Laboratorium

Sieci neuronowe

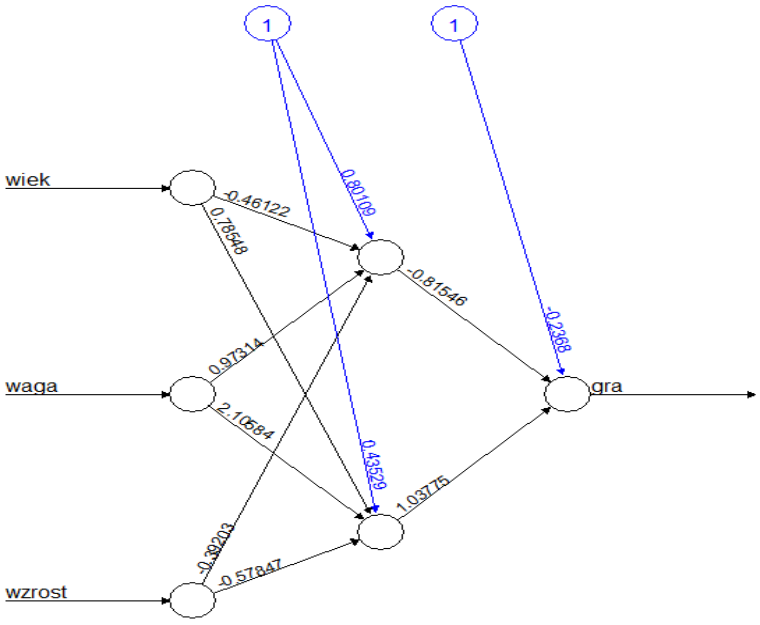
Na dzisiejszych laboratoriach przećwiczymy w praktyce jak tworzyć i stosować sztuczne sieci neuronowe.

Gra w siatkówkę

Zapoznamy się z działaniem sieci neuronowej na prostym przykładzie. Mamy małą bazę danych ludzi, których zachęcaliśmy do gry w siatkówkę na plaży (a przy okazji popytaliśmy o parametry ich organizmu). Część ludzi zgodziła się zagrać, a część nie.

wiek	waga	wzrost	gra
23	75	176	TRUE
25	67	180	TRUE
28	120	175	FALSE
22	65	165	TRUE
46	70	187	TRUE
50	68	180	FALSE
48	97	178	FALSE

Załóżmy, że wytrenowaliśmy sieć neuronową o następującej strukturze:



Error: 0.40071 Steps: 114

Sieć neuronowa, niczym mały sztuczny mózg, miała nauczyć się rozpoznawać, które osoby będą chciały grać w siatkówkę. Sprawdziliśmy jak sieć poradziła sobie z naszymi siedmioma osobami. Otrzymaliśmy wynik :

[1,]	0.79852846757
[2,]	0.80094344704
[3,]	-0.01451814786
[4,]	0.80092650708
[5,]	0.80094353050
[6,]	0.80094353051
[7,]	0.01518550709

Widać, że gdyby zaokrąglić wartości do 1 (TRUE) i 0 (FALSE) to nasza sieć wypadła całkiem nieźle. Nie zgadła jedynie rekordu 6.

Pytanie: jak obliczone zostały powyższe wartości?
Sieć neuronowa dostaje trzy dane na wejściu. Wartości liczbowe są mnożone przez wagi na strzałkach i przekazywane kolejnym neuronom, które sumują wszystko co dostają. Dodane są też do tego wartości dodatkowe (bias) zaznaczone na niebiesko. Zsumowane wartości przepuszczone są przez funkcję aktywacji (fct.act) wynoszącą w tym przypadku $f_{act}(x) = \frac{1}{1+e^{-x}}$ po czym neuron przesyła je dalej.

Zadanie 1

Napisz prostą funkcję, która sprawdzi czy sieć działa dobrze tj. funkcja `forwardPass` na input dostanie wiek, wagę, wzrost, a na output zwróci liczbę przewidującą granie w siatkówkę. Wagi z sieci pobierz z rysunku w tym pdf (dużo przepisywania)
Uwaga! W powyższym modelu funkcja aktywacji nie działa na neuronie output, tylko na dwóch neuronach ukrytych (hidden). Wystarczy uzupełnić miejsca z wielokropkiem.

```
def forwardPass(wiek, waga, wzrost):
     hidden1 = …
     hidden1_po_aktywacji = …
     hidden2 = …
     hidden2_po_aktywacji = …
     output =  …
     return output
```

Sprawdź jej działanie dla dwóch wybranych przez ciebie rekordów np.

forwardPass(23,75,176) = 0.798528

W razie problemów zachęcam do skorzystania z Google / Youtube (wyszukanie: sieć neuronowa, perceptron, neural network) lub z wykładów.

Zadanie 2

Zadanie to przypomina zadania z laboratoriów o klasyfikacji. Tym razem klasyfikatorem dla bazy danych irysów będzie sieć neuronowa.
Do stworzenia i wytrenowania sieci neuronowej MLP (Multilayer perceptron) użyjemy paczki sklearn. Zadanie należy rozwiązywać z wykorzystaniem pomocy z internetu, polecam np. samoczuek:

- https://python-course.eu/machine-learning/neural-networks-with-scikit.php (część "Complete Iris Dataset Example"). Lub:
- https://www.kaggle.com/code/avk256/iris-with-mlpclassifier/notebook

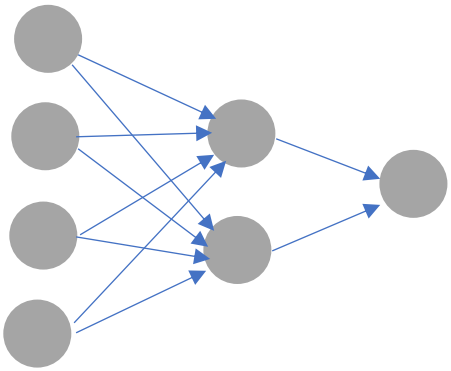
Co należy zrobić?

a) Załadować paczkę `sklearn` , bazę danych z irysami I podzielić irysy na część testową i treningową używając komendy: `train_test_split` (70% / 30%).

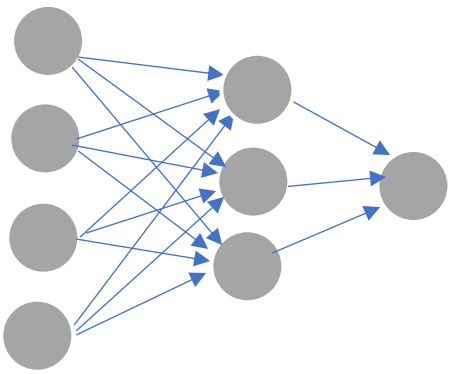
b) Sieć neuronowa nie akceptuje napisów, jedynie liczby. Na szczęście, podążając za samouczkiem `train_labels` i `test_labels` są skonwertowane na liczby. Jakie to liczby? Jakim napisom odpowiadają?

c) Możesz przeskalować dane, ale ten punkt nie jest obowiązkowy.

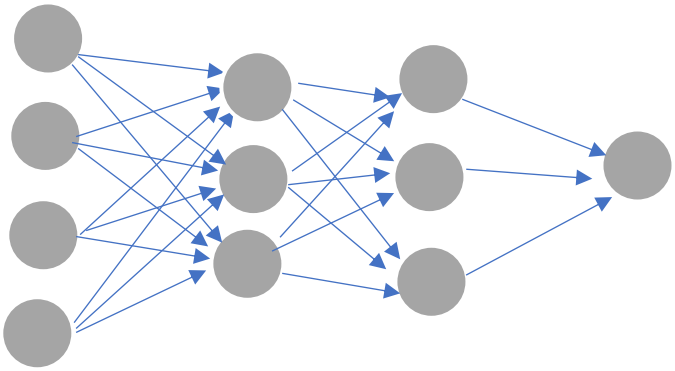
d) Skonstruuj I wytrenuj model sieci neuronowej z czteroneuronową warstwą wejściową (bo są cztery pomiary irysów), jedną ukrytą warstwą z dwoma neuronami i warstwą wyjściową z jednym neuronem decydującym o gatunku irysa. Obrazek ilustrujący topologię sieci:



e) Dokonaj ewaluacji sieci na zbiorze testowym. Wyświetl jej dokładność.
f) Sprawdź czy model sieci z trzema neuronami działa lepiej.



g) Sprawdź czy model z dwiema warstwami neuronowymi, po 3 neurony każda, działa lepiej.

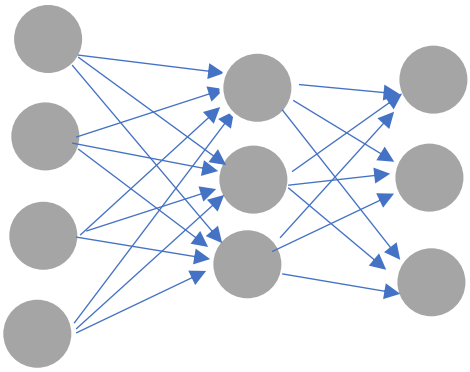


Podczas pokazywania powiedz, która architektura z powyższych trzech sieci wypadła najlepiej (najwyższe accuracy na zbiorze testowym). Wystarczy porównać trzy liczby.

Uwaga! Jeśli pojawiają się ostrzeżenia związane z „Stochastic Omptimizer Convergence", można spróbować dodać więcej iteracji do uczenia algorytmu (1000, 2000, 3000). Jednak można też to ostrzeżenie zignorować.

Zadanie 3

Innym sposobem odgadywania irysów jest przyporządkowanie każdej wartości klasy jednemu neuronowi wyjściowemu (np. górny to setosa, środkowy versicolor, dolny virginica). Gdy neuron górny zwraca 1, a pozostałe coś bliższego zeru, to znaczy, że sieć rozpoznała irysa jako setosę. Spróbuj zaimplementować ten model i sprawdzić czy działa równie sprawnie jak poprzednie. Możesz skorzystać ze źródeł internetowych. Ważne jest tutaj odpowiednie przygotowanie bazy danych (4 kolumny wejściowe + 3 kolumny wyjściowe binarne).



Zadanie 4

Dokonaj klasyfikacji (diagnoza cukrzycy) za pomocą prostej jednokierunkowej sieci neuronowej, czyli wielowarstwowego perceptronu (Multilayer Perceptron, MLP).

Pomocne linki:

- https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html (dokumentacja i przykłady)
- https://www.pluralsight.com/guides/machine-learning-neural-networks-scikit-learn (samouczek)
- https://analyticsindiamag.com/a-beginners-guide-to-scikit-learns-mlpclassifier/ (samouczek)

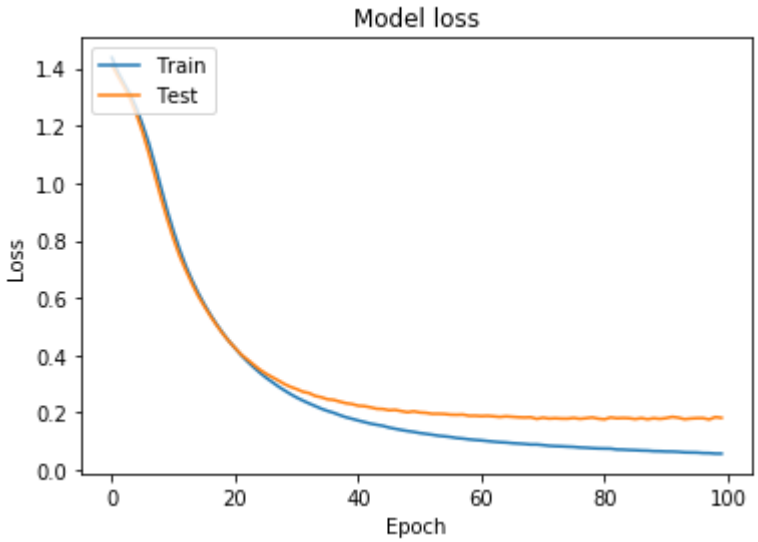
a) Podziel zbiór danych na testowy (30%) i treningowy (70%).
b) Zbuduj model sieci o dwóch warstwach ukrytych – pierwsza ma 6 neuronów, a druga 3 neurony, z funkcją aktywacji ReLU.
c) Wytrenuj model na zbiorze treningowym na maksymalnie 500 iteracjach.
d) Dokonaj ewaluacji na zbiorze testowym: podaj dokładność i macierz błędu.
e) Czy sieć neuronowa poradziła sobie lepiej niż klasyfikatory z poprzednich laboratoriów? Spróbuj przetestować inną sieć (np. zmień warstwy i neurony) by zobaczyć, czy będzie działała lepiej.

Zadanie 5

Wykonaj zadanie 4 wykorzystując inną paczkę: keras. Jest to bardziej zaawansowana biblioteka dedykowana sieciom neuronowym.

a) Wykonaj kroki b-d z poprzedniego zadania.

b) Nanieś na wykres krzywą błędu (loss curve) dla zbioru treningowego i zbioru walidującego (u nas zbiorem walidującym może być zbiór testowy). Wykres powinien wyglądać mniej więcej tak:



c) Czy trenowanie powinno się przerwać w pewnym momencie, by uniknąć przeuczenia? Jak odczytać to z wykresu? A może mamy do czynienia z niedouczeniem? Podpowiedzi:
   - Wykład
   - https://machinelearningmastery.com/learning-curves-for-diagnosing-

machine-learning-model-performance/
    - https://rstudio-conf-2020.github.io/dl-keras-tf/notebooks/learning-curve-diagnostics.nb.html

d) Przetestuj jak sieć będzie uczyła się z różnymi optimizerami (adam, itd.) i różnymi funkcjami aktywacji (reLU, itp.). Wyświetl i porównaj wykresy krzywych uczenia się dla różnych konfiguracji.

e) Dodatkowo, w miarę możliwości, zwizualizuj sieć neuronową w formie grafu, obrazka. Jeśli się da to najlepiej z uwzględnieniem poszczególnych neuronów, krawędzi i wag. Być może przydatne będą linki:
    - https://datascience.stackexchange.com/questions/12851/how-do-you-visualize-neural-network-architectures
    - https://towardsdatascience.com/visualizing-artificial-neural-networks-anns-with-just-one-line-of-code-b4233607209e