Jakub Mielczarek 203943 203943@edu.p.lodz.pl
Łukasz Gołębiewski 203882 203882@edu.p.lodz.pl

Zadanie 3.: Klasyfikacja

1. Cel

Celem zadania jest rozwiązanie problemu poprawnej klasyfikacji wskazanych zbiorów danych z wykorzystaniem narzędzi inteligentnej analizy danych, w tym perceptronu wielowarstwowego. Dodatkowo należało przeprowadzić analizę wyników uczenia za pomocą macierzy konfuzji, przedstawić graficznie przebieg uczenia perceptronu, wykorzystać inny biblioteczny algorytm klasyfikacji oraz dokonać ekstrakcji cech zbioru danych przed rozpoczęciem nauki perceptronu. Zbiory danych na których uczono perceptron to "Iris dataset" zawierający opis czterech wymiarów 150 kwiatów z podziałem na gatunek oraz "MNIST dataset" zawierający 60000 obrazków w skali szarości 28x28 pikseli przedstawiających ręcznie napisane cyfry od zera do dziewięciu wraz z informacją o tym jaka cyfra znajduje się na obrazku. Dodatkowo jest oddzielny zbiór testowy zawierający 10000 obrazków.

2. Opis implementacji

Implementacja została wykonana w języku Java. Zgodnie z wymaganiami zadania dodane zostały funkcje umożliwiające wyliczenia macierzy konfuzji. Dodatkowy inny klasyfikator zamiast perceptronu to algorytm drzewa decyzyjnego pochodzącego z biblioteki Apache Spark (pakiet mllib). W celu ekstrakcji cech z obrazków użyto algorytmu HOG (Histogram of Oriented Gradients) z biblioteki openCV. Za pomocą programu gnuplot rysowane są wykresy przebiegu uczenia perceptronu. W stosunku do bazowego projektu z poprzedniego zadania z implementacją perceptronu wielowarstwowego dodane zostały następują klasy:

— ConfusionMatrix.java - klasa odpowiedzialna za wyliczania macierzy konfuzji.
— HogManager.java - klasa odpowiedzialna za ekstrakcję cech za pomocą HOG z użyciem openCV.
— IdxManager.java - klasa odpowiedzialna za ekstrakcję danych z formatów *idx w których znajdowały się dane z jasnościami pikseli obrazków.

Dodatkowo klasa ta opakowuje odczytane dane, została zserializowania dzięki czemu nie trzeba za każdym razem prasować plików z danymi a jedynie odczytać zdeserializowany obiekt.

— SubsetManager.java - klasa umożliwiająca podzielenie zbioru danych na zbiór testowy i treningowy.

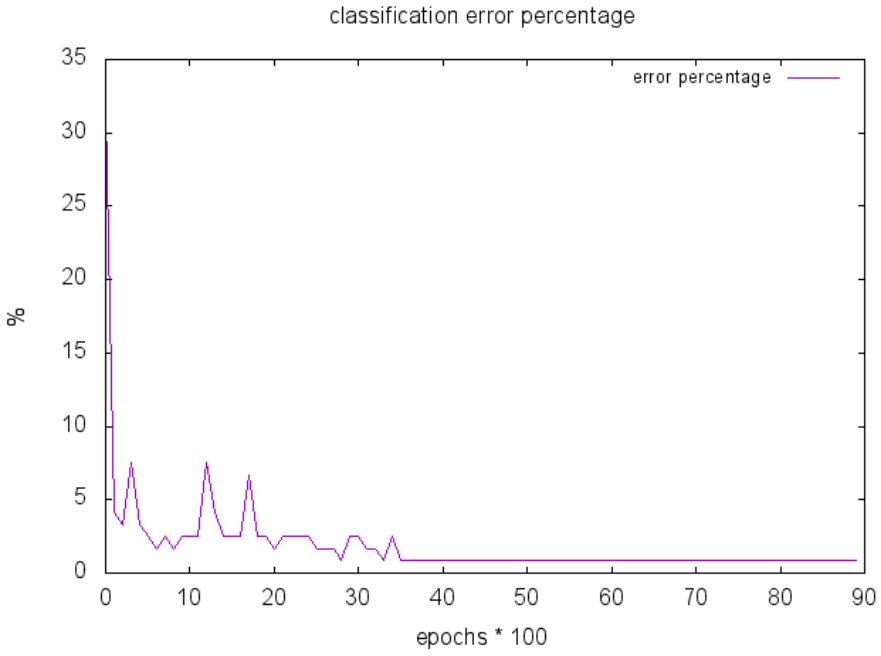
3. Materiały i metody

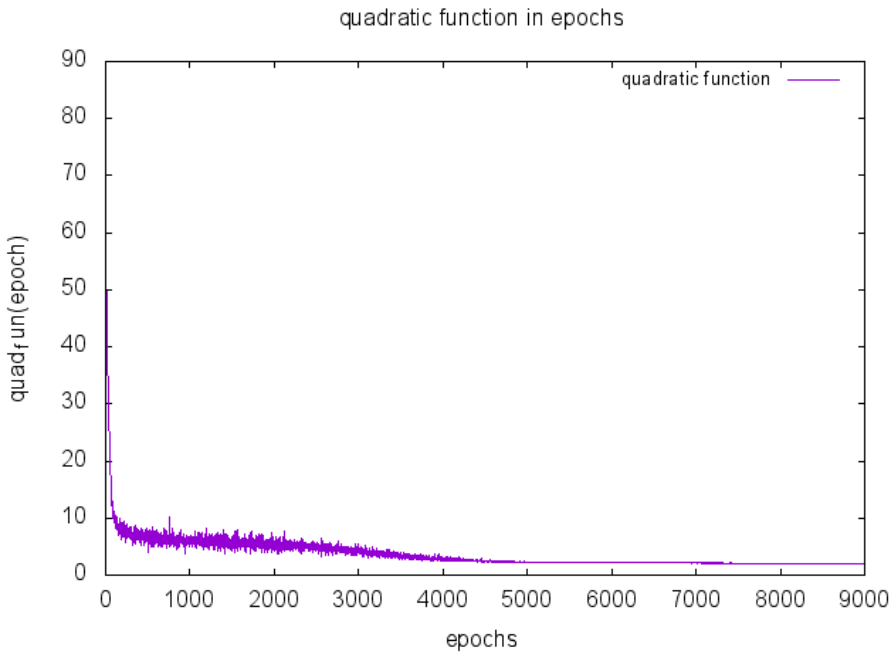
Pierwsza część badawcza polega na nauce perceptronu i drzewa decyzyjnego w celu klasyfikacji gatunków irysów. W rozważanych przypadkach brany jest pod uwagę wpływ na naukę takich parametrów jak ilość epok, czas uczenia, ilość neuronów w warstwach ukrytych, podział na zbiór testowy i treningowy.
Druga część badawcza polega na nauce perceptronu i drzewa decyzyjnego w celu klasyfikacji obrazków z MNIST dataset. Badany jest wpływ na naukę takich parametrów jak zastosowanie ekstrakcji cech, ilość epok, ilość neuronów w warstwach ukrytych, czas uczenia.
W przypadku pierwszej części zbiór danych na zbiór testowy i treningowy był dzielony losowo. W drugiej częśći wykorzystano przygotowanie oddzielnie zbiory danych treningowych i testowych.

4. Wyniki

4.1. Część badawcza 1: Iris dataset

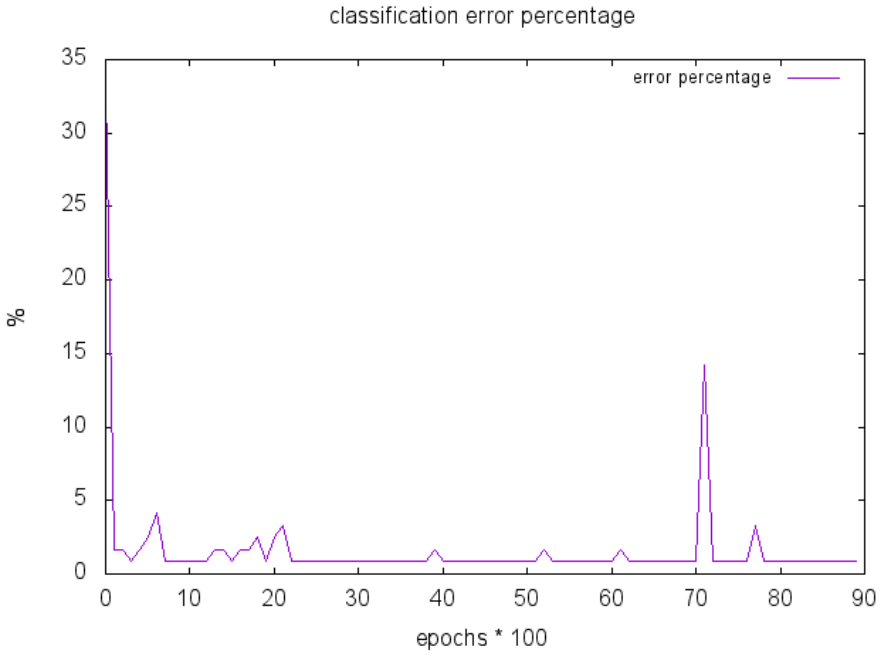
(P1) Dane wejściowe: perceptron 4-12-3, bias, learning rate = 0.01, momentum = 0.8, ilość epok = 9000, dane treningowe: 80% losowo

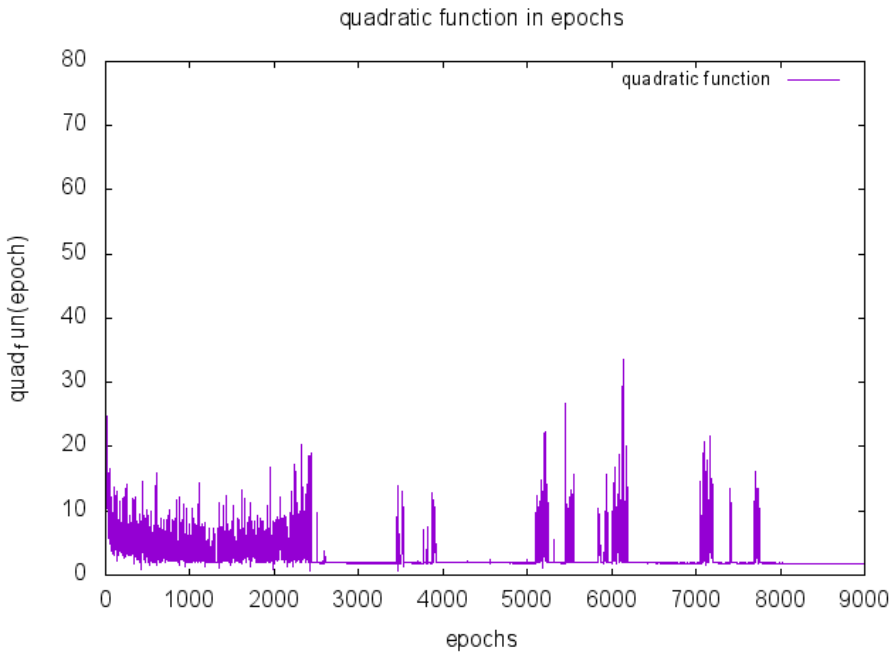




Macierz konfuzji dla zbioru testowego:
Klasa: iris-setosa TP: 9 FN: 0 FP: 0 TN: 21
Klasa: iris-versicolor TP: 12 FN: 0 FP: 0 TN: 18
Klasa: iris-virginica TP: 9 FN: 0 FP: 0 TN: 21
Suma: TP: 30 FN: 0 FP: 0
Czas uczenia: 1.828s

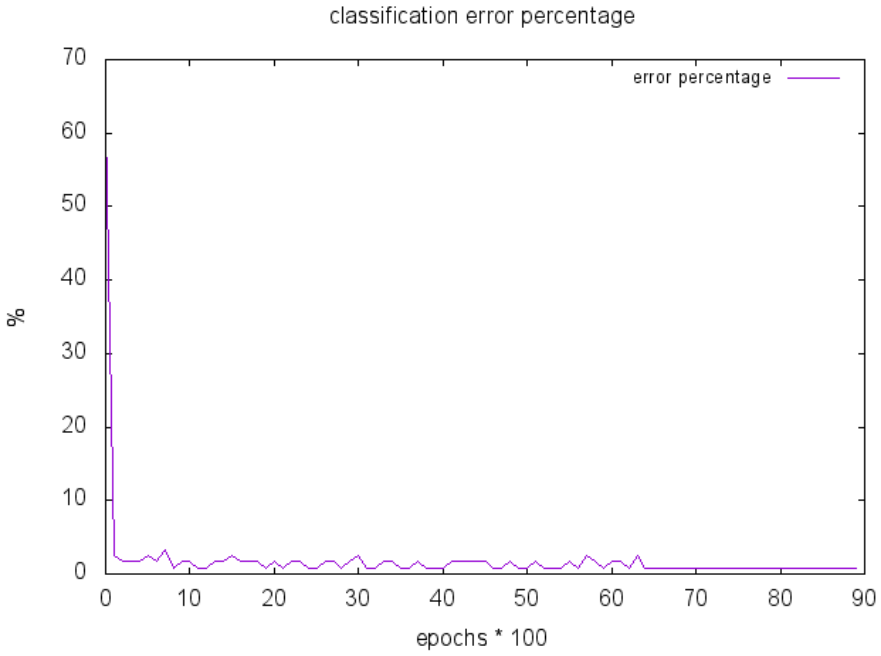
(P2) Dane wejściowe: Jak w P1 ale learning rate = 0.5, momentum = 0.5

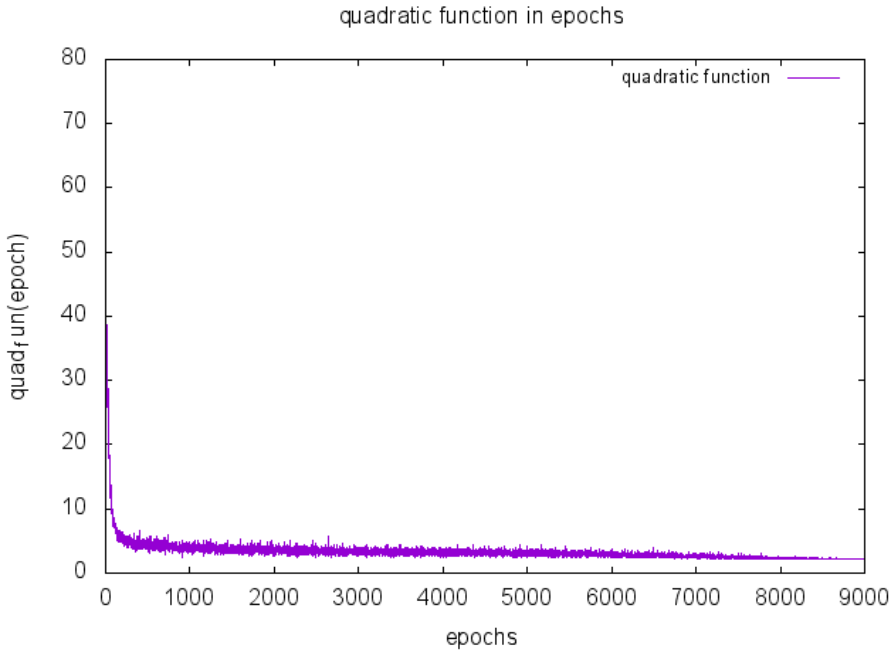




Macierz konfuzji dla zbioru testowego:
Klasa: iris-setosa TP: 12 FN: 0 FP: 0 TN: 18
Klasa: iris-versicolor TP: 8 FN: 2 FP: 0 TN: 20
Klasa: iris-virginica TP: 8 FN: 0 FP: 2 TN: 20
Suma: TP: 28 FN: 2 FP: 2
Czas uczenia: 1.797s

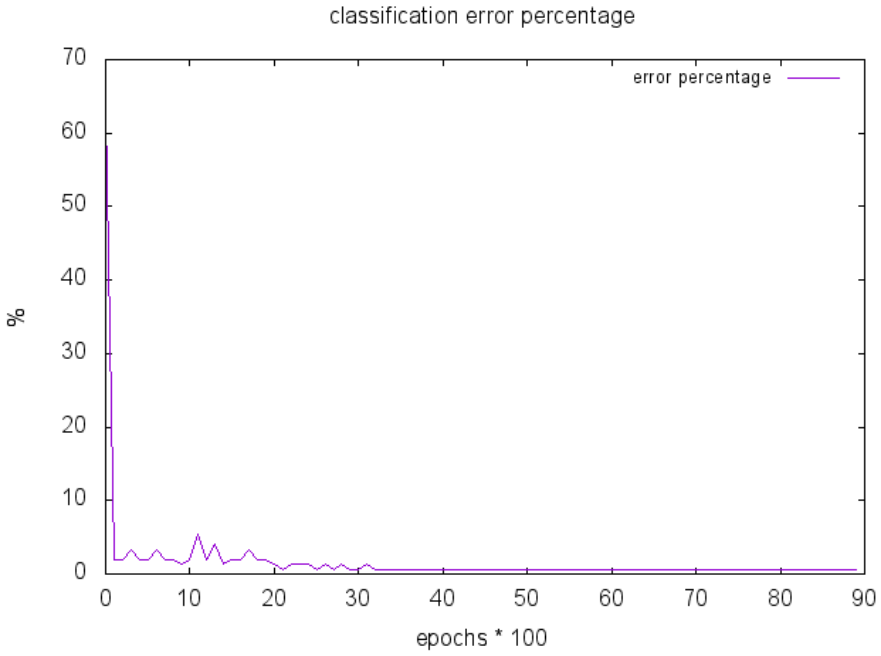
(P3) Dane wejściowe: Jak w P1 ale dla perceptronu 4-100-3

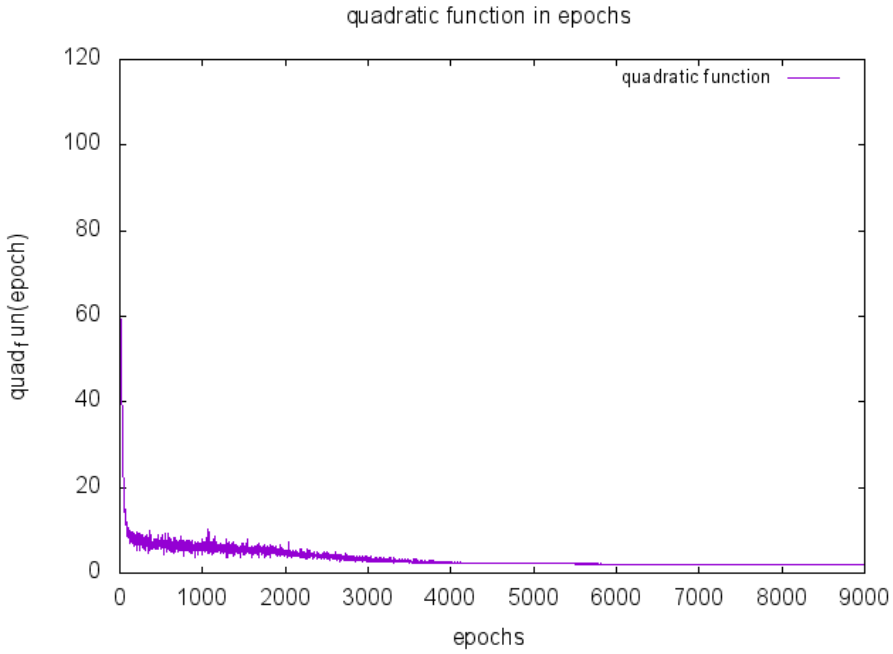




Macierz konfuzji dla zbioru testowego:
Klasa: iris-setosa TP: 10 FN: 0 FP: 0 TN: 20
Klasa: iris-versicolor TP: 9 FN: 1 FP: 0 TN: 20
Klasa: iris-virginica TP: 10 FN: 0 FP: 1 TN: 19
Suma: TP: 29 FN: 1 FP: 1
Czas uczenia: 11.364s

(P4) Dane wejściowe: Jak w P1 ale tylko zbiór treningowy (100%)

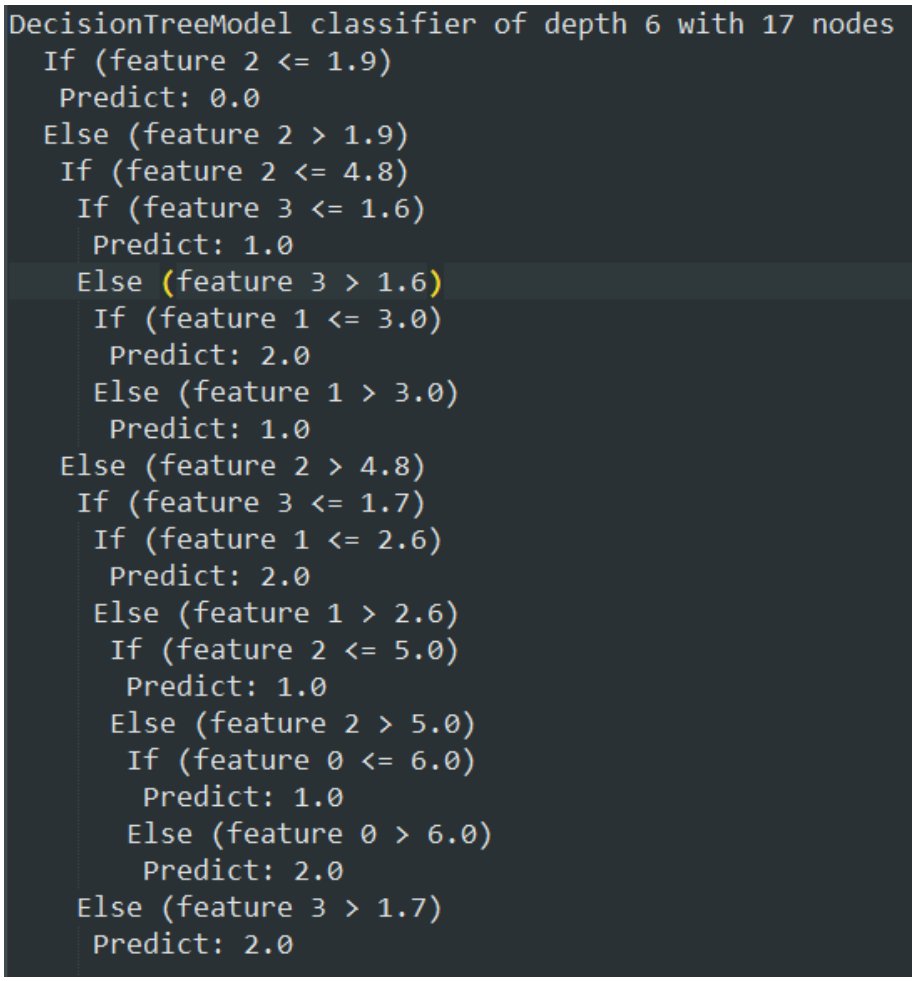




Macierz konfuzji dla zbioru treningowego:
Klasa: iris-setosa TP: 50 FN: 0 FP: 0 TN: 100
Klasa: iris-versicolor TP: 49 FN: 1 FP: 0 TN: 100
Klasa: iris-virginica TP: 50 FN: 0 FP: 1 TN: 99
Suma: TP: 149 FN: 1 FP: 1
Czas uczenia: 2.243s

(P5) Drzewo decyzyjne Apache Spark. Dane wejściowe: zbiór danych treningowych 80% losowo, maxDepth = 10, maxBins = 32, impurity = "gini"

Macierz konfuzji dla zbioru testowego:
Klasa: iris-setosa TP: 12 FN: 0 FP: 0 TN: 18
Klasa: iris-versicolor TP: 9 FN: 1 FP: 0 TN: 20
Klasa: iris-virginica TP: 8 FN: 0 FP: 1 TN: 21
Suma: TP: 29 FN: 1 FP: 1
Czas uczenia: 1.133s
Błąd drzewa decyzyjnego: 0.033



```
DecisionTreeModel classifier of depth 6 with 17 nodes
  If (feature 2 <= 1.9)
   Predict: 0.0
  Else (feature 2 > 1.9)
   If (feature 2 <= 4.8)
    If (feature 3 <= 1.6)
     Predict: 1.0
    Else (feature 3 > 1.6)
     If (feature 1 <= 3.0)
      Predict: 2.0
     Else (feature 1 > 3.0)
      Predict: 1.0
   Else (feature 2 > 4.8)
    If (feature 3 <= 1.7)
     If (feature 1 <= 2.6)
      Predict: 2.0
     Else (feature 1 > 2.6)
      If (feature 2 <= 5.0)
       Predict: 1.0
      Else (feature 2 > 5.0)
       If (feature 0 <= 6.0)
        Predict: 1.0
       Else (feature 0 > 6.0)
        Predict: 2.0
    Else (feature 3 > 1.7)
     Predict: 2.0
```

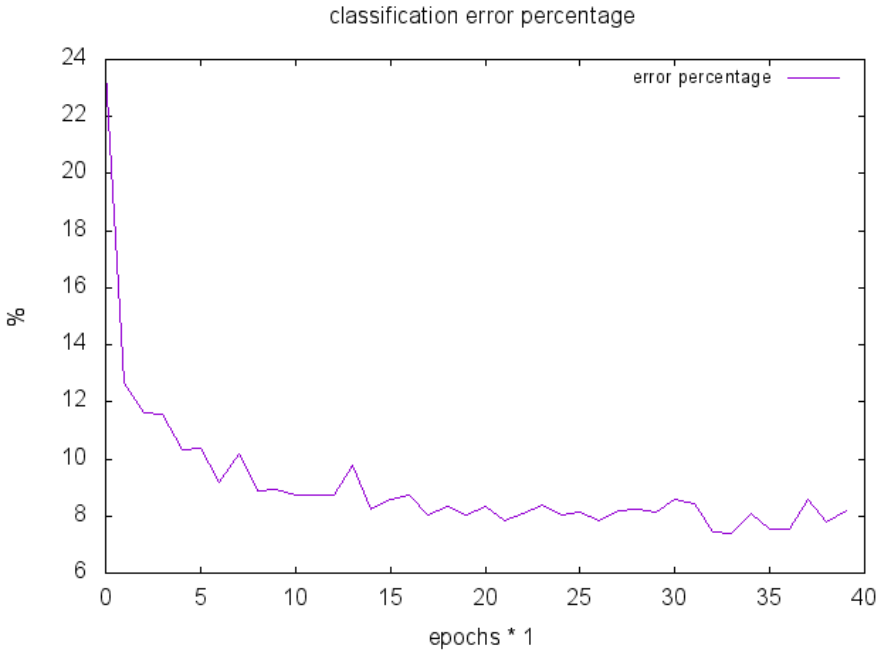
(P6) Drzewo decyzyjne Apache Spark. Dane wejściowe jak w P5 ale tylko zbiór treningowy (100%)

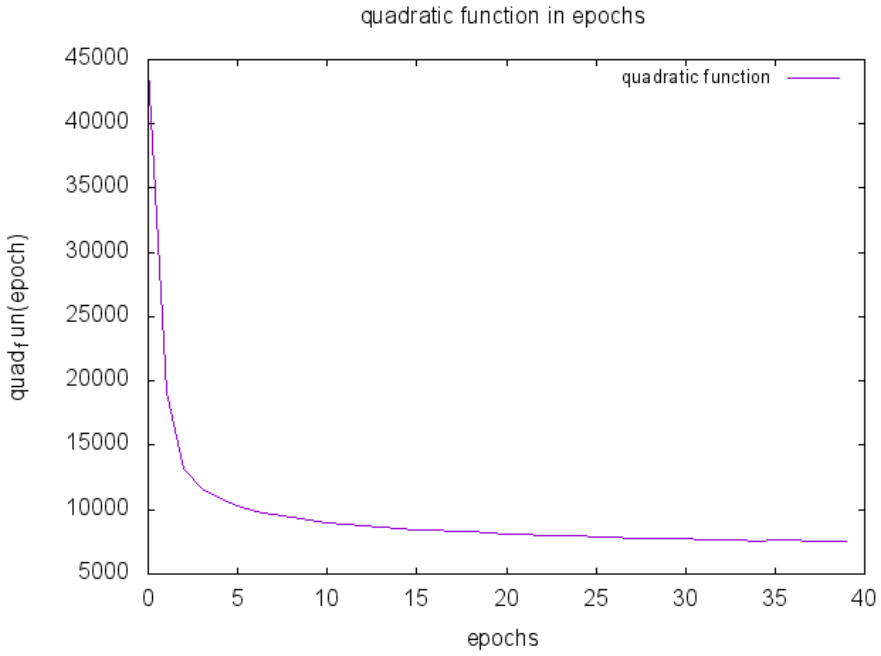
Macierz konfuzji dla zbioru treningowego:
Klasa: iris-setosa TP: 50 FN: 0 FP: 0 TN: 100
Klasa: iris-versicolor TP: 50 FN: 0 FP: 0 TN: 100
Klasa: iris-virginica TP: 50 FN: 0 FP: 0 TN: 100
Suma: TP: 150 FN: 0 FP: 0
Czas uczenia: 1.233s
Błąd drzewa decyzyjnego: 0.0

```
DecisionTreeModel classifier of depth 5 with 17 nodes
  If (feature 2 <= 1.9)
   Predict: 0.0
  Else (feature 2 > 1.9)
   If (feature 3 <= 1.7)
    If (feature 2 <= 4.9)
     If (feature 3 <= 1.6)
      Predict: 1.0
     Else (feature 3 > 1.6)
      Predict: 2.0
    Else (feature 2 > 4.9)
     If (feature 3 <= 1.5)
      Predict: 2.0
     Else (feature 3 > 1.5)
      If (feature 0 <= 6.7)
       Predict: 1.0
      Else (feature 0 > 6.7)
       Predict: 2.0
   Else (feature 3 > 1.7)
    If (feature 2 <= 4.8)
     If (feature 0 <= 5.9)
      Predict: 1.0
     Else (feature 0 > 5.9)
      Predict: 2.0
    Else (feature 2 > 4.8)
     Predict: 2.0
```

4.2. Część badawcza 2: MNIST dataset

(P7) Dane wejściowe 60000 obrazkow treningowych, 10000 tysiecy obrazkow testowych, normalizacja danych wejściowych do przedziału [0;1], perceptron 784-15-10, bias, learning rate = 0.01, momentum = 0.2, ilość epok = 40

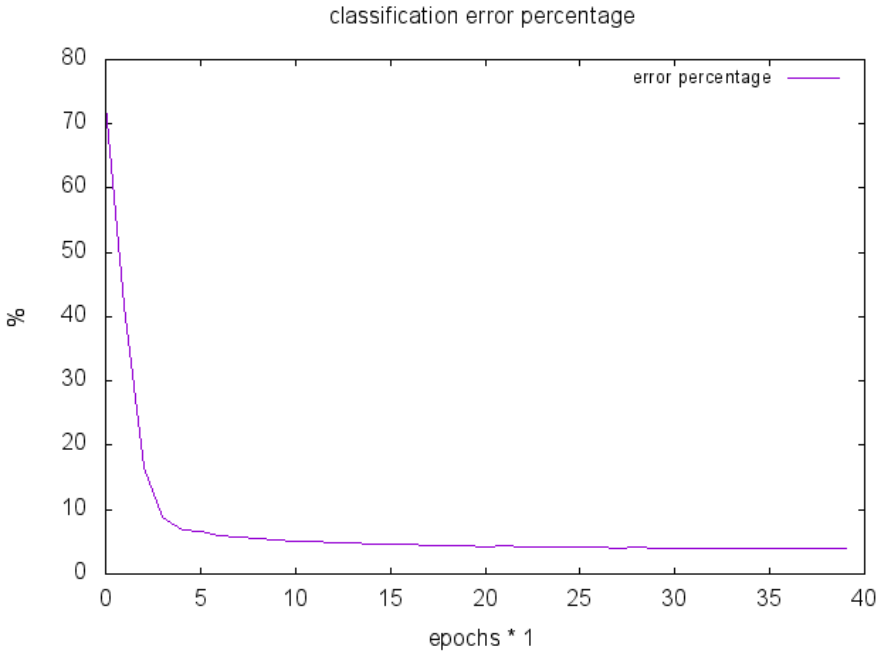


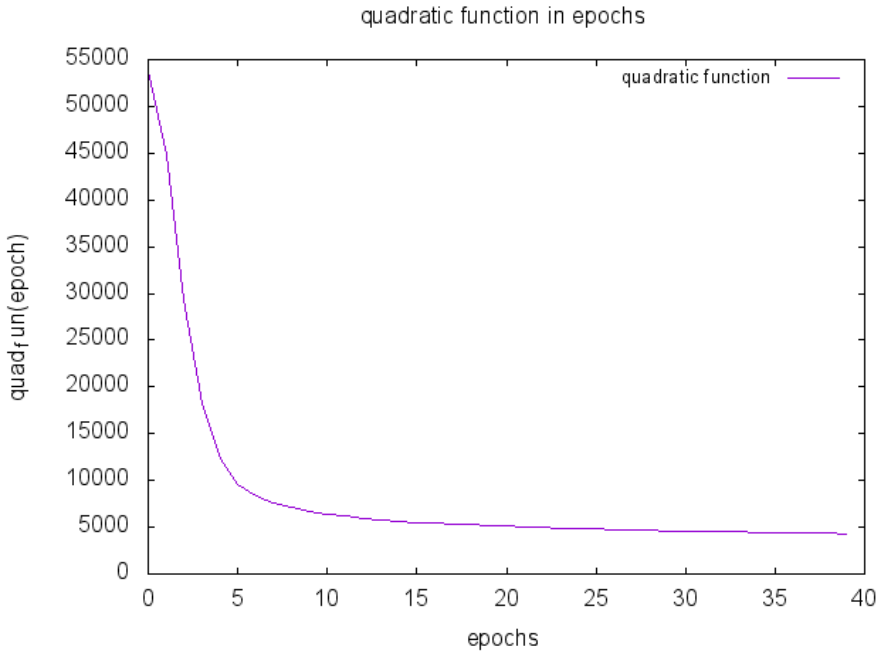


Macierz konfuzji dla zbioru testowego:
Klasa: 0 TP: 946 FN: 34 FP: 54 TN: 8966
Klasa: 1 TP: 1107 FN: 28 FP: 51 TN: 8814
Klasa: 2 TP: 943 FN: 89 FP: 89 TN: 8879
Klasa: 3 TP: 883 FN: 127 FP: 72 TN: 8918
Klasa: 4 TP: 895 FN: 87 FP: 77 TN: 8941
Klasa: 5 TP: 709 FN: 183 FP: 45 TN: 9063
Klasa: 6 TP: 921 FN: 37 FP: 85 TN: 8957
Klasa: 7 TP: 910 FN: 118 FP: 35 TN: 8937
Klasa: 8 TP: 902 FN: 72 FP: 204 TN: 8822

Klasa: 9 TP: 933 FN: 76 FP: 139 TN: 8852
Suma: TP: 9149 FN: 851 FP: 851
Czas uczenia: 192.194s

(P8) Dane jak w P7 ale zastosowano ekstrakcję cech HOG przez co perceptron miał postać 81-15-10 oraz nie normalizowano danych wejściowych dla zbioru treningowego.

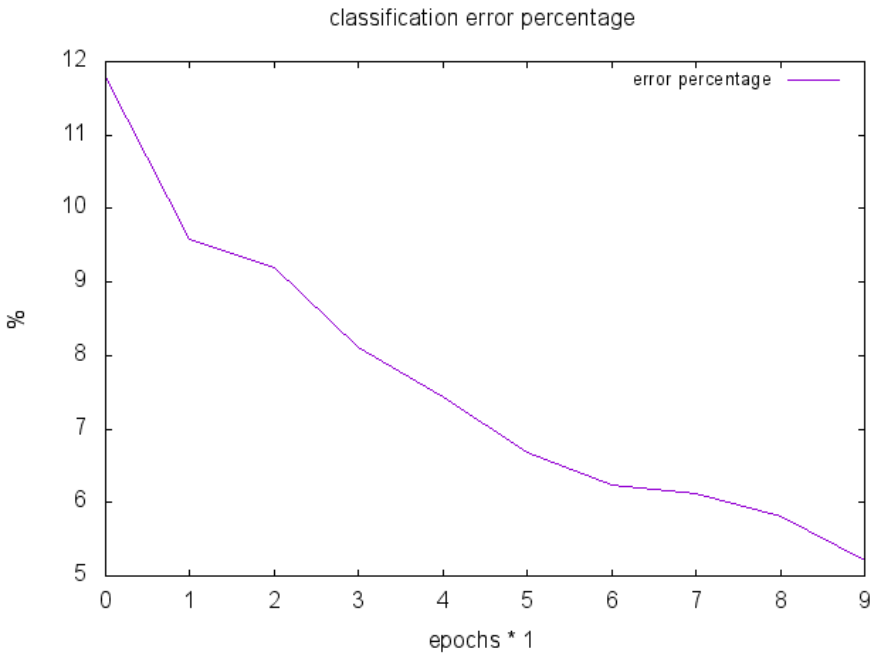


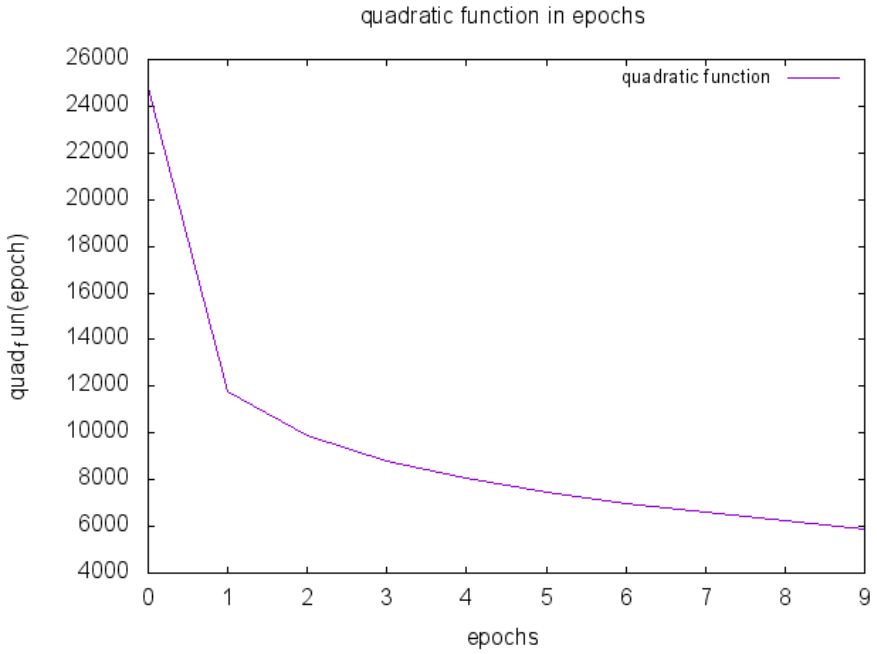


Macierz konfuzji dla zbioru testowego:
Klasa: 0 TP: 969 FN: 11 FP: 19 TN: 9001
Klasa: 1 TP: 1108 FN: 27 FP: 19 TN: 8846

Klasa: 2 TP: 1008 FN: 24 FP: 35 TN: 8933
Klasa: 3 TP: 978 FN: 32 FP: 70 TN: 8920
Klasa: 4 TP: 937 FN: 45 FP: 44 TN: 8974
Klasa: 5 TP: 867 FN: 25 FP: 22 TN: 9086
Klasa: 6 TP: 936 FN: 22 FP: 18 TN: 9024
Klasa: 7 TP: 989 FN: 39 FP: 45 TN: 8927
Klasa: 8 TP: 929 FN: 45 FP: 49 TN: 8977
Klasa: 9 TP: 926 FN: 83 FP: 32 TN: 8959
Suma: TP: 9647 FN: 353 FP: 353
Czas uczenia: 34.176s
Czas ekstrakcji cech: 5.214s

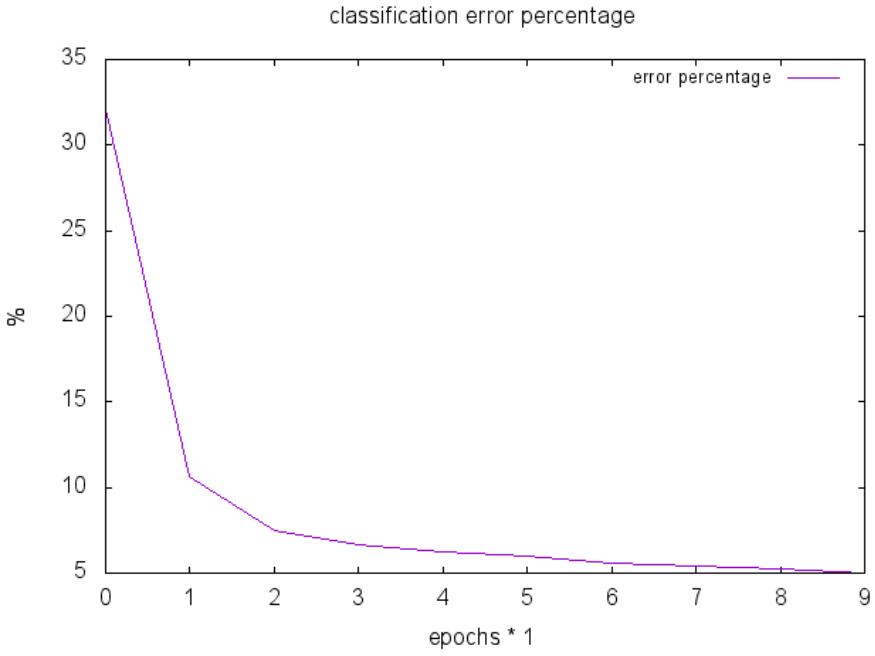
(P9) Dane jak w P7 ale w warswie ukrytej użyto 150 neuronów.

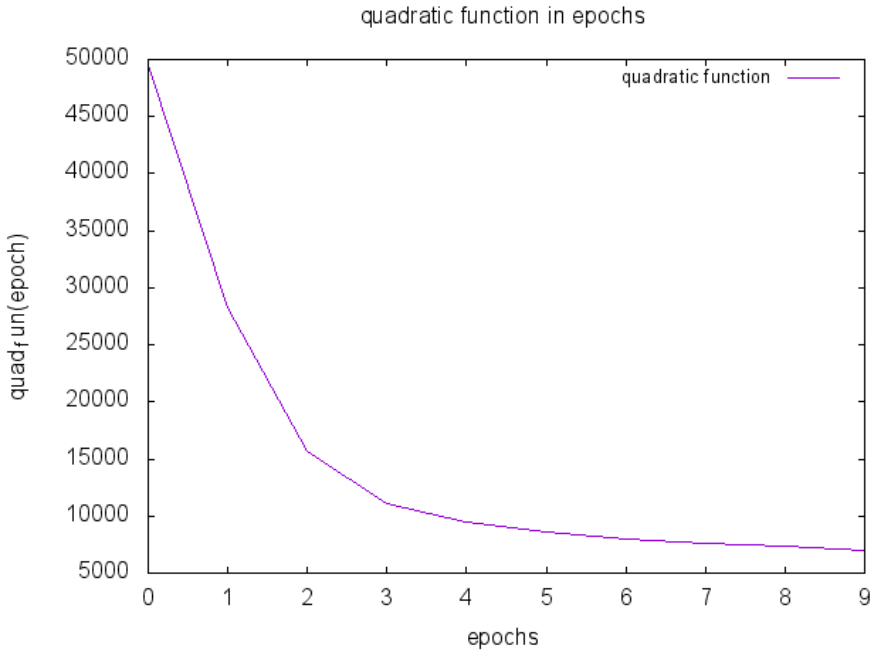




Macierz konfuzji dla zbioru testowego:
Klasa: 0 TP: 967 FN: 13 FP: 55 TN: 8965
Klasa: 1 TP: 1105 FN: 30 FP: 13 TN: 8852
Klasa: 2 TP: 966 FN: 66 FP: 55 TN: 8913
Klasa: 3 TP: 956 FN: 54 FP: 88 TN: 8902
Klasa: 4 TP: 930 FN: 52 FP: 57 TN: 8961
Klasa: 5 TP: 786 FN: 106 FP: 26 TN: 9082
Klasa: 6 TP: 917 FN: 41 FP: 52 TN: 8990
Klasa: 7 TP: 972 FN: 56 FP: 44 TN: 8928
Klasa: 8 TP: 931 FN: 43 FP: 103 TN: 8923
Klasa: 9 TP: 932 FN: 77 FP: 45 TN: 8946
Suma: TP: 9462 FN: 538 FP: 538
Czas uczenia: 657.515s

(P10) Dane jak w P8 ale użyto 150 neuronów w warstwie ukrytej.

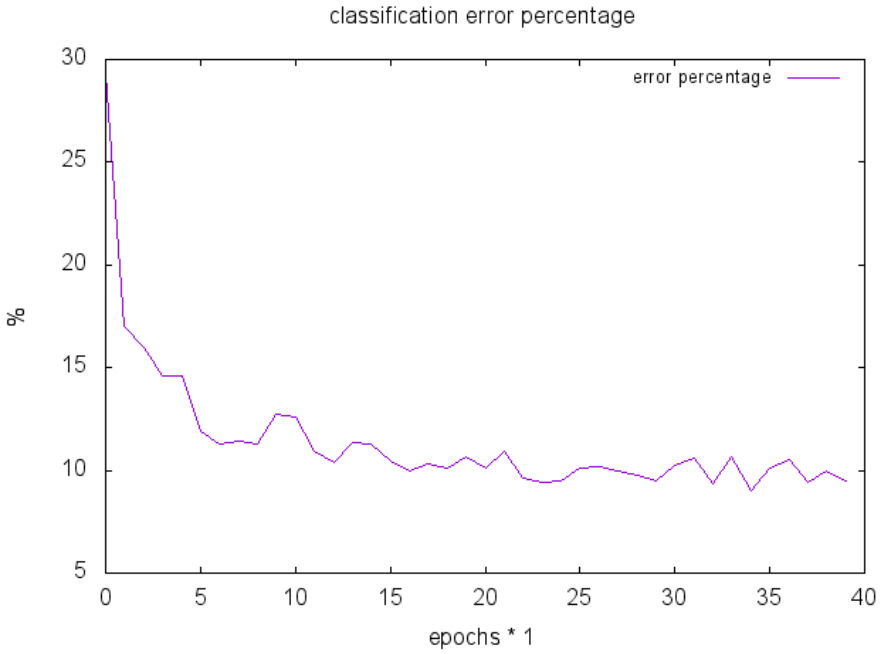


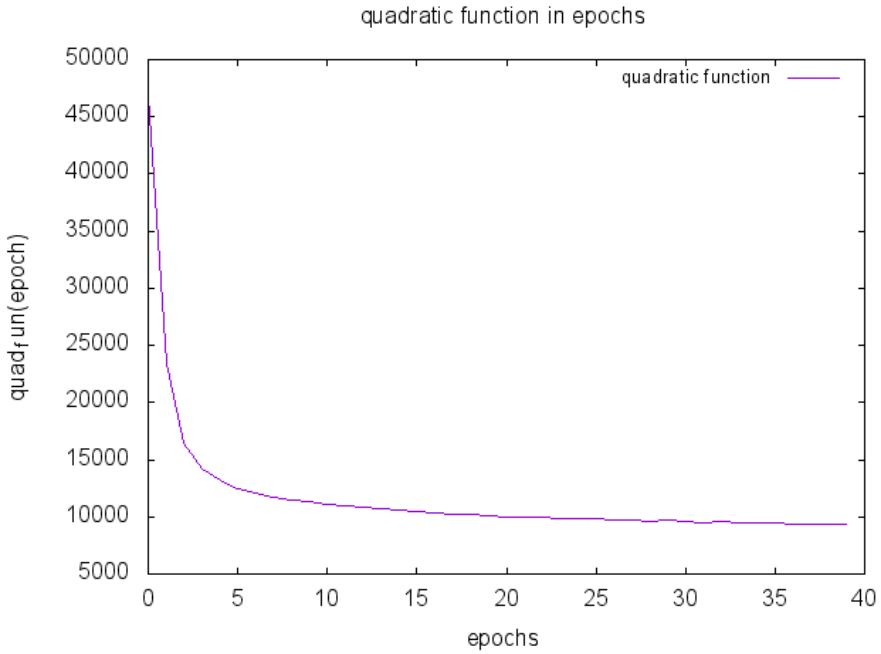


Macierz konfuzji dla zbioru testowego:
Klasa: 0 TP: 967 FN: 13 FP: 29 TN: 8991
Klasa: 1 TP: 1105 FN: 30 FP: 20 TN: 8845
Klasa: 2 TP: 1005 FN: 27 FP: 60 TN: 8908
Klasa: 3 TP: 972 FN: 38 FP: 63 TN: 8927
Klasa: 4 TP: 933 FN: 49 FP: 51 TN: 8967
Klasa: 5 TP: 863 FN: 29 FP: 28 TN: 9080
Klasa: 6 TP: 929 FN: 29 FP: 15 TN: 9027
Klasa: 7 TP: 960 FN: 68 FP: 46 TN: 8926

Klasa: 8 TP: 920 FN: 54 FP: 66 TN: 8960
Klasa: 9 TP: 917 FN: 92 FP: 51 TN: 8940
Suma: TP: 9571 FN: 429 FP: 429
Czas uczenia: 62.902s
Czas ekstrakcji cech: 5.224s

(P11) Dane jak w P7 ale użyto do testów danych treningowych.

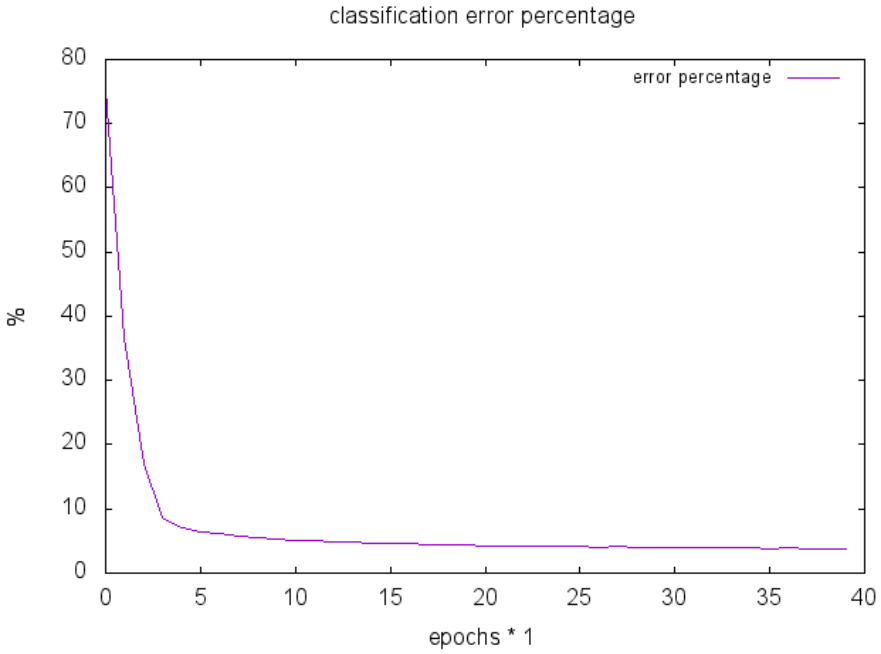


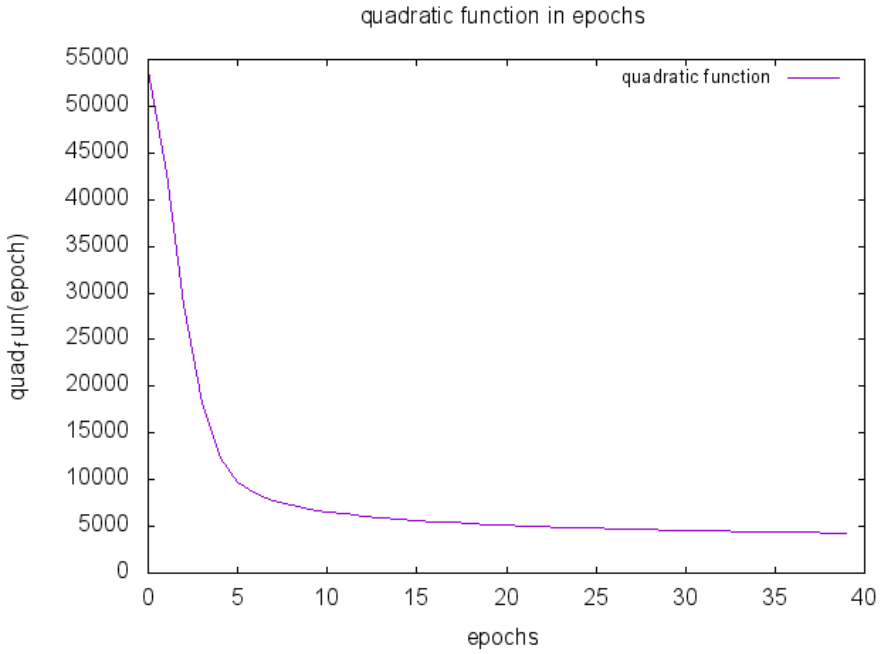


Macierz konfuzji dla zbioru treningowego:
Klasa: 0 TP: 5633 FN: 290 FP: 358 TN: 53719
Klasa: 1 TP: 6270 FN: 472 FP: 215 TN: 53043

Klasa: 2 TP: 5312 FN: 646 FP: 506 TN: 53536
Klasa: 3 TP: 5551 FN: 580 FP: 873 TN: 52996
Klasa: 4 TP: 5267 FN: 575 FP: 607 TN: 53551
Klasa: 5 TP: 4742 FN: 679 FP: 793 TN: 53786
Klasa: 6 TP: 5508 FN: 410 FP: 318 TN: 53764
Klasa: 7 TP: 5495 FN: 770 FP: 232 TN: 53503
Klasa: 8 TP: 5183 FN: 668 FP: 948 TN: 53201
Klasa: 9 TP: 5335 FN: 614 FP: 854 TN: 53197
Suma: TP: 54296 FN: 5704 FP: 5704
Czas uczenia: 217.970s

(P12) Dane jak w P8 ale użyto do testów danych treningowych.





Macierz konfuzji dla zbioru treningowego:
Klasa: 0 TP: 5826 FN: 97 FP: 124 TN: 53953
Klasa: 1 TP: 6560 FN: 182 FP: 116 TN: 53142
Klasa: 2 TP: 5746 FN: 212 FP: 271 TN: 53771
Klasa: 3 TP: 5870 FN: 261 FP: 323 TN: 53546
Klasa: 4 TP: 5554 FN: 288 FP: 310 TN: 53848
Klasa: 5 TP: 5257 FN: 164 FP: 147 TN: 54432
Klasa: 6 TP: 5805 FN: 113 FP: 139 TN: 53943
Klasa: 7 TP: 5978 FN: 287 FP: 277 TN: 53458
Klasa: 8 TP: 5504 FN: 347 FP: 338 TN: 53811
Klasa: 9 TP: 5604 FN: 345 FP: 251 TN: 53800
Suma: TP: 57704 FN: 2296 FP: 2296
Czas uczenia: 33.209s
Czas ekstrakcji cech: 5.573s

(P13) Drzewo decyzyjne Apache Spark. Dane wejściowe: maxDepth = 15, maxBins = 256, impurity = "gini"

Macierz konfuzji dla zbioru testowego:
Klasa: 0 TP: 922 FN: 58 FP: 81 TN: 8939
Klasa: 1 TP: 1084 FN: 51 FP: 49 TN: 8816
Klasa: 2 TP: 920 FN: 112 FP: 115 TN: 8853
Klasa: 3 TP: 897 FN: 113 FP: 171 TN: 8819
Klasa: 4 TP: 846 FN: 136 FP: 110 TN: 8908
Klasa: 5 TP: 773 FN: 119 FP: 89 TN: 9019
Klasa: 6 TP: 882 FN: 76 FP: 85 TN: 8957
Klasa: 7 TP: 889 FN: 139 FP: 83 TN: 8889
Klasa: 8 TP: 815 FN: 159 FP: 170 TN: 8856

Klasa: 9 TP: 876 FN: 133 FP: 143 TN: 8848
Suma: TP: 8904 FN: 1096 FP: 1096
Czas uczenia: 64.862s
Błąd drzewa decyzyjnego: 0.1096
Ilość węzłów: 4083

5. Dyskusja i wnioski

5.1. Część badawcza 1

Dwie pierwsze badane konfiguracje P1 oraz P2 miały odpowiedzieć na pytanie w jaki sposób parametry perceptronu wpływają na jakoś procesu uczenia. Na podstawie wyników można łatwo stwierdzić, że małe wartości learning rate dają lepsze wyniki. Przyczyną tego jest dość duża ilość danych testowych - w jednej epoce przez perceptron propagowane jest wstecz 150 krotek z danymi. Dodatkowo ustawiono dużą liczbę epok: 9000. W przypadku małych wartość learning rate dane były już sklasyfikowane około 3500 epoki.
Konfiguracja P3 miała pokazać wpływ liczby neuronów w warstwie ukrytej na jakość uczenia. Dla zwiększonej do 100 liczby neuronów warstwy ukrytej proces uczenia przebiegał znacznie szybciej w porównaniu do przypadku P1 gdzie zastosowano 12 neuronów. Zwiększenie liczby neuronów pozwala sieci rozbić problem klasyfikacji na bardziej nieliniowe przedziały poprawnych odpowiedzi dla podanych wejść.
Konfiguracja P4 została przeprowadzona tylko na zbiorze treningowym. Pozwala ona zauważyć że w każdym z przypadków nie udaje sie poprawnie zakwalifikować jednego kwiatka gatunku iris-versicolor.
Konfiguracje P5 i P6 zostały przeprowadzone na drzewie decyzyjnym. Przypadek P5 pokazuje że algorytm dla tego zbioru danych radzi sobie równie sprawnie co perceptron oraz szybciej uzyskuje wyniki. Dodatkowo w przypadku działania tylko na zbiorze testowym drzewo poprawnie klasyfikuje wszystkie dane.

5.2. Część badawcza 1

Konfiguracja P7 miała na celu sprawdzenie jakości uczenia dla dość małej liczby neuronów w warswie ukrytej gdzie zastosowano ich 15. Nauka trwa aż do osiągnięcia progu błędu klasyfikacji na poziomie 8% w 18 epoce po czym następują jedynie niewielkie wahania aż do 40 epoki.
W przykładzie P8 pokazano że zastosowanie ekstrakcji cech na zbiorze danych pozwala znacznie poprawić jakość procesu uczenia oraz szybkość całego procesu. Błąd klasyfikacji szybko osiąga 5% po czym utrzymuje tą wartość bez wahań. Również czas całego procesu jest prawie pięciokrotnie krótszy, po ekstrakcji cech mamy jedynie 81 danych wejściowych w jednej krotce zamiast 784.
Przykłady P9 i P10 pokazują że zwiększenie liczby neuronów pozwala poprawić jakość uczenia jednak zmiany te są bardziej znaczące niż w przypadkach z irysami. Podobnie jak w przykładzie P8 zastosowanie HOG pozwala poprawić

wyniki jeszcze bardziej.
Wyniki z przykłady P11 i P12 są bardzo podobnie do P7 i P8 mimo że do testów użyto zbioru treningowego. Zastosowanie tych samych danych w testach nie poprawiło znacząco wyników prezentowanych w macierzach konfuzji.
Użycie drzewa decyzyjnego to tego typu problemu nie jest już tak efektywnie jak w przypadku irysów. W przykładzie P13 uzyskano błąd klasyfikacji ponad 10% co jest dobrym wynikiem jednak jest to wynik gorszy od wyniku perceptronu.

Literatura

[1] Yann LeCun, Corinna Cortes and Chris Burges *MNIST handwritten digit database*

[2] UCI Machine Learning Repository *Iris Data Set*