# Itseluokitteleva neuroverkko

Noel Guilland D-050

# Johdanto

## Työn tausta

Työ käsittelee neuroverkkoja, joten niihin liittyvä matematiikka on oleellista työn kannalta.

Itseluokittelemis-ominaisuus pyrkii hyödyntämään käsittelemätöntä dataa koneoppimiseen.

## Työn tavoitteet

Rakentaa toimiva neuroverkko alusta asti.

Suunnitella itseluokittelu menetelmiä.

Verrata itseluokittelun suoriutumista rakennettuun neuroverkkoon

# Miksi itseluokittelu?

Itseluokittelun toive on hyödyntää käsittämätöntä dataa valmiiksi käsiteldyn datan kanssa rinnakkain.

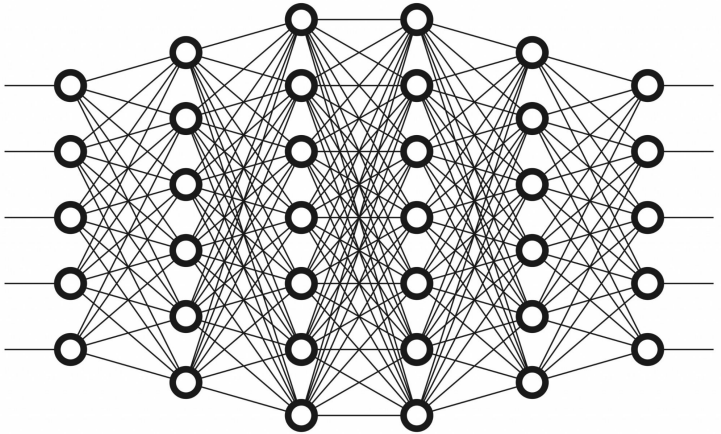
Keino itseluokittelulle on alkeellisesti se, että syötämme neuroverkolle luokittelemattoman esimerkin sen käsittelemästä datan muodossa, jolloin se esittää oman arvauksen parametreihin perustuen.

Tämä edellyttäisi sen, että ihmisten ei tarvitsisi käydä kaikkea tiettyyn tehtävään liittyvää dataa läpi manuaalisesti, vain he voisivat käydä läpi vain osan, mutta silti hyödyntää muita tehtävään oleellisia esimerkkejä.

# Taustatietoa neuroverkoista - Neuroverkko käsitteenä

Käsitteenä neuroverkkoa voidaan visualisoida kuten viereisessä kuvassa näytetään.

Neuroverkoilla on neuroneita jotka jaetaan neuronikerroksiin. Neuronikerrokset ovat yhteydessä niitä naapuroiviin kerroksiin kaikilla neuronien välisillä permutaatioilla.



Esimerkillinen kuvaus neuroverkosta

# Taustatietoa neuroverkoista - Neuroverkko käytännössä

Käytännössä neuroverkko toimii matriisilaskennan avulla. Neuroverkko ottaa syötteenään tietoa, joka on vektorimuodossa.

Neuroverkon syöttöarvoja kuljetetaan, ja muokataan ‘painojen’ ja ‘vakiotermien’ avulla. Neuronikerrosten yhteyksissä käytetään aktivaatio funktiota, joka ottaa syötteenä- ja tulostaa vektorin.

$$\begin{bmatrix} h_1 \\ h_2 \\ \dots \\ h_j \end{bmatrix} = \sigma \left( \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1k} \\ w_{21} & w_{22} & \dots & w_{2k} \\ \dots & \dots & \dots & w_{(j-1)k} \\ w_{j1} & w_{j2} & w_{j(k-1)} & w_{jk} \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_2 \\ \dots \\ z_k \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_j \end{bmatrix} \right)$$

Esimerkki neuronikerrosten välisistä matriisilaskuista, sigma merkitsee aktivaatiofunktiota.

# Taustatietoa neuroverkoista - Neuroverkon säätäminen

Jotta neuroverkon suoriutumista voitaisiin säätää tehokkaasti, tarvitsemme tavan tietämään mitä säätää ja kuinka paljon.

Backward propagation -algoritmi hyödyntää neuroverkon häviö-yhtälöä, ja käyttää ketjusääntöä päästäkseen painojen ja vakio-termien virheen vaikutukseen.

$$L = -\sum_{i=1}^{C} y_i \log(\sigma\left(a_k W_k + b_k\right))$$

$$\frac{\partial L}{\partial W_k} = \frac{\partial L}{\partial z_k} \frac{\partial z_k}{\partial W_k}$$

Esimerkillinen tapa saada kerroksen k painojen vaikutuksen häviöön.

# Taustatietoa neuroverkoista - Muutosten iterointi

Neuroverkon häviötä voidaan käsitellä moniulotteisina yhtälönä, jossa vaikuttavia tekijöitä on yhtä monta kun painoja ja vakioita.

Kaikkien tekijöiden vaikutukset häviöön yhdistetään yhteen gradienttiin, josta päätellään paras suunta painojen ja vakioiden muutokselle.

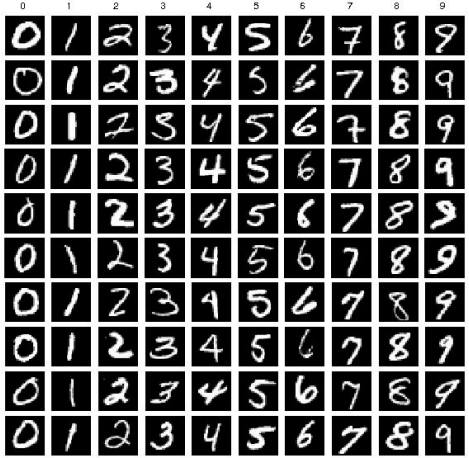
Kun häviötä arvioidaan, sen suuruus riippuu siitä kuinka kaukana neuroverkon ‘arvaus’ oli oikeasta vastauksesta. Arvojen muutosta suhteutetaan häviöön, eli kun häviö on suuri, muutos on myös suuri.

Parametrejä voidaan käsitellä pisteenä – joka laskeutuu laaksoon – eli kaikista alhaisimman häviön tilaan.

# Työn toteutuminen - Työkalut ja työn tuotteet

Neuroverkko rakennettiin Python -ohjelmointikielellä ja lisäksi työssä käytettiin NumPy-kirjastoa. Neuroverkon versionhallinta tapahtui githubin kautta.

Yhteensä työn aikana kehitettiin kaksi eri neuroverkkorunkoa, sekä kaksi itseluokittelukeinoa. Kutsumme neuroverkkorunkoja Prototyyppi 1:ksi, ja -2:ksi.



Neuroverkkon ‘datasetti’ eli se asia jota se mallinsi oli MNIST handwritten digits -datasetti. Data-setti koostui yksilukuisista 28x28 grayscale -kuvista.

# Työn toteutuminen - Prototyyppi 1

Prototyyppi 1 -neuroverkko toimii kahden eri silmukan sisällä: training_loop ja testing.

Mallin toiminta perustu moneen koodattuun funktioon, jotka oli vastuussa neuroverkon toiminnasta. Oppimissilmukka seurasi Stochastic gradient descent -oppimista, eli parametrejä säädettiin jokaisen koulutuskerran jälkeen.

Heikkouksia oli odotettu, sillä malli ei perustunut ennen tehtyyn koodiin, eli kaikki rakenteet oli alkuperäisiä.

Mallin tarkkuus oli heikko, ja taustalla oletetaan olevan ohjelmointivirhe. Prototyyppi 1 -mallia hyödynnettiin Prototyyppi 2 mallin rakenteessa, uudempi prototyyppi rakennettiin alussa tämän virheen korjaukseen.

# Työn toteutuminen - Prototyyppi 2

Prototyyppi 2 -neuroverkon koulutus tapahtuu train_neural_network -funktion sisällä. Funktion silmukkaan kuuluu mm. forward_propagation, backward_propagation ja update_parameters -toiminnot.

Silmukka toimii mini-batch gradient descent oppimisella, eli arvojen muutosta tehdään ottamalla monen muutoksen keskiarvo.

```python
# Train the neural network
def train_neural_network(X, Y, otos, layer_sizes, learning_rate, epochs, batch_size, threshold):
    parameters = initialize_parameters(layer_sizes)
    start_time = datetime.datetime.now()
    luokiteltu = False # Ettei luokittele montaa kertaa, ottaa aikaa ja tekee mallista epätarkan
    accuracies = np.zeros(epochs)
    losses = []
    #with open("ItseluokittelevaNN" + str(otos) + ".csv", mode="w", newline="") as file:
    #    writer = csv.writer(file)
    #    writer.writerow(["Epoch", "Tarkkuus"])

    # Vain 15000 tiedossa
    X_initial = X[:otos]
    Y_initial = Y[:otos]
    X_unlabeled = X[otos:] # 45000, 784
    Y_unlabeled = np.zeros((X_unlabeled.shape[0], 10)) # 45000, 10
    itseluokitteluBatch = len(X_unlabeled)//epochs

    for epoch in range(epochs):
        for i in range(0, X_initial.shape[0], batch_size):
            X_batch = X_initial[i:i + batch_size].T  # Transpose to shape (features, batch_size)
            Y_batch = Y_initial[i:i + batch_size].T  # Transpose to shape (classes, batch_size)
            AL, caches = forward_propagation(X_batch, parameters)
            loss = compute_loss(AL, Y_batch)
            grads = backward_propagation(X_batch, Y_batch, parameters, caches)
            parameters = update_parameters(parameters, grads, learning_rate)
        learning_rate *= 0.99
        accuracies[epoch] = test_neural_network(x_test.T, y_test.T, parameters) # Pidetään listaa kaikista tarkuuksista
        print(f"Epoch {epoch + 1}/{epochs}, Loss: {loss:.4f}, Accuracy: {accuracies[epoch]:.4f}")
```

Prototyyppi 2 -mallin koulutuskierros.

# Työn toteutuminen - Itseluokittelukeinot

## Menetelmä 1 “Kiihtyvyyden seuraaminen”

Itseluokittelukeino toimii niin, että se odottaa tiettyyn tarkkuuteen pääsemistä, jonka jälkeen se alkaa syöttämään luokittelemattomia esimerkkejä neuroverkolle luokiteltavaksi.

## Menetelmä 2 “Jatkuva syöttö”

Keino perustuu siihen, että esimerkkejä syötetään malliin sen koulutuksen osana, mutta luokiteltavien esimerkkien määrä riippuu neuroverkon viimeksi mitatusta tarkkuudesta.

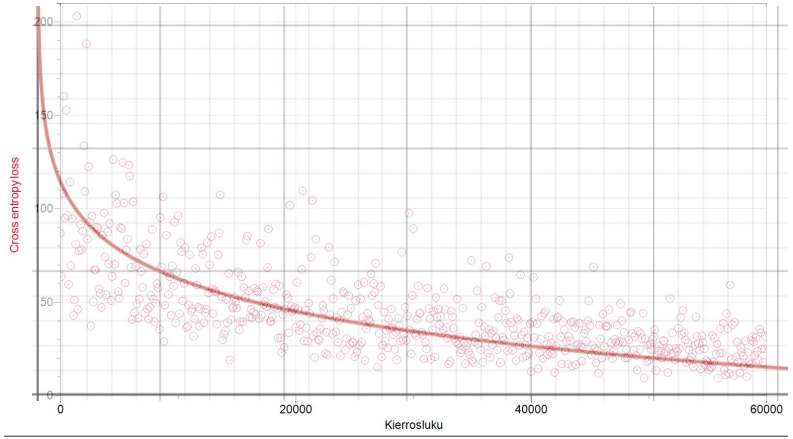
$$\hat{m} \approx E \cdot m,\ \hat{m} \in \mathbb{Z}_+,\ E \in [0,1]$$

# Tulokset ja Johtopäätökset - Prototyypit ja Toimivuus

Neuroverkkomallien rakennuksen tulos on positiivinen, varsinkin Prototyyppi 2. -mallin tapauksessa.

Tähän tulokseen voidaan päätyä monella toimivuuden merkitsijällä, esimerkiksi häviön aikakehitys muistuttaa vahvasti logaritmifunktiota, joka kertoo meille että categorical cross-entropy -häviö on suoriutunut.

Lisäksi Prototyyppi 2. -mallin tarkkuus on – koulutusfunktion parametrit huomiossa – tyypillisiä feedforward- neuroverkolle joka käsittelee MNIST-datasettiä.

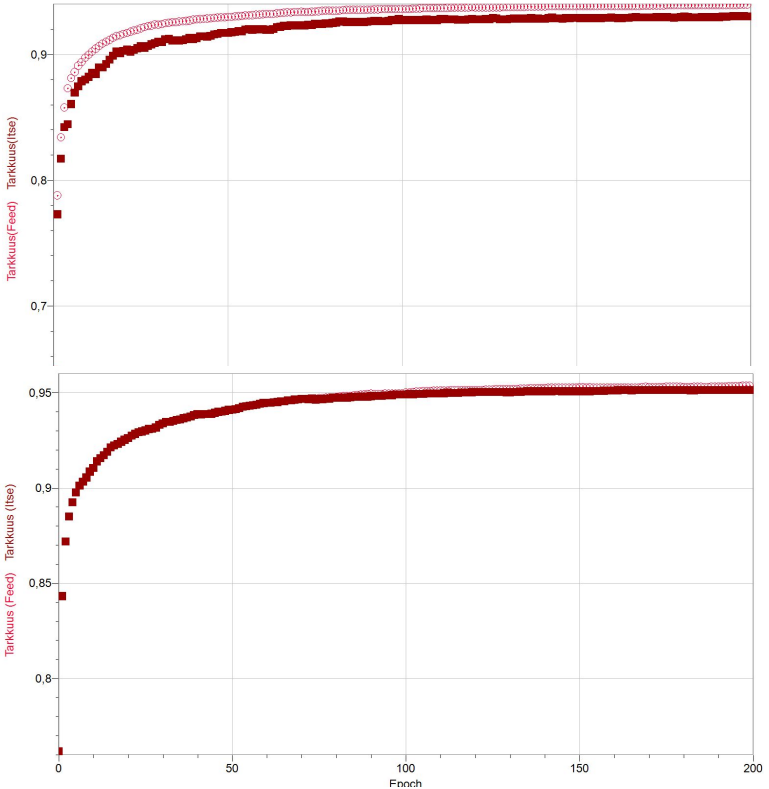


# Tulokset ja Johtopäätökset - Itseluokittelukeinot

Nykyisten tulosten mukaan emme voi todeta itseluokittelukeinojen positiivista vaikutusta neuroverkon tarkkuuteen.

Tulevaisuudessa itseluokittelu-keinojen yhteydessä voisi tapahtua datan esikäsittelyä, joka saattaisi mahdollistaa itseluokittelukeinojen tarkkuuden nostamisen.

Kuvissa näkyy itseluokittelevan- ja perusmallin tarkkuuksien aikakehitys.



# Lähteet:

https://gadictos.com/wp-content/uploads/2019/05/1_1mpE6fsq5LNxH31xeTWi5w-scaled.jpeg (Neuroverkko kuva)

https://iq.opengenus.org/mnist-handwritten-recognition-dataset/ (MNIST kuva)

Muut kuvat itse laadittuja tai tutkimusraportista.