SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Diplomski studij računarstva

Laboratorijska vježba 4

Neuronske mreže

Aproksimiranje kontinuirane funkcije neuronskom mrežom

Denis Lazor, DRB

Osijek, 2020.

SADRŽAJ

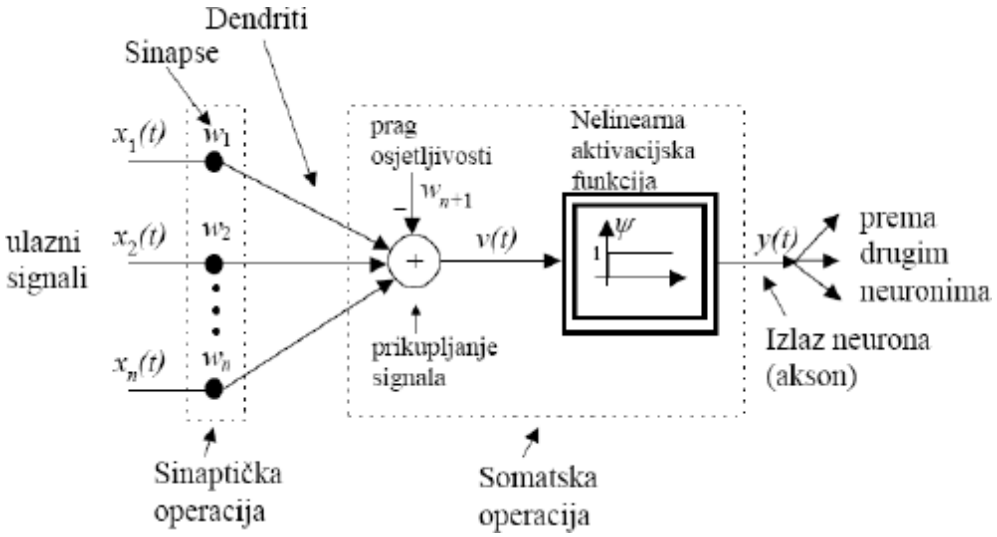
1. UVOD 1
2. NEURONSKE MREŽE 2
3. PROGRAMSKI KOD 4
4. REZULTATI 5
4.1. Linearne i blizu linearne aproksimacije 5
4.2. Nelinearne aproksimacije 8
4.2.1. Ovisnost o broju uzoraka za učenje 8
4.2.2. Ovisnost o broju slojeva 12
4.2.3. Ovisnost o broju neurona 16
4.3. Najbolje rješenje 20
5. ZAKLJUČAK 22

1. UVOD

Zadatak 4. vježbe je bio inicijalizirati regresijsku neuronsku mrežu sa određenim kombinacijama parametara te pomoću nje aproksimirati zadanu kontinuiranu funkciju. Također trebalo je proučiti utjecaj određenih parametara na točnost aproksimacije funkcije.

2. NEURONSKE MREŽE

Neuronske mreže su sustavi za procesuiranje informacija koji su inspirirani biološkim živčanim sustavom kao što je mozak. Sastoji se od velikog broja međusobno povezanih procesnih elemenata tzv. neurona. Svaki neuron je sumirajući element povezan sa aktivacijskom funkcijom. Prvi model neurona koji je bio osmišljen još 1943. od strane McCollocha i Pittsa se zvao „perceptron“ i njegova jedina razlika od modernijih modela neurona je bila ta što je kao prijenosnu funkciju koristio diskontinuiranu step funkciju. Kasnije su se počee koristiti kontinuirane funkcije. Najveća revolucija se dogodila uvođenjem višeslojnih mreža i njihovih algoritama za učenje.



Slika 2.1. Shematski prikaz perceptrona

Iz slike 2.1 je vidljivo kako svaki neuron prikuplja signale od prethodnog sloja(pomnožene sa težinama), te uz dodatak praga osjetljivosti dolazi do prijenosne funkcije odnosno nelinearne aktivacijske funkcije. Izlaz iz te funkcije potom odlazi do svakog neurona u idućem sloju gdje se proces ponavlja. Neuroni se najčešće dijele na statičke i dinamičke, gdje statički neuroni ovise isključivo o trenutnim vrijednostima signala i težina, dok kod dinamičkih postoje određene povratne veze i promjenjive aktivacijske funkcije.

Da bi se neuronska mreža definirala, pored osnovnih parametara koji opisuju oblik i tip mreže, odnosno arhitekturu, potrebno je odrediti i algoritam učenja.
Proces učenja je u biti proces optimizacije pomoću algoritma gdje se pronalaze težine između neurona koje najbolje opisuju rješenje odnosno aproksimaciju problema. Proces učenja najčešće uključuje slijedeće korake:

- Dovođenje na ulaz neuronske mreže niz slučajeva (uzoraka) koje želimo naučiti raspoznavati.
- Odrediti pogrešku između dobivenog izlaza i željenog izlaza.
- Promijeniti težine da bi se izlaz bolje aproksimirao

Imamo 3 osnovna tipa učenja neuronskih mreža:

- Nadzirano učenje – učenje na temelju poznatih uzoraka i rezultata
- Učenje pojačavanjem – uključuje povratnu vezu iz okoline
- Nenadzirano učenje – učenje iz pravilnosti ulaznih podataka.

Najčešće se koristi nadzirano učenje, a najčešće korišteni algoritam učenja je sa povratnom propagacijom pogreške (eng. backpropagation).

Neuronske mreže zbog svoje sposobnosti učenja i aproksimacije se najčešće koristi za slijedeće primjene:

- Raspoznavanje znakova teksta (i analiza slike),
- Prepoznavanje govora,
- Adaptivno uklanjanje šuma,
- Predviđanje cijena dionica(financije),
- Medicinska dijagnostika.

Parametri neuronske mreže:

- hidden_layer_sizes – predstavlja broj neurona u skrivenim slojevima neuronske mreže kao ntorka, u gore navedenom primjeru (100,) je navedeno da mreža ima jedan skriveni sloj s 100 neurona.
- activation – predstavlja aktivacijsku funkciju neurona, te može biti odabrano:

  -> ‘identity’, f(x) = x.

  ->‘logistic’, f(x) = 1 / (1 + exp(-x)).

  ->‘tanh’, f(x) = tanh(x).

  -> ‘relu’, f(x) = max(0, x).
- solver – predstavlja metodu koja će se koristiti za optimizaciju težina neuronske mreže, te može biti odabrano:

  -> ‘lbfgs’, metoda iz porodice kvazi-Newton metoda.

  -> ‘sgd’, engl. Stochastic Gradient Descent.

  -> ‘adam’ novija metoda 'sgd'-a
- alpha – L2 regularizacijski parametar
- max_iter – maksimalni broj iteracija učenja neuronske mreže

3. PROGRAMSKI KOD

U postojeći kod dodana je funkcija za prikaz rezultata na ekranu. Funkcija je prikazana na slici 3.1

```
def print_results(Neurons,Layers,Alpha,Activation,Solver,MaxIterations,SampleSize):
    print("\nNeurons per layer: " + str(Neurons) + ", Layers: " + str(Layers) + ", Alpha: " + str(Alpha) +
          ", MaxIterations: " + str(MaxIterations) + ", Activation: " + str(Activation) + ", Solver: " + str(Solver) +
          ", SampleSize: " + str(SampleSize))
```

Slika 3.1. Funkcija za prikaz rezultata

Kod prikazan na slici 3.2 predstavlja funkcije za dohvaćanje podataka iz interaktivnog prozora. Prvo se uzima naziv datoteke sa uzorcima kao string i izdvaja se dio sa veličinom i 'cast-a' se u varijablu tipa int. Zatim se računa broj uzoraka prema broju zareza u textbox-u za upis broja neurona za svaki sloj. Nazivi aktivacijske funkcije i funkcije za učenje mreže se dohvaćaju preko *currentText()* funkcije(ranije *currentIndex()*).

```
SAMPLE_SIZE = int(self.comboSample.currentText()[5:7])  # Getting the size of sample used for learning -- Added by Denis Lazor
LAYERS = self.tbxNeuroPerLayer.text().count(',') + 1  # Counting the number of layers --  Added by Denis Lazor
    ACTIVATION_F = self.comboActivation.currentText()  # --Altered by Denis Lazor // ...currentIndex() --> ...currentText()
    SOLVER = self.comboSolver.currentText()            # -||-
```

Slika 3.2. Prikaz funkcija za dohvaćanje podataka iz interaktivnog prozora

Kod prikazan na slici 3.3. predstavlja inicijalizaciju regresijske neuronske mreže.

```
#Create neural network --Added by Denis Lazor
mlp = MLPRegressor(hidden_layer_sizes=N_PER_LAYER, activation=ACTIVATION_F, solver=SOLVER, alpha=ALPHA, max_iter=MAX_ITER)
```

Slika 3.3. Inicijalizacije neuronske mreže

Na slici 3.4. je prikazan poziv funkcije *print_results().*

```
print_results(N_PER_LAYER,LAYERS,ALPHA,ACTIVATION_F,SOLVER,MAX_ITER,SAMPLE_SIZE)
```

Slika 3.4. Poziv print_results() funkcije

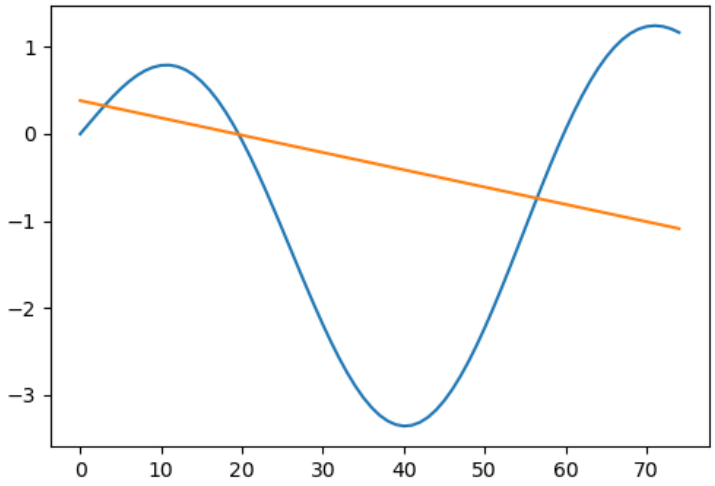
4. REZULTATI

Eksperimenti su provedeni na način da su se testirale različit kombinacije aktivacijskih funkcija i algoritama za učenje. Promjenom ostalih parametara došlo se do zaključka da određene kombinacije daju samo linearnu aproksimaciju zadane funkcije(*Identity* aktivacijska funkcija sama po sebi + još neke kombinacije). Zadan funkcija je nelinearna pa za takve kombinacije će biti prikazan samo jedan graf dok su za ostale kombinacije aktivacijskih funkcija i algoritama za učenje provedeni dodatni eksperimenti radi dobivanja informacije o njihovom utjecaju na točnost aproksimacije zadane kontinuirane funkcije. To se uradilo na način da se mijenjao jedan parametar dok su ostali bili fiksni, to jest fiksan je bio broj uzoraka, a broj neurona i slojeva se mijenjao pošto to polje ovisi jedno o drugom. Radi dodatnog uvida u utjecaj određenih parametara, za najbolju kombinaciju aktivacijske funkcije i algoritma učenja provedene su dodatni testovi. U nekim slučajevima je bilo anomalija da je zadana funkcija krivo prikazana kao i njena aproksimacija pa se za kombinacije uzimao prikaz koji se najčešće dobivao.

4.1. Linearne i blizu linearne aproksimacije

Tablica 1 Adam – Identity kombinacija

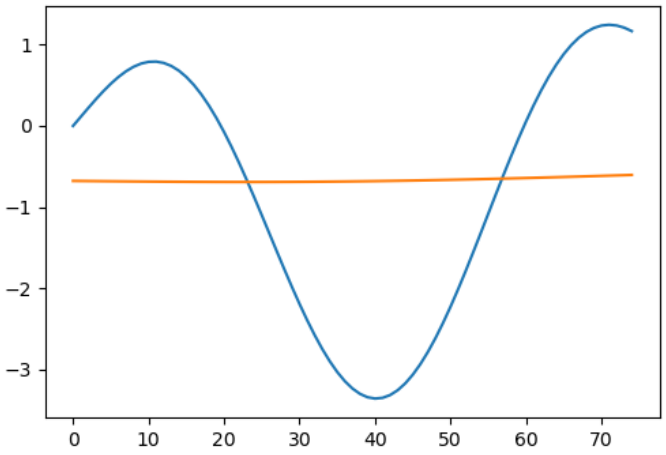
Učenje - aktivacija	Adam - Identity
Broj uzoraka	10
Broj slojeva	1
Broj neurona	5
MSE	2.5388



Slika 4.1. Adam – Identity kombinacija

Tablica 2 Adam –Logistic kombinacija

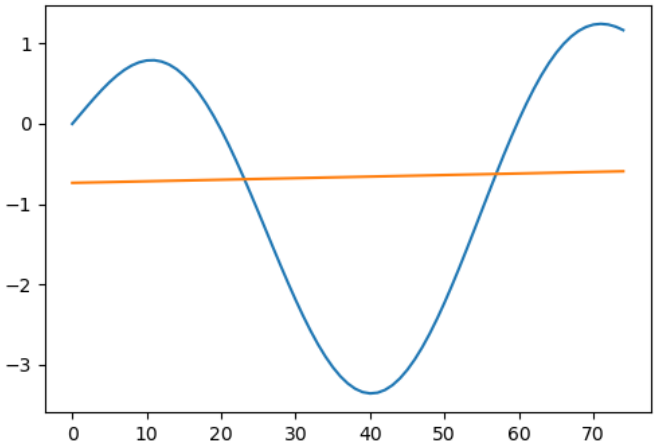
Učenje - aktivacija	Adam -Logistic
Broj uzoraka	10
Broj slojeva	1
Broj neurona	5
MSE	2.4579



Slika 4.2. Adam – Logistic kombinacija

Tablica 3 Lbfgs – Identity kombinacija

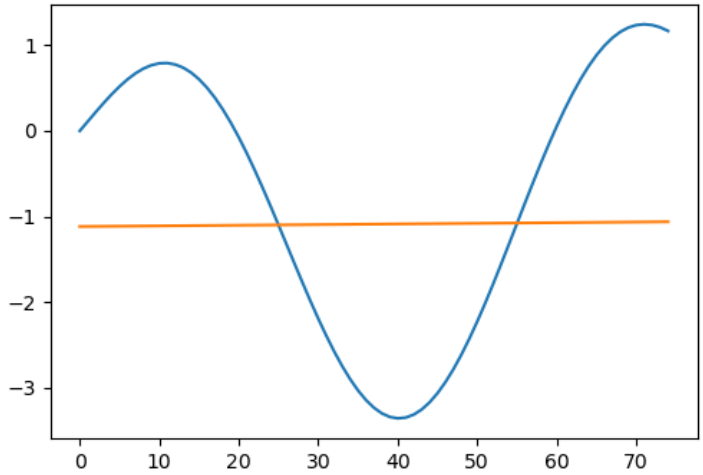
Učenje - aktivacija	Lbfgs - Identity
Broj uzoraka	10
Broj slojeva	1
Broj neurona	5
MSE	2.4980



Slika 4.3.. Lbfgs – Identity kombinacija

Tablica 4 Sgd – Identity kombinacija

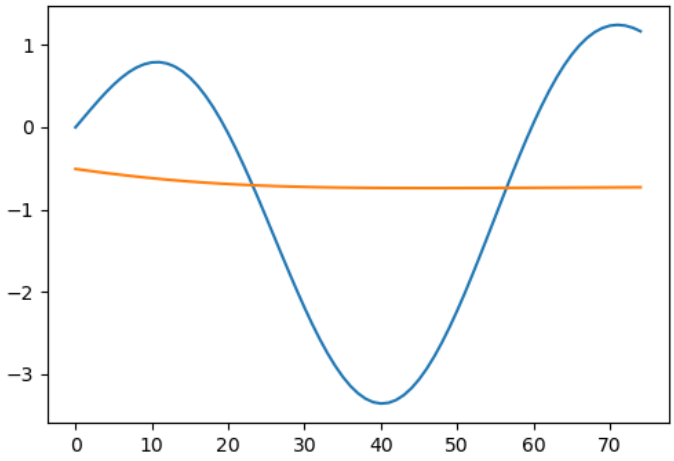
Učenje - aktivacija	Sgd - Identity
Broj uzoraka	10
Broj slojeva	1
Broj neurona	5
MSE	2.5856



Slika 4.4. Sgd – Identity kombinacija

Tablica 5 Sgd - Logistic kombinacija

Učenje - aktivacija	Sgd - Logistic
Broj uzoraka	10
Broj slojeva	1
Broj neurona	5
MSE	2.4031



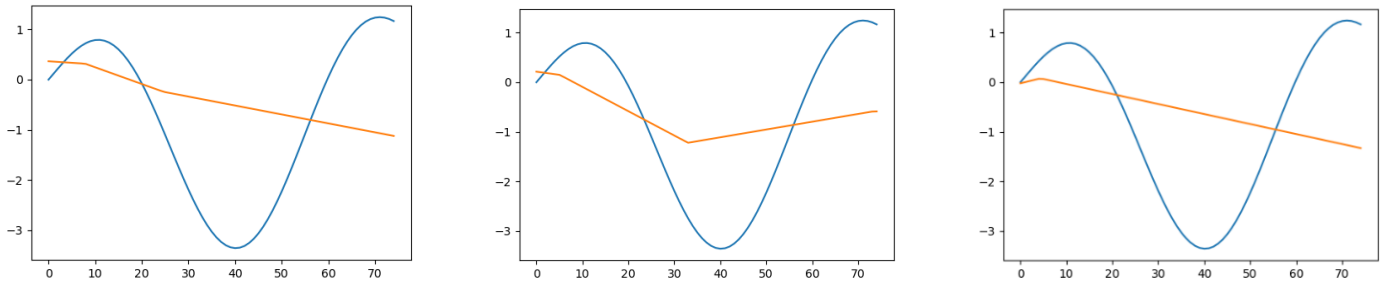
Slika 4.5. Sgd - Logistic kombinacija

4.2. Nelinearne aproksimacije

4.2.1. Ovisnost o broju uzoraka za učenje

Tablica 6 Adam - Relu kombinacija

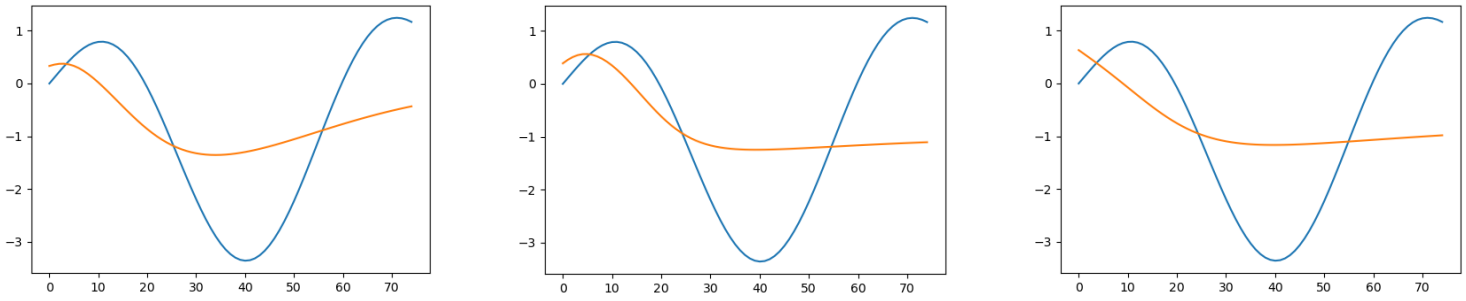
Učenje - aktivacija	Adam - Relu		
Broj slojeva	1		
Broj neurona	5		
Broj uzoraka	10	30	60
MSE	2.6483	1.7071	2.6656



Slika 4.2.1.1. Adam - Relu kombinacija

Tablica 7 Adam - Tanh kombinacija

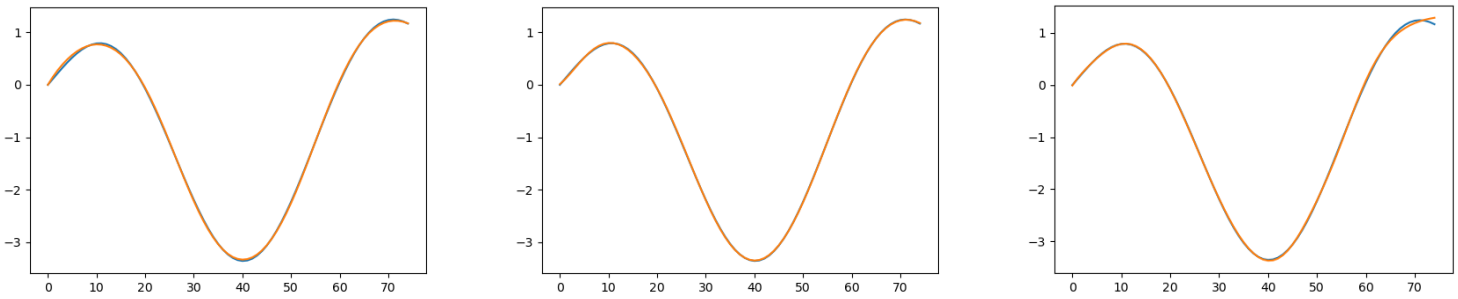
Učenje - aktivacija	Adam - Tanh		
Broj slojeva	1		
Broj neurona	5		
Broj uzoraka	10	30	60
MSE	1.4763	1.8274	1.9143



Slika 4.2.1.2. Adam - Tanh kombinacija

Tablica 8 Lbfgs - Logistic kombinacija

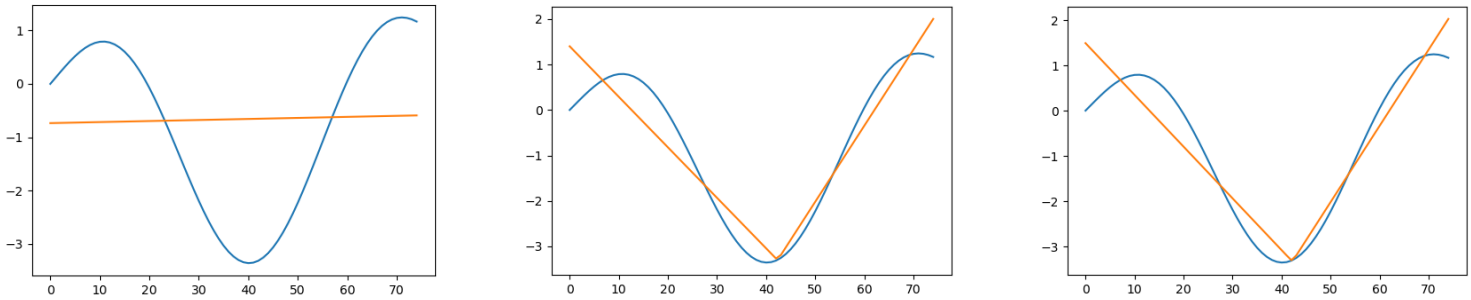
Učenje - aktivacija	Lbfgs - Logistic		
Broj slojeva	1		
Broj neurona	5		
Broj uzoraka	10	30	60
MSE	0.0005	0.0001	0.0006



Slika 4.2.1.3. Lbfgs – Logistic kombinacija

Tablica 9 Lbfgs - Relu kombinacija

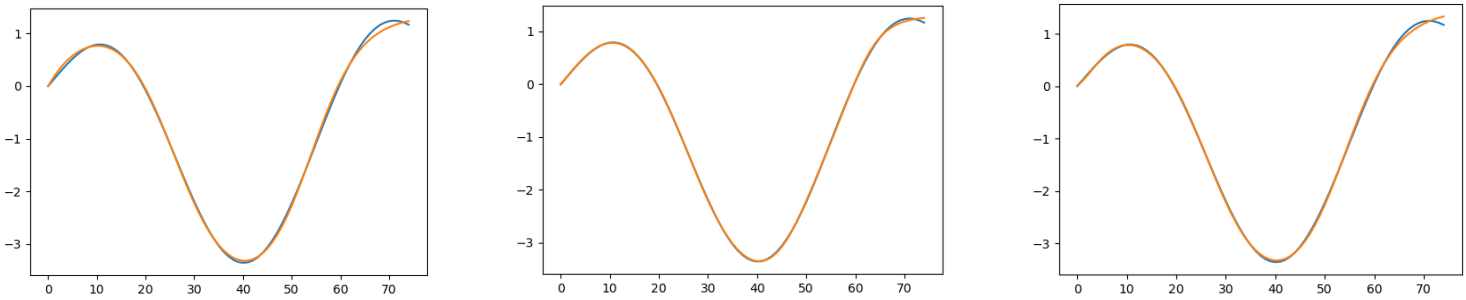
Učenje - aktivacija	Lbfgs - Relu		
Broj slojeva	1		
Broj neurona	5		
Broj uzoraka	10	30	60
MSE	2.4980	0.2532	0.2514



Slika 4.2.1.4. Lbfgs - Relu kombinacija

Tablica 10 Lbfgs - Tanh kombinacija

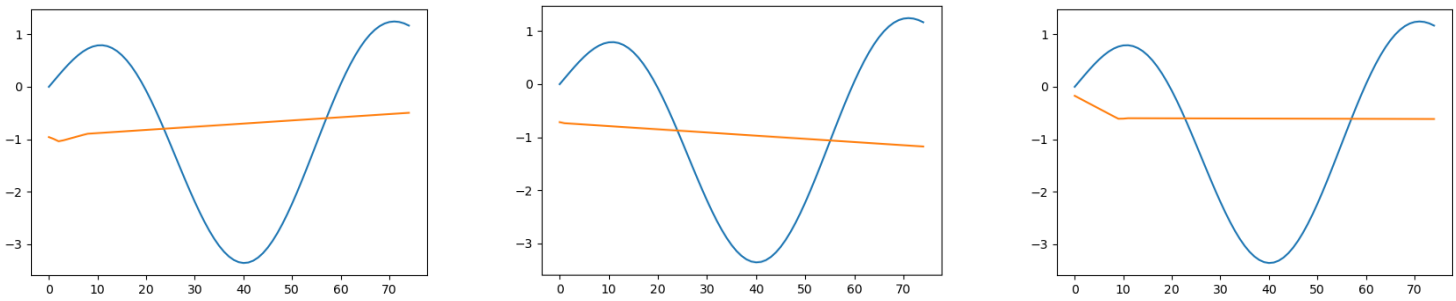
Učenje - aktivacija	Lbfgs - Tanh		
Broj slojeva	1		
Broj neurona	5		
Broj uzoraka	10	30	60
MSE	0.0024	0.0004	0.0013



Slika 4.2.1.5. Lbfgs - Tanh kombinacija

Tablica 11 Sgd - Relu kombinacija

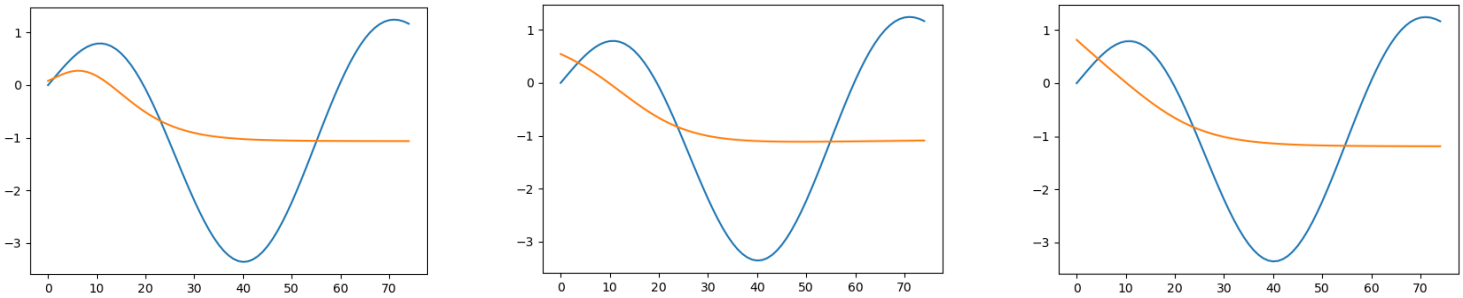
Učenje - aktivacija	Sgd - Relu		
Broj slojeva	1		
Broj neurona	5		
Broj uzoraka	10	30	60
MSE	2.5292	2.5188	2.4684



Slika 4.2.1.6. Sgd - Relu kombinacija

Tablica 12 Sgd - Tanh kombinacija

Učenje - aktivacija	Sgd - Tanh		
Broj slojeva	1		
Broj neurona	5		
Broj uzoraka	10	30	60
MSE	2.0241	2.0186	2.0614

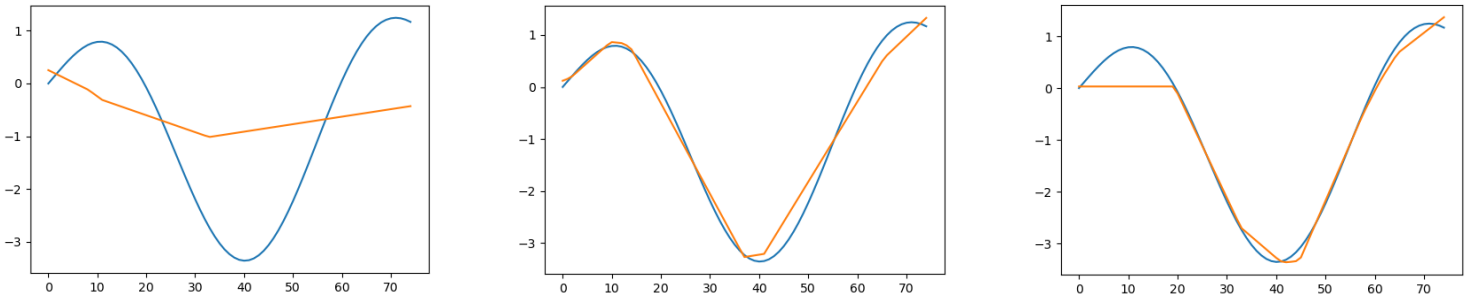


Slika 4.2.1.7. Sgd - Tanh kombinacija

4.2.2. Ovisnost o broju slojeva

Tablica 13 Adam - Relu kombinacija

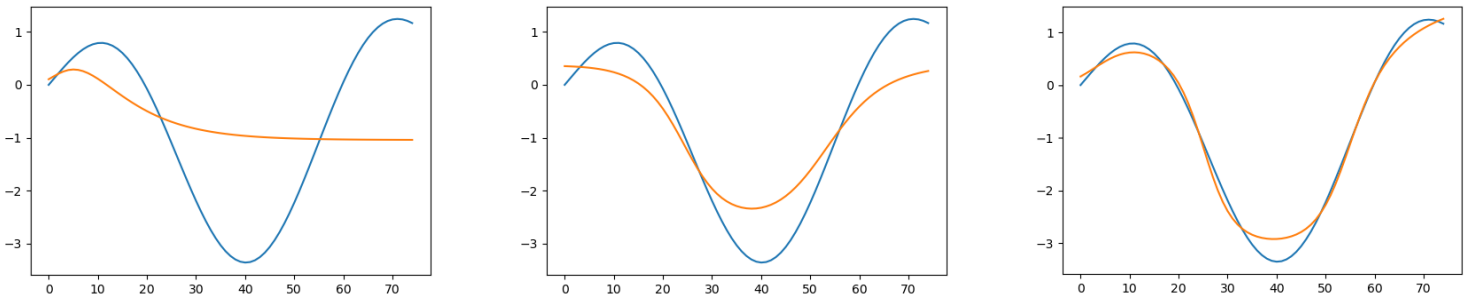
Učenje - aktivacija	Adam - Relu		
Broj uzoraka	10		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
MSE	1.8834	0.05710	0.0853



Slika 4.2.2.1. Adam - Relu kombinacija

Tablica 14 Adam - Tanh kombinacija

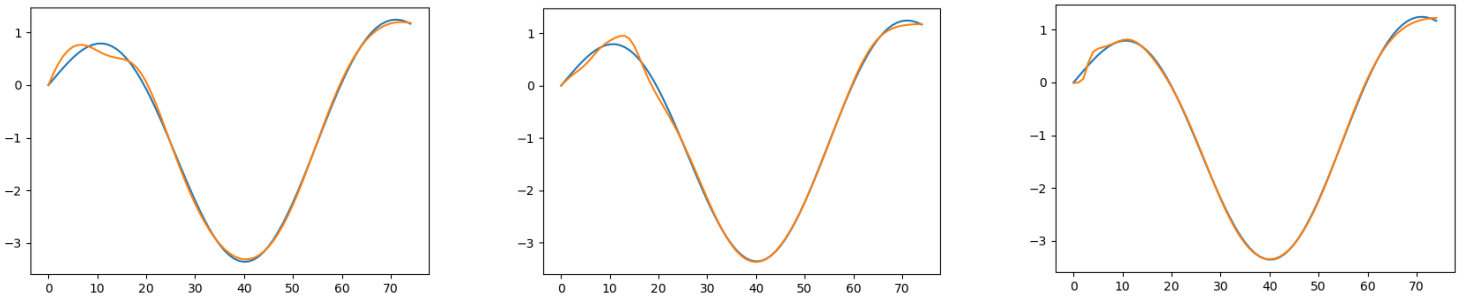
Učenje - aktivacija	Adam - Tanh		
Broj uzoraka	10		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
MSE	2.0864	0.4264	0.0304



Slika 4.2.2.2. Adam - Tanh kombinacija

Tablica 15 Lbfgs - Logistic kombinacija

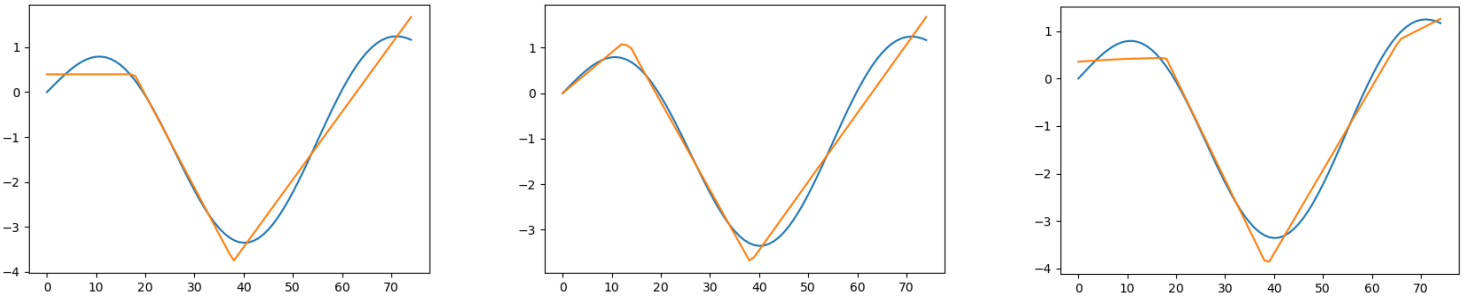
Učenje - aktivacija	Lbfgs - Logistic		
Broj uzoraka	10		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
MSE	0.0073	0.0050	0.0019



Slika 4.2.2.3. Lbfgs – Logistic kombinacija

Tablica 16 Lbfgs - Relu kombinacija

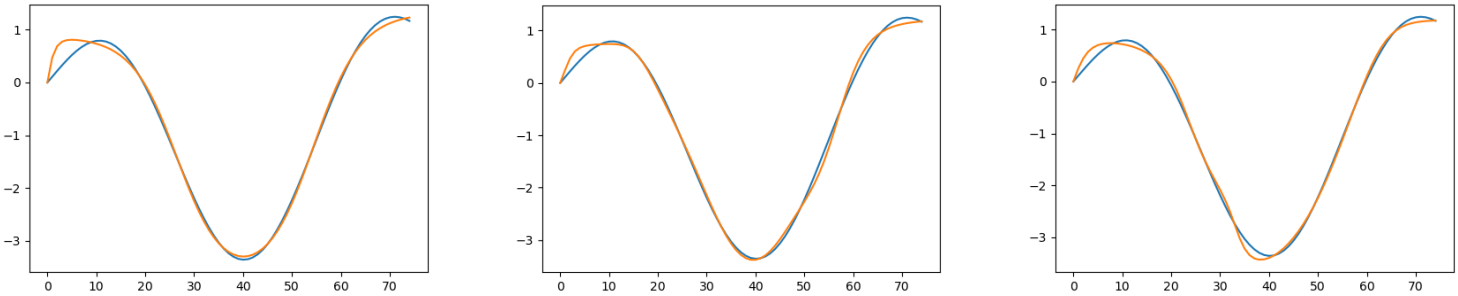
Učenje - aktivacija	Lbfgs - Relu		
Broj uzoraka	10		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
MSE	0.0844	0.0703	0.0475



Slika 4.2.2.4. Lbfgs - Relu kombinacija

Tablica 17 Lbfgs - Tanh kombinacija

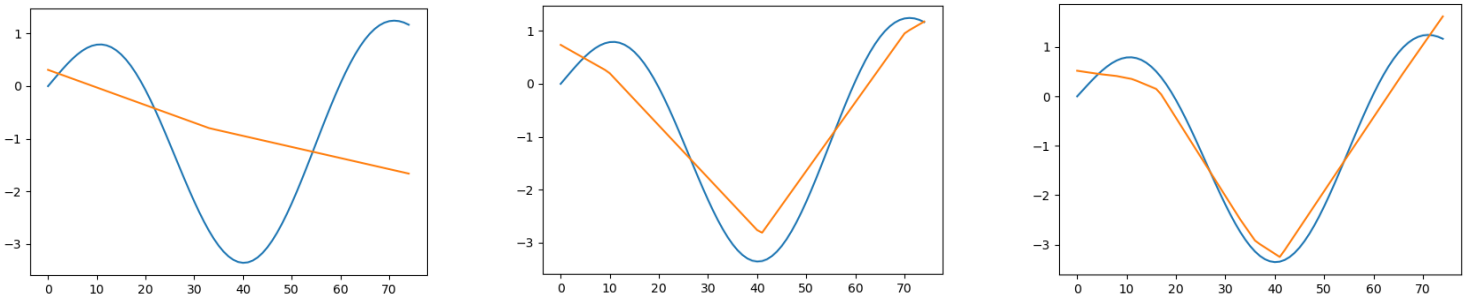
Učenje - aktivacija	Lbfgs - Tanh		
Broj uzoraka	10		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
MSE	0.0084	0.0144	0.0079



Slika 4.2.2.5. Lbfgs - Tanh kombinacija

Tablica 18 Sgd - Relu kombinacija

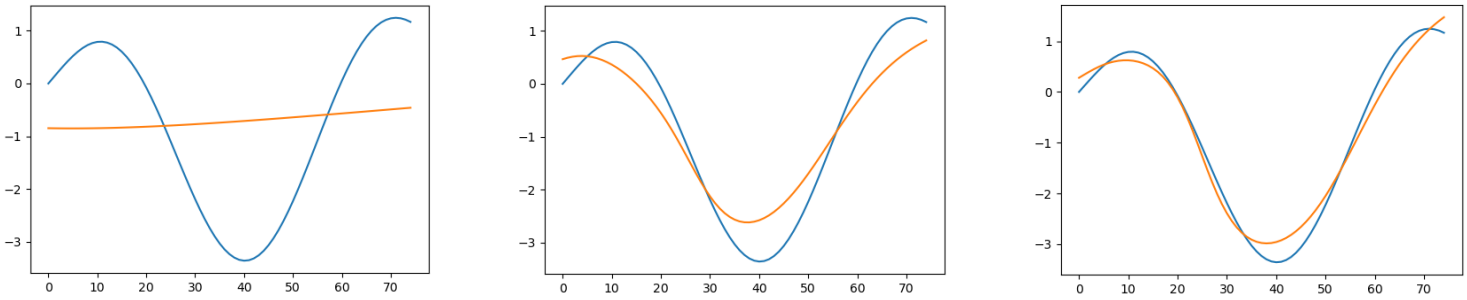
Učenje - aktivacija	Sgd - Relu		
Broj uzoraka	10		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
MSE	2.5752	0.3065	0.1073



Slika 4.2.2.6. Sgd - Relu kombinacij

Tablica 19 Sgd - Tanh kombinacija

Učenje - aktivacija	Sgd - Tanh		
Broj uzoraka	10		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
MSE	2.4585	0.2505	0.0517

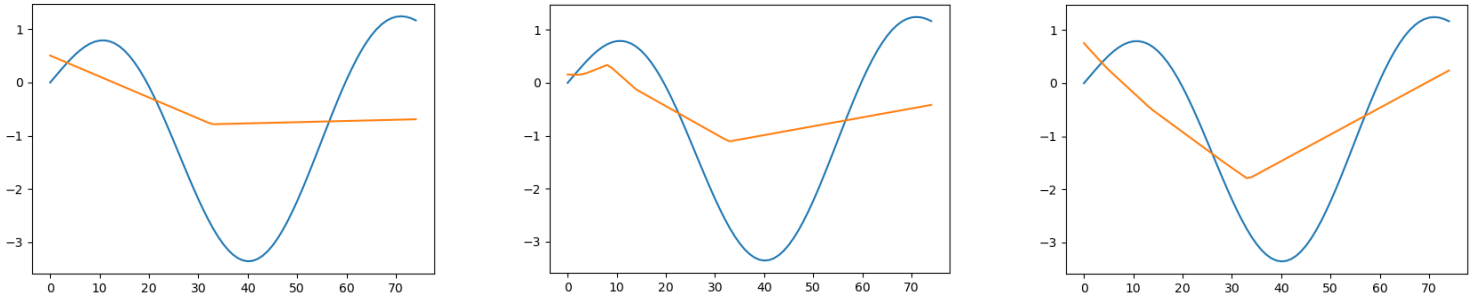


Slika 4.2.2.7. Sgd - Tanh kombinacija

4.2.3. Ovisnost o broju neurona

Tablica 20 Adam - Relu kombinacija

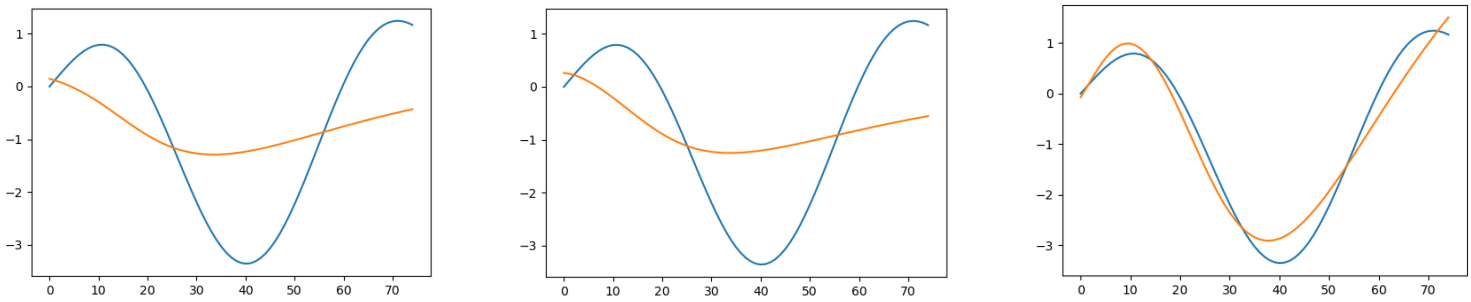
Učenje - aktivacija	Adam - Relu		
Broj uzoraka	10		
Broj slojeva	1		
Broj neurona	5	10	30
MSE	2.0960	1.7053	1.1563



Slika 4.2.3.1. Adam - Relu kombinacija

Tablica 21 Adam - Tanh kombinacija

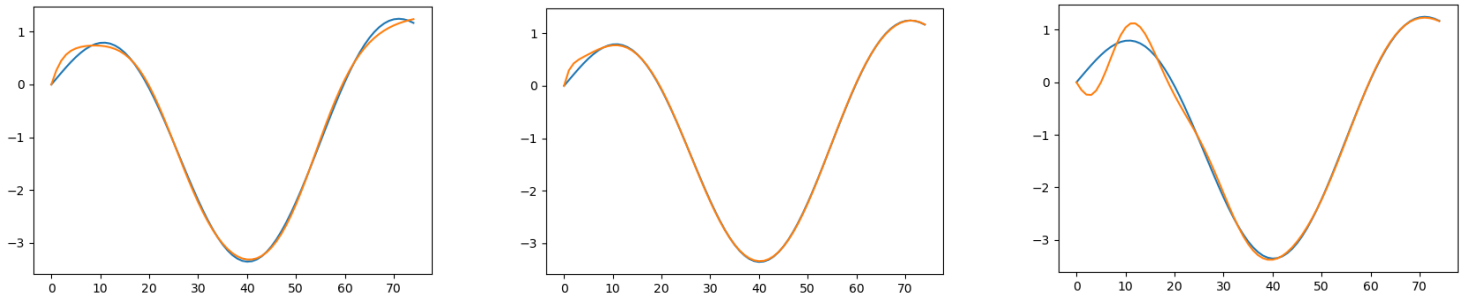
Učenje - aktivacija	Adam - Tanh		
Broj uzoraka	10		
Broj slojeva	1		
Broj neurona	5	10	30
MSE	1.6212	1.6749	0.1104



Slika 4.2.3.2. Adam - Tanh kombinacija

Tablica 22 Lbfgs - Logistic kombinacija

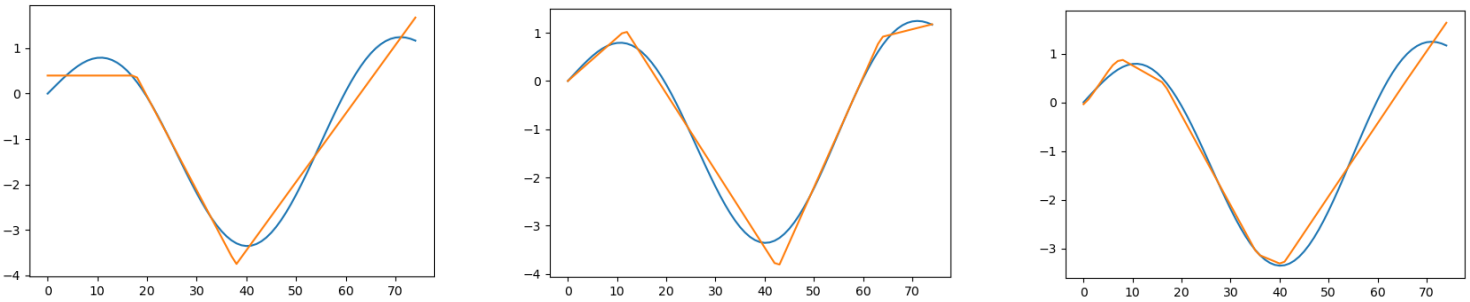
Učenje - aktivacija	Lbfgs - Logistic		
Broj uzoraka	10		
Broj slojeva	1		
Broj neurona	5	10	30
MSE	0.0055	0.0019	0.0275



Slika 4.2.3.3. Lbfgs – Logistic kombinacija

Tablica 23 Lbfgs - Relu kombinacija

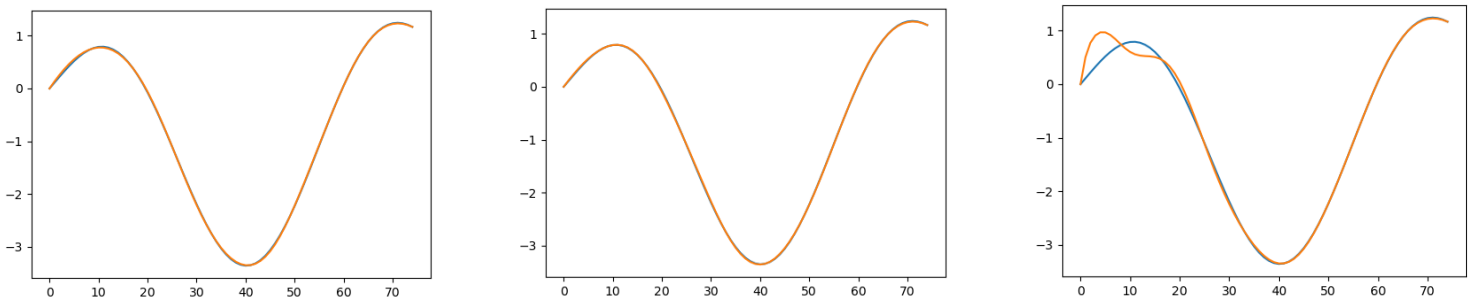
Učenje - aktivacija	Lbfgs - Relu		
Broj uzoraka	10		
Broj slojeva	1		
Broj neurona	5	10	30
MSE	0.0844	0.0358	0.0660



Slika 4.2.3.4. Lbfgs - Relu kombinacija

Tablica 24 Lbfgs - Tanh kombinacija

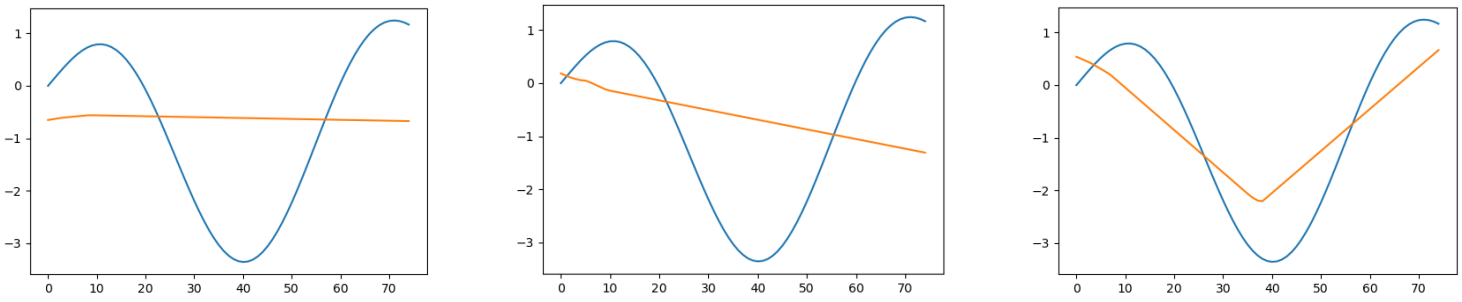
Učenje - aktivacija	Lbfgs - Tanh		
Broj uzoraka	10		
Broj slojeva	1		
Broj neurona	5	10	30
MSE	0.0020	0.0001	0.0233



Slika 4.2.3.5. Lbfgs - Tanh kombinacija

Tablica 25 Sgd - Relu kombinacija

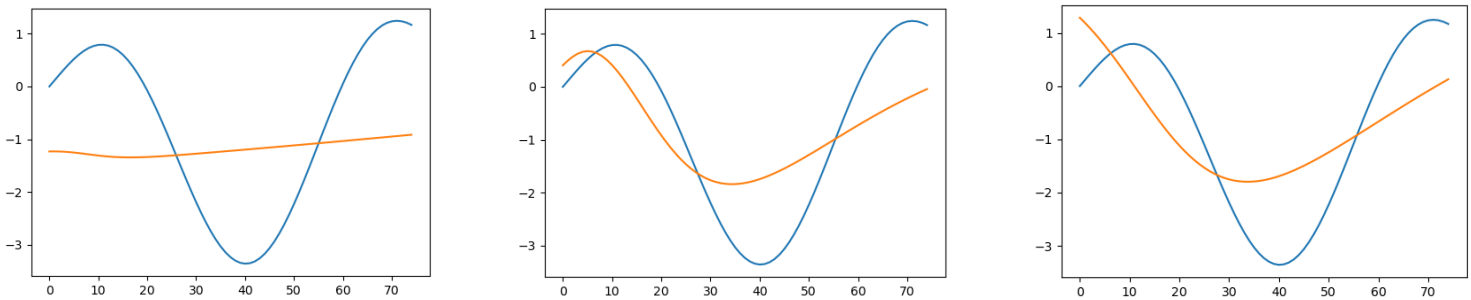
Učenje - aktivacija	Sgd - Relu		
Broj uzoraka	10		
Broj slojeva	1		
Broj neurona	5	10	30
MSE	2.5142	2.6103	0.6996



Slika 4.2.3.6. Sgd - Relu kombinacija

Tablica 26 Sgd - Tanh kombinacija

Učenje - aktivacija	Sgd - Tanh		
Broj uzoraka	10		
Broj slojeva	1		
Broj neurona	5	10	30
MSE	2.1306	1.4153	1.1823



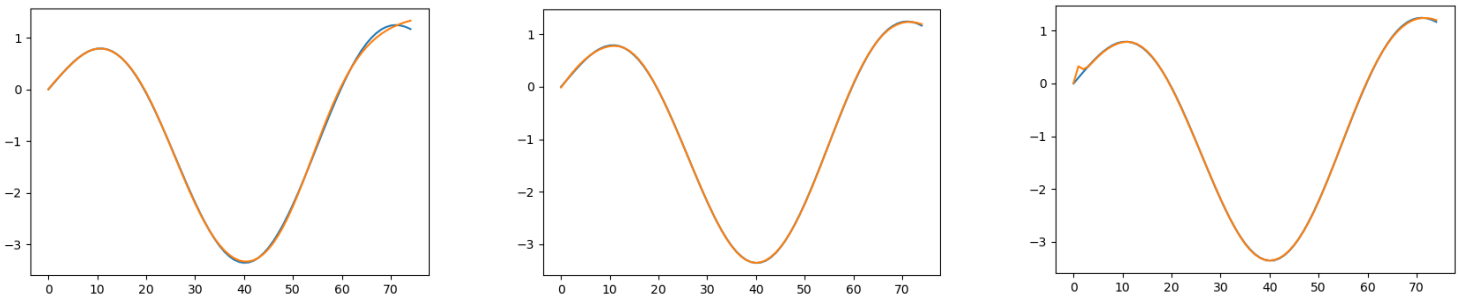
Slika 4.2.3.7. Sgd - Tanh kombinacija

4.3. Najbolje rješenje

Najbolje rezultate od svih kombinacija aktivacijskih funkcija i algoritama učenja dala je kombinacija: '**Lbfgs – Logistic'.** Za tu kombinaciju su provedeni dodatni testovi radi boljeg uvida u utjecaj pojedinih parametara. To jest za sve veličine slojeva i broja neurona isprobane su sve veličine uzoraka za učenje.

Tablica 27 Lbfgs - Logistic kombinacija

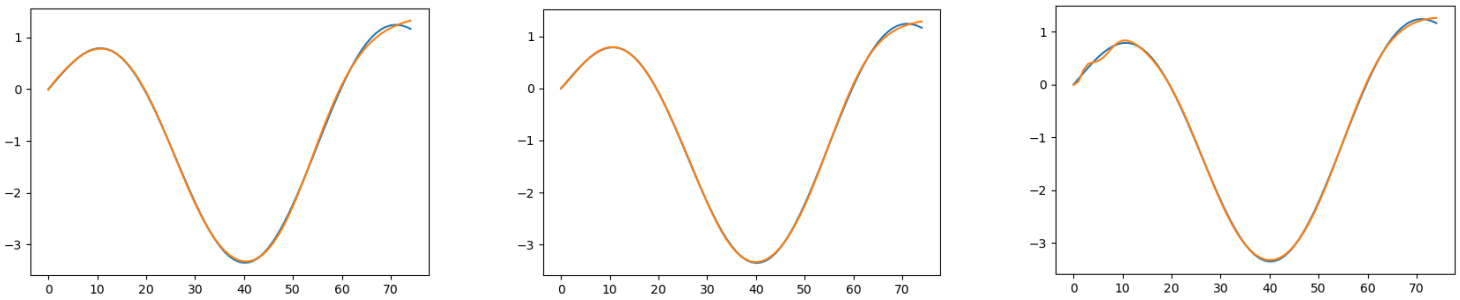
Učenje - aktivacija	Lbfgs - Logistic		
Broj uzoraka	30		
Broj slojeva	1		
Broj neurona	5	10	30
MSE	0.0014	0.0001	0.0008



Slika 4.3.1. Lbfgs – Logistic kombinacija

Tablica 28 Lbfgs - Logistic kombinacija

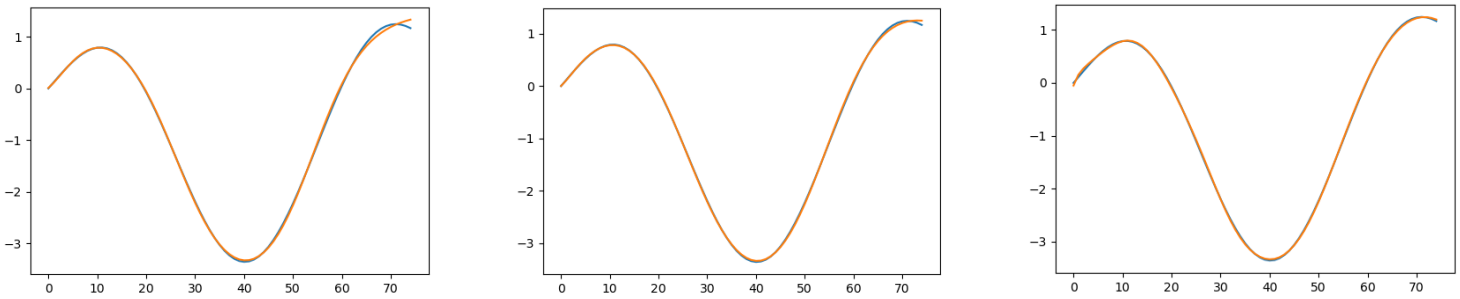
Učenje - aktivacija	Lbfgs - Logistic		
Broj uzoraka	30		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
MSE	0.0014	0.0007	0.0010



Slika 4.3.2. Lbfgs – Logistic kombinacija

Tablica 29 Lbfgs - Logistic kombinacija

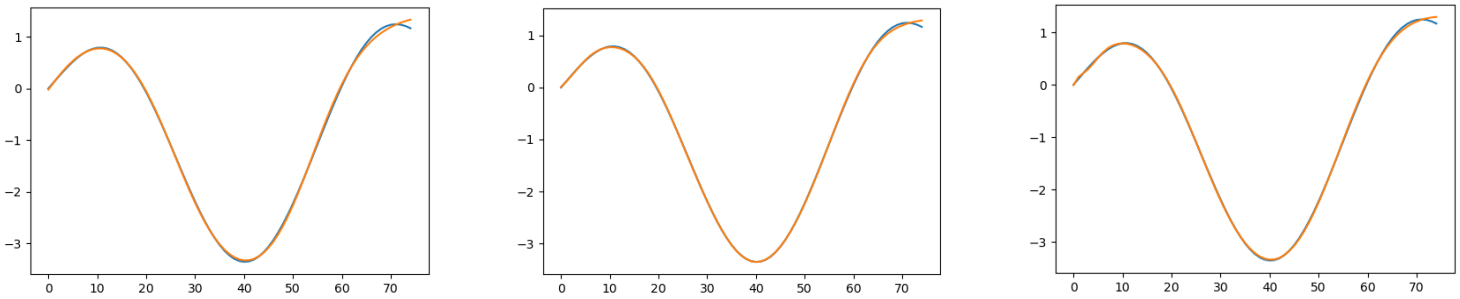
Učenje - aktivacija	Lbfgs - Logistic		
Broj uzoraka	60		
Broj slojeva	1		
Broj neurona	5	10	30
MSE	0.0013	0.0004	0.0004



Slika 4.3.3. Lbfgs – Logistic kombinacija

Tablica 30 Lbfgs - Logistic kombinacija

Učenje - aktivacija	Lbfgs - Logistic		
Broj uzoraka	60		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
MSE	0.0014	0.0005	0.0007



Slika 4.3.4. Lbfgs – Logistic kombinacija

5. ZAKLJUČAK

Iz dobivenih grafičkih prikaza provedenih eksperimenata se vidi da broj uzoraka za učenje nema neki veliki utjecaj na preciznost modela, barem ne za ovako male brojeve uzoraka. Isto se vidi iz grafova dobivenih dodatnim testiranjem najboljeg rješenja iako daje nešto bolje rezultate nego manji broj uzoraka. Broj slojeva ima puno veći utjecaj. Rezultati su redovito bili najbolji za najveći broj slojeva, koji je u ovom slučaju 3. Isto tako, rezultati su redovito bili najbolji za najveći broj neurona, koji je u ovom slučaju 30. Zaključak je da slojevi i neuroni pridonose preciznosti mreže, ali isto tako pridonose i njenoj kompleksnosti. Za slučajeve gdje je je veći broj slojeva i neurona nego u ovoj vježbi, model bi vjerovatno bio puno sporiji, a možda i manje precizan ako je kompleksnost jako velika. Nakon dobivene određene preciznosti nema smisla dalje komplicirati mrežu zbog jako malih ili nikakvih dobitaka u preciznosti. Što se tiče broja uzoraka za učenje, uvijek je poželjno da mreža ima što više podataka iz kojih treba učiti, ali opet prevelik broj uzorak često nije poželjan zbog različitih šumova u podatcima, vremenskog trajanja, preveliko prilagođavanje modela podacima za učenje i nekad je jednostavno moguće napraviti dobar model sa manjim skupom pa nema potreba za jako velikim skupovima podataka, naravno oviseći o kompleksnosti problema koji se rješava. Što se tiče aktivacijskih funkcija i algoritama učenja svi imaju svoje prednosti i nedostatke to jest sve ovisi o problemu koji se rješava. Najčešće se koriste kombinacije različitih aktivacijskih funkcija u modelu ovisno o problemu, a ne samo jedna. Zato '*identity*' aktivacijska funkcija koja stoji sama uvijek daje linearni izlaz i ne može aproksimirati nelinearnu funkciju jer se zapravo na ovaj način dobije rješenje koje radi kao jedan sloj. Što se tiče '*relu*' aktivacijske funkcije koja je prilično popularna, ona također u određenim kombinacijama radi prilično dobro, ali zbog prirode aktivacijske funkcije dobivena krivulja nije glatka. Bez obzira što krivulja nije glatka ta aktivacijska funkcija je popularna zbog brzine, rješavanja problema nestajućeg gradijenta, optimizacije broja neurona koji su okinuti, pogotovo njegova nadograđena inačica *'leaky relu'*. Pošto se ovdje radi o aproksimacije jednostavne i glatke krivulje, '*logistic*' i 'tanh' aktivacijske funkcije pružaju najbolje rezultate što se tiče preciznosti jer bolje aproksimiraju glatkost gradijenta nego ostale. Od algoritama učenja najbolje rezultate je dao algoritam *'Limited-memory BFGS'* , a najgore *'SGD'* i nadograđeni SGD '*Adam'*. LBFGS metoda daje najbolje rezultate pošto je to metoda drugog reda, to jest uzima u obzir drugu derivaciju pa dobro opisuje lokalne gradijente te iste te gradijente koristi za izgradnju *'Hessian-a'* kojeg aproksimira određenim izrazom radi uštede memorije i time preciznije ažurira vrijednosti težina. Ali taj algoritam troši puno resursa pa nije pogodan za rješavanje velikih problema.