodatkowo obrobione – wydzielono macierze danych i wektory klas oraz poprawiono nazwy kolumn tak, by mogły być przekazywane do klasyfikatorów:

library(readr)  
train = read\_csv("data/mnist\_train.csv")  
test = read\_csv("data/mnist\_test.csv")  
  
train.labels = train[, 1]  
train.labels = as.factor(train.labels$label)  
test.labels = test[, 1]  
test.labels = as.factor(test.labels$label)  
  
train.data = as.matrix(train[, -1])  
test.data = as.matrix(test[, -1])  
  
names(train.data) = make.names(names(train.data))  
names(test.data) = make.names(names(test.data))

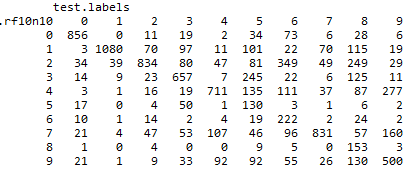
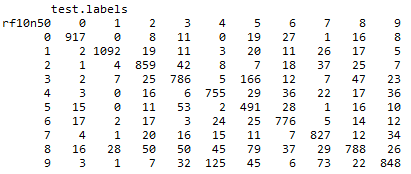
## Klasyfikator I – Random Forest (Lasy Losowe)

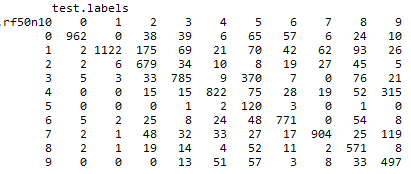
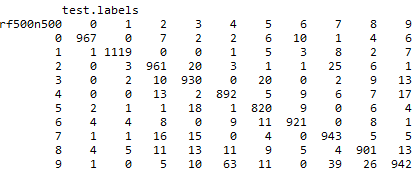
Pierwszy z użytych klasyfikatorów to lasy losowe. Lasy losowe to rozszerzenie drzew decyzyjnych, które polegają na schodzeniu od korzenia do liści, a decyzja o wyborze lewego lub prawego dziecka danego węzła zostaje podjęta w oparciu o wartość parametru, który testuje dany węzeł.  
W lasach losowych, zamiast jednego drzewa, tworzony jest „las” składający się z wielu różnych drzew. Wynikiem klasyfikacji jest cyfra, która została wybrana przez największą liczbę drzew.



Postanowiłem uruchomić klasyfikator RandomForest z wartościami: 10 drzew, 10 węzłów.  
Model został użyty do klasyfikacji danych testowych. Stworzono również macierz błędów:

library(randomForest)  
model.rf10n10 = randomForest(train.data, train.labels, ntree = 10, maxnodes = 10)  
pred.model.rf10n10 = predict(model.rf10n10, test.data, type = "class")  
cm.rf10n10 = table(pred.model.rf10n10, test.labels)  
acc.rf10n10 = sum(diag(cm.rf10n10))/sum(cm.rf10n10)

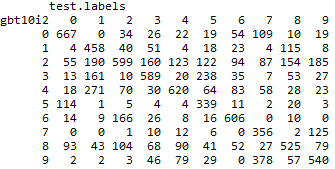
**Macierz błędów RF dla 10 drzew, 10 węzłów:**  
  
**Dokładność to 59,74%**  
  
Wartość taka świadczy o dwóch rzeczach:  
- z pewnością nie był to wybór losowy, gdyż w takim przypadku należałoby oczekiwać ok. 10%,  
- jest to wynik mizerny, gdyż dla zbioru MNIST istnieją klasyfikatory o dokładności 99,5%+  
Sprawdziłem więc, jak zmiana liczby drzew oraz liczby węzłów wpłynie na uzyskane wyniki.   
  
  
**Macierz błędów RF dla 10 drzew, 50 węzłów:**  
  
**Dokładność to 81,39%**

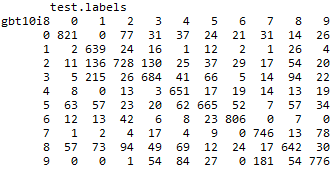
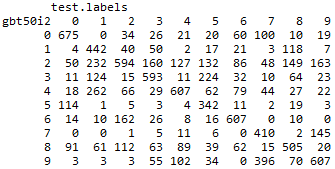
**Macierz błędów RF dla 50 drzew, 10 węzłów:  
  
Dokładność to 72,33%.**Jak widać, zwiększenie liczby węzłów znacznie lepiej wpłynęło na jakość klasyfikacji.  
RF50N10 to liczny las, w którym poszczególne drzewa mają relatywnie niską dokładność.  
RF10N50 to natomiast mniejszy las, ale z dużo dokładniejszymi drzewami. Możemy spodziewać się, że w takim przypadku werdykt był bardziej jednomyślny. Warto odnotować również fakt, że cyfra „5” jest zdecydowanie najtrudniejsza w klasyfikacji, szczególnie dla lasów o małej liczbie węzłów.  
  
Ostatnia kombinacja miała na celu sprawdzenie jak bardzo jesteśmy w stanie polepszyć dokładność przez dalsze zwiększanie parametrów.   
**Macierz błędów RF dla 500 drzew, 500 węzłów:  
  
Dokładność to 93,96%**  
  
Uzyskany wynik wydaje się być całkiem dobry. Tak jak poprzednio, największe problemy sprawia klasyfikacja cyfry „5”. Ciekawą obserwacją jest również to, iż cyfra „4” była często uznawana za „9”.

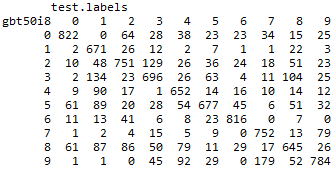
## Klasyfikator II – Gradient Boosted Trees

Podobnie jak w przypadku pierwszego klasyfikatora, także i tutaj mówimy o rozszerzeniu drzew decyzyjnych. Technika gradient boosting polega na stworzeniu grupy „słabych”, prostych klasyfikatorów (analizujących podzbiór parametrów), a następnie iteracyjnym łączeniu ich w całość, celem uzyskania jednego, dokładnego klasyfikatora.  
Klasyfikator ten pozwala na manipulację kilkoma parametrami, między innymi:  
- n.trees - liczba prymitywnych klasyfikatorów (prostych drzew)  
- interaction.depth – liczba podziałów przypadających na węzeł  
  
Oto kod odpowiadający za wygenerowanie modelu, klasyfikację danych testowych oraz stworzenie macierzy błędów:

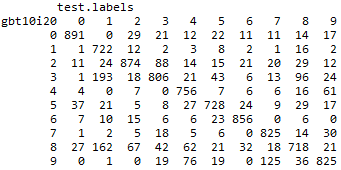
library(gbm)  
model.gbt10i2 = gbm.fit(train.data, train.labels, distribution="multinomial", n.trees=10, interaction.depth=2)  
pred.model.gbt10i2 = predict(model.gbt10i2, test.data, n.trees=model.gbt10i2$n.trees)   
cm.gbt10i2 = table(pred.model.gbt10i2, test.labels)  
acc.gbt10i2 = sum(diag(cm.gbt10i2))/sum(cm.gbt10i2)

**Macierz błędów GBT dla 10 drzew i 2 podziałów na węzeł:** **Dokładność to 52,99%**Podobnie jak w przypadku klasyfikatora Random Forest, uzyskana dokładność jest kiepska, ale z pewnością nie jest to wybór losowy. Warto więc sprawdzić jak zwiększanie parametrów wpłynie na uzyskane wyniki.

**Macierz błędów GBT dla 10 drzew i 8 podziałów na węzeł:**  
 **Dokładność to 71,58%  
  
Macierz błędów GBT dla 50 drzew i 2 podziałów na węzeł:** **Dokładność to 53,82%**

Nie ulega żadnym wątpliwościom to, że zwiększenie parametru interaction.depth ma dużo większy wpływ na uzyskaną dokładność, która wzrosła z 53% do 72%. Pięciokrotne zwiększenie liczby drzew miało natomiast marginalny wpływ na dokładność, polepszając ją zaledwie o punkt procentowy.   
Bardzo ciekawe wydaje się to, że GBT popełnia całkowicie inne błędy niż Random Forest.  
Znacznie lepiej rozpoznawane są cyfry „5” i „9”, natomiast dużo gorzej - „1”.  
  
Oto wynik zwiększenia obu parametrów, analogicznie do pierwszego klasyfikatora.  
**Macierz błędów GBT dla 50 drzew i 8 podziałów na węzeł:**  
 **Dokładność to 72,66%**

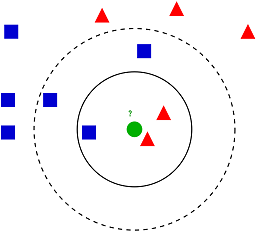
Tym razem, strategia ta okazała się być dużo mniej efektywna – wynik jest bardzo podobny do tego, który uzyskano dla pięciokrotnie mniejszej liczby drzew. Wygląda więc na to, że warto wykonać jeszcze jeden test, w którym znacznie zwiększony zostanie parametr interaction.depth.

**Macierz błędów GBT dla 10 drzew i 20 podziałów na węzeł:  
  
Dokładność to 80,01%**

Założenie okazało się słuszne – uzyskana dokładność jest bliska tej, którą otrzymałem dla Random Forest przy 10 drzewach i 50 węzłach.

## Klasyfikator III – K-Nearest Neighbours (KNN)

W algorytmie K najbliższych sąsiadów, model tworzony jest poprzez porównywanie parametrów zbioru treningowego z jego etykietami. Następnie, dla każdego obiektu (parametrów) w zbiorze testowym, wyszukiwane jest K najbliższych mu sąsiadów, których odległość jest liczona – na ogół – metryką euklidesową.



Najbardziej pasująca wartość parametru K może być odnaleziona stosując automatyczną walidację krzyżową, ale – konsekwentnie względem poprzednich dwóch klasyfikatorów – spróbuję odnaleźć ją ręcznie.

## Źródła

1. Materiały z laboratoriów  
2. Dokumentacja R

## Załączniki

1. Kod źródłowy w R (siec.txt)