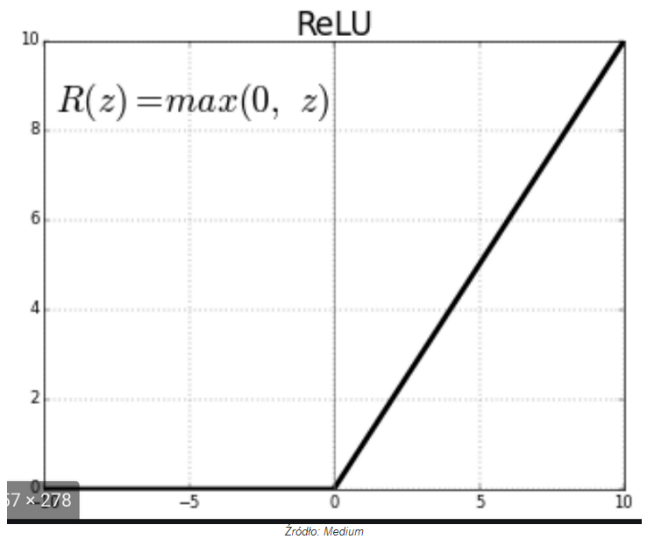
# Opis podjętego do rozwiązania problemu.

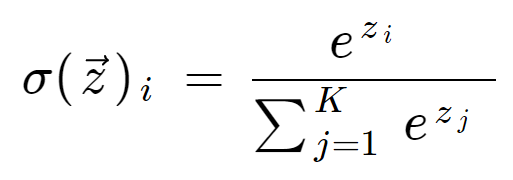
Pliki train.csv oraz test.csv zawierają obrazki ręcznie narysowanych cyfr od 0 do 9. Każdy obraz zawiera łącznie 784 piksele (28x28). Każdy piksel zawiera liczbę 0-255 definiującą jego wysycenie w skali szarości (im mniejsza liczba tym piksel jest jaśniejszy). Piksele w piku są oznaczone według następującego przykładu: wartość kryjąca się pod pixel31 oznacza poziom jasności piksela znajdującego się w macierzy 28x28 w 4 kolumnie i 2 wierszu. Naszym celem jest zbudowanie wielowarstwowej, jednokierunkowej sieci neuronalnej klasyfikującej kolejne obrazy do odpowiednich cyfr.

# Funkcje Aktywacji

1. ReLU - jedna z najpopularniejszych funkcji aktywacji; rektyfikowana jednostka liniowa. Neuron z funkcją aktywacji ReLU przyjmuje dowolne wartości rzeczywiste jako swoje wejście (a), ale aktywuje się tylko wtedy, gdy te wejście (a) są większe niż 0.



1. Softmax - Funkcja aktywacji Softmax odwzorowuje nieznormalizowane dane wejściowe na zestaw potęgowanych i znormalizowanych prawdopodobieństw. W kontekście uczenia maszynowego funkcja aktywacji Softmax jest używana w problemach klasyfikacji wieloklasowej w celu uogólnienia regresji logistycznej, gdy istnieją więcej niż dwie klasy wyników.



# Propagacja wsteczna

1. W celu optymalizacji zestawu wag użyliśmy pochodnych funkcji aktywacji. Dla danego wektora uczącego obliczamy odpowiedź sieci (warstwa po warstwie). Każdy neuron wyjściowy oblicza swój błąd, oparty na różnicy pomiędzy obliczoną odpowiedzią ý oraz poprawną odpowiedzią *y*. Błędy propagowane są do wcześniejszych warstw.

## **Problemy uczenia metodą wstecznej propagacji błędu**

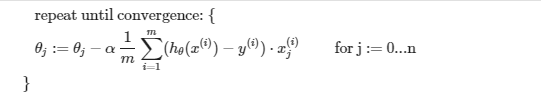
* Minima lokalne funkcji kosztu. W odróżnieniu od funkcji kosztu dla problemu sieci liniowych w przypadku sieci nieliniowych funkcja kosztu ma zazwyczaj wiele lokalnych minimów. Algorytm wstecznej propagacji błędu jako algorytm spadku gradientowego nie jest odporny na ich występowanie.

# Propagacja w przód

Dane wejściowe są przesyłane w kierunku do przodu przez sieć. Każda warstwa ukryta przyjmuje dane wejściowe, przetwarza je zgodnie z funkcją aktywacji i przechodzi do kolejnej warstwy.

# Zanikający gradient

Błąd propagowany od warstwy wyjściowej zanika przy niższych warstwach funkcje ograniczone „nasycają się” i posiadają niezerowy gradient tyko w wąskim przedziale aktywacji bliskim 0

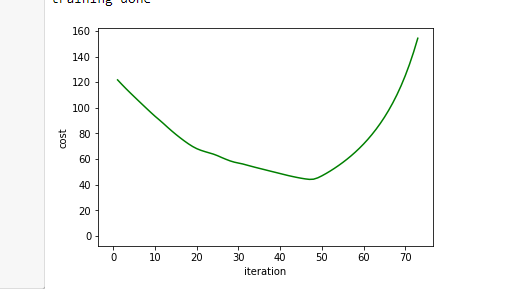


Dla wielu zmiennych (w naszym wypadku 784) wejściowych notacja dla naszej funkcji hipotetycznej wygląda w następujący sposób (x0 = 1)

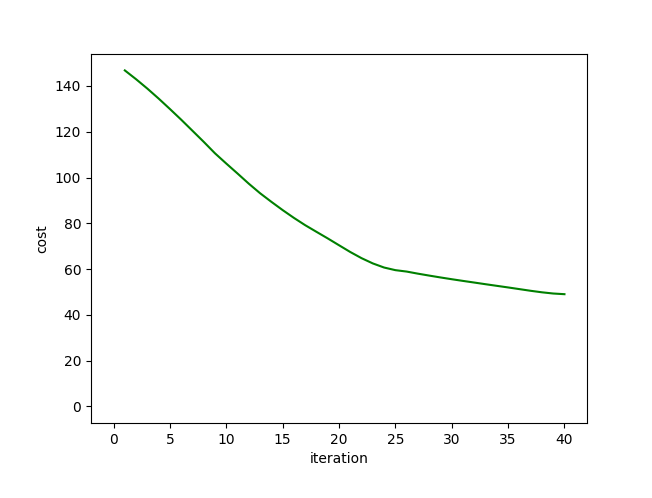


# Funkcja kosztu

To błąd średniokwadratowy różnicy wartości predykowanej i wyjściowej. Naszym celem było takie dobranie współczynnika alfa, liczby iteracji oraz warstw, żeby wartość w minimum lokalnym była jak najniższa.



Rysunek 1. Funkcja Kosztu dla 75 iteracji

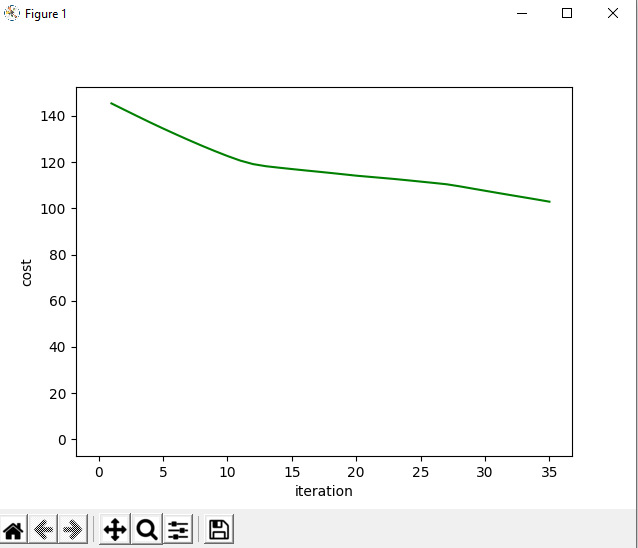


Rysunek 2. Funkcja kosztu dla numeru iteracji prezentowanego w programie

# Liczba warstw

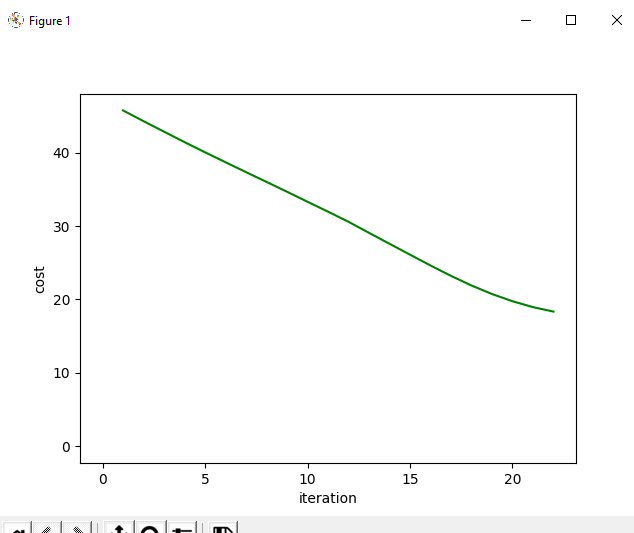
Początkowo dla naszej sieci zastosowaliśmy 6 warstwowy model, który spotkał się z dosyć marnym współczynnikiem predykcji. Zmniejszając liczbę warstw wzrastała nam wartość błędu, natomiast zwiększając nieracjonalnie doprowadzaliśmy do przetrenowania modelu -> funkcja kosztu była stała.

layers\_dims = [784,200,100,50,10]



Rysunek 3. Wykres funkcji kosztu dla 5 warstwowego modelu

# Współczynnik momentum(alfa)

Współczynnik uczenia wpływa na szybkość uczenia poprzez bezpośrednie oddziaływanie na gradient zmiany. Zbyt duża jego wartość powoduje poruszanie się po wierzchołkach płaszczyzny błędu i pomijanie zagłębień minimami. Natomiast zmniejszając współczynnik możemy zaobserwować wolniejsze osiąganie minimum przez funkcję kosztu.

Rysunek . Funkcja kosztu dla modelu 9-warstwowego

# Podsumowanie

Ponieważ budowanie tej sieci było ograniczone brakiem korzystania z bibliotek uczenia maszynowego oraz zgodnie z badaniami stosowanie konwolucyjnych sieci neuronowych jest dużo lepsze od klasycznych zdajemy sobie sprawę z niedoskonałości naszego modelu. Po wielu próbach udało nam się osiągnąć zadowalające nas minimum na funkcji kosztu. Jednak optymalizacja funkcji kosztu nie miała kompletnie żadnego odzwierciedlenia na błąd predykcji. Do predykcji liczb pliku testowego zastosowaliśmy te same funkcje propagacji w przód wyłączając propagację wsteczną i naszym zdaniem na wyjściu powinniśmy otrzymać mniej lub bardziej poprawnie sklasyfikowane rekordy.

# Przegląd literatury dotyczący naszego problemu

* 1. 68 technik uczenia maszynowego są podzielone na 6 szeroko pojętych kategorii:

■ klasyfikatory liniowe

■ algorytm k najbliższych sąsiadów

■ wzmocnione kikuty

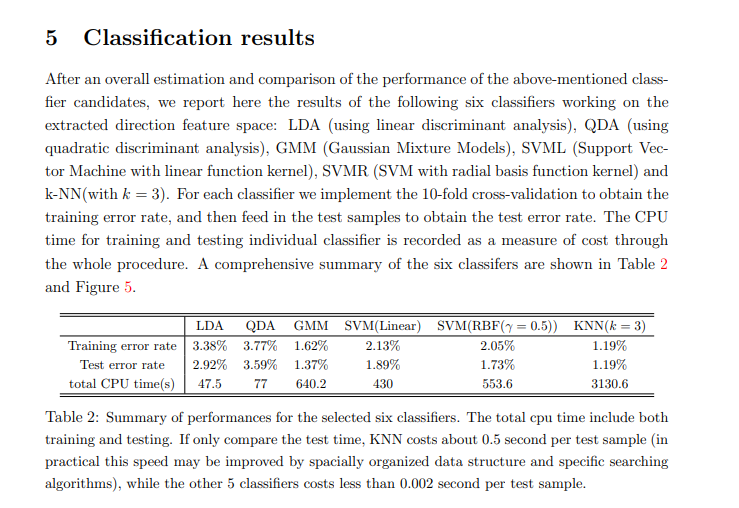
■ klasyfikatory nieliniowe

■ Maszyna wektorów nośnych (SVMs)

■ sieci neuronowe (bez struktury splotowej)

■ sieci splotowe

* 1. Klasyfikatory sieci neuronowych zwykle działają znacznie lepiej niż inne typy klasyfikatorów. W szczególności struktura splotu w sieciach neuronowych zapewnia doskonałą wydajność klasyfikacji. W rzeczywistości rekordową wydajność, około 0,27% wskaźnika błędów lub 27 błędów w pełnym zestawie 10000 testów, osiąga komitet sieci splotowych (z odkształceniem sprężystym przy zwiększaniu zbioru uczącego) . Poza technikami sieci neuronowych metody k-najbliższego sąsiada również generowały niskie współczynniki błędów, a następnie wirtualne maszyny SVM. Należy zauważyć, że w obu przypadkach wstępne przetwarzanie jest niezbędne do osiągnięcia sukcesu.



* 1. Jednak znaczną poprawę można osiągnąć tylko wtedy, gdy klasyfikatory składowe działają lepiej niż przypadek [5], podczas gdy sześć klasyfikatorów, na których się skupiamy, ma zwykle niski współczynnik błędu..
  2. – Dokladnosc tego modelu to: **0.99. Ten model wykorzystuje konwolucyjne sieci neuronowe.**
  3. Najlepszy poziom błędu testu wynoszący około 0,7% uzyskuje się przez łączenie wiele klasyfikatorów sieci neuronowych (Boosted LeNet 4). Jednym z głównych wyzwań w rozpoznawaniu odręcznych cyfr jest klasa wewnętrzna wariancja, ponieważ ludzie nie zawsze piszą tę samą cyfrę w dokładnie ten sam sposób. Klasyfikatory oparte na modelach generatywnych (Multivariate Gaussian, Mixture Gaussian) i modele dyskryminacyjne są stosowane do porównania wydajność zarówno pod względem dokładności, jak i szybkości.

Dokładność tego modelu: 0,99. Ten mod (Wyniki klasyfikacji Po ogólnej ocenie i porównaniu wydajności wyżej wymienionych kandydatów klasyfikujących, przedstawiamy tutaj wyniki następujących sześciu klasyfikatorów pracujących na wyodrębnionej przestrzeni cech kierunkowych: LDA (z wykorzystaniem liniowej analizy dyskryminacyjnej), QDA ( wykorzystując kwadratową analizę dyskryminacyjną), GMM (Gaussian Mixture Models), SVML (Support Vector Machine z jądrem funkcji liniowej), SVMR (SVM z jądrem radialnej funkcji bazowej) i k-NN (z k = 3). 10-krotna walidacja krzyżowa w celu uzyskania wskaźnika błędu uczenia, a następnie wprowadzenie próbek testowych w celu uzyskania wskaźnika błędu testu. Czas procesora na uczenie i testowanie poszczególnych klasyfikatorów jest rejestrowany jako miara kosztów w trakcie całej procedury. podsumowanie sześciu klasyfikatorów przedstawiono w Tabeli 2 i na Rysunku 5. LDA QDA GMM SVM (Liniowa) SVM (RBF (γ = 0,5)) KNN (k = 3) Wskaźnik błędu treningu 3,38% 3,77% 1,62% 2,13% 2,05% 1,19 % Wskaźnik błędu testu 2,92% 3,59% 1,37% 1,89% 1,73% 1,19% całkowity czas procesora (s) 47,5 77 640,2 430 553,6 3130,6 Tabela 2: Podsumowanie wydajności dla wybranych sześciu klasyfikatorów. Całkowity czas procesora obejmuje zarówno szkolenie, jak i testowanie. Porównując tylko czas testu, KNN kosztuje około 0,5 sekundy na próbkę testową (w praktyce szybkość tę można poprawić dzięki przestrzennie zorganizowanej strukturze danych i specjalnym algorytmom wyszukiwania), podczas gdy pozostałe 5 klasyfikatorów kosztuje mniej niż 0,002 sekundy na próbkę testową. Z wyników wnioskujemy, że wszystkie sześć klasyfikatorów osiąga rozsądne dobre wyniki, gdy używane są cechy kierunkowe. Co więcej, 3-NN ma najniższy poziom błędów 6 Rysunek 5: Podsumowanie wyników dla sześciu wybranych klasyfikatorów. 1,19% kosztem dłuższego czasu procesora (3130,6 sekundy). Wręcz przeciwnie, klasyfikatory liniowe i kwadratowe wymagają najkrótszego czasu na szkolenie i test, jednak skutkują wyższym wskaźnikiem błędu (ponad 3%). W miarę rozsądny kompromis można osiągnąć dzięki SVM i GMM; ten ostatni ma niski poziom błędów (1,37%), który jest tylko drugi za 3-NN, podczas gdy zachowuje znacznie niższy koszt w porównaniu do 3-NN.) el używa konwolucyjnych sieci neuronowych

# Konwolucyjne sieci neuronowe

Głębokie sieci konwolucyjne (CNN) potrafią stopniowo filtrować różne części danych uczących i wyostrzać ważne cechy w procesie dyskryminacji wykorzystanym do rozpoznawania lub klasyfikacji wzorców. W każdej konwolucji możemy wyróżnić:

• Ilość parametrów w każdej warstwie: Ilość kanałów \* Ilość filtrów \* szerokość filtra \* wysokość filtra • Ilość jednostek ukrytych w każdej warstwie: Ilość filtrów \* szerokość wzorca \* wysokość wzorca.

Konwolucje pozwalają na ekstrakcję

prostych cech w początkowych warstwach sieci, np. rozpoznają krawędzie o różnej orientacji lub różnokolorowe plamy, a następnie plastry, koła w kolejnych warstwach. Każda warstwa konwolucyjna zawiera cały zbiór filtrów (np. 8 filtrów), a każdy z nich generuje osobną mapę aktywacji 2D. Układamy te mapy aktywacyjne na stercie wzdłuż wymiaru głębokości sieci i produkujemy obraz wyjściowy.

Linki: <https://ichi.pro/pl/funkcje-aktywacji-relu-i-softmax-148521511785096#:~:text=Lepiej%20jest%20to%20zrozumie%C4%87%20na,a)%20s%C4%85%20wi%C4%99ksze%20ni%C5%BC%200>.

https://brain.fuw.edu.pl/edu/index.php/Uczenie\_maszynowe\_i\_sztuczne\_sieci\_neuronowe/Wyk%C5%82ad\_4

<https://ieeexplore.ieee.org/abstract/document/6296535>

<https://github.com/tgjeon/kaggle-MNIST>