Symbool-herkenner

Leerdoelen

- Praktisch inzicht krijgen in wat nu werkelijk essentieel is voor een eenvoudig neural net en wat "bijkomende" optimalisaties (hoe belangrijk ook) zijn.
- Vertrouwen krijgen in het feit dat neural nets niet alleen uit de hoge hoed van Google worden getoverd, maar dat je die met gebruikmaking van veel gezond verstand en weinig wiskunde gewoon zelf kunt maken.

Inhoud

- Maak in puur Python zonder gebruik van 3rd party libaries een neural net dat kruisjes van rondjes kan onderscheiden. Een hidden layer is niet nodig. De resolutie van de camera is 3 x 3 pixels, het aantal grijswaarden is 2, namelijk zwart of wit. Je mag de beelden met de hand invoeren als 2D array's.
- Test ook met onvolmaakte kruisjes (die bijv. een pootje missen) en onvolmaakte rondjes (bijv. met een gat).
- Begin met een directe, intuitieve implementatie met behulp van de zelf te onwerpen klassen Node en Link, de softmax function aan de uitgangs-laag en het back-propagation algorithm, gebruikmakend van willekeurige variaties van de weight factors. Varieer 1 weight factor per leercyclus over de hele trainingset en behoud alleen de wijziging die gemiddeld over deze cyclus het beste scoort. Gebruik de mean squared error als cost function.
- Stap daarna over op een programma dat werkt met matrices en vectoren, dat in principe hetzelfde doet. Voeg eventueel een hidden layer toe en bekijk de invloed van de dimensie (aantal nodes) hiervan op het fout-percentage. Gebruik eerst zelf gemaakte matrix-vector operaties en daarna NumPy. Gebruik als niet-lineaire functie de sigmoid functie.
- Stap ten slotte als je tijd en interesse hebt, over op het gebruik van steepest descent (gradienten-methode) en kijk wat de invloed is op de convergentie, dat wil zeggen het verloop van de cost function als functie van het aantal doorlopen volledige iteraties

(waarbij voor elke iteratie de training-set volledig wordt doorlopen). Naast sigmoid functie s heb je hiervoor de analytische afgeleide daarvan, s * (1 – s) nodig.

- Zorg dat je bij de eerste twee programma's regel voor regel kunt uitleggen hoe je code werkt. Echter voor verantwoording van het gebruik van de analytische afgeleide in het laatste programma (steepest descent) mag je verwijzen naar literatuur op het web. Formeel begrip van de bijbehorende wiskunde is niet vereist en de laatste opdracht is facultatief.