TI2736-A
Assignment 1: Artificial Neural Network

David Akkerman - 4220390
Jan Pieter Waagmeester - 1222848

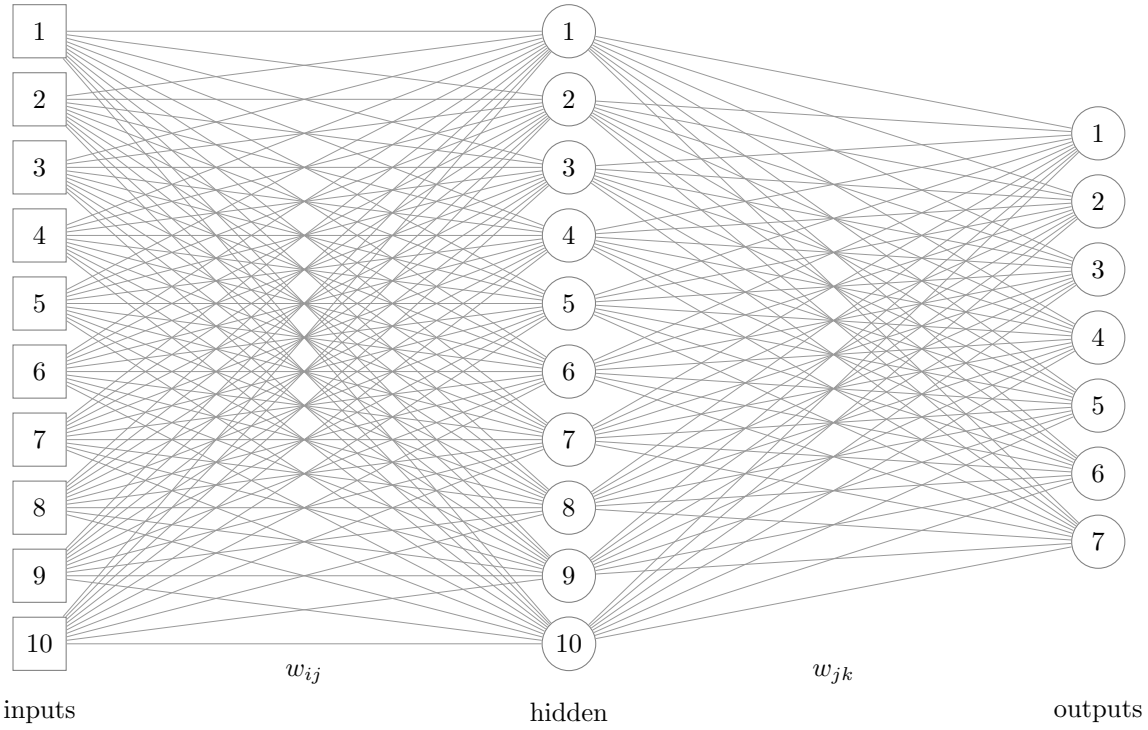
19 september 2014

Onze groepssgenoot is om persoonlijke redenen gestopt met het vak. Deze opdracht hebben we met ons tweeën gemaakt.

1. Omdat er 10 eigenschappen zijn hebben we 10 inputneurons nodig.
2. Voor de 7 verschillende classes hebben we 7 outputneurons nodig.
3. We beginnen met 10 hidden neurons.
4. We zullen de sigmoid-functie gebruiken:

$$Y^{sigmoid} = \left(\frac{1}{1 + e^{-X}} \right)$$

5. Ons netwerk kan voorgesteld worden als in het volgende plaatje, in dit geval met 10 *hidden neurons*. Dit aantal kan uiteraard gevarieerd worden.

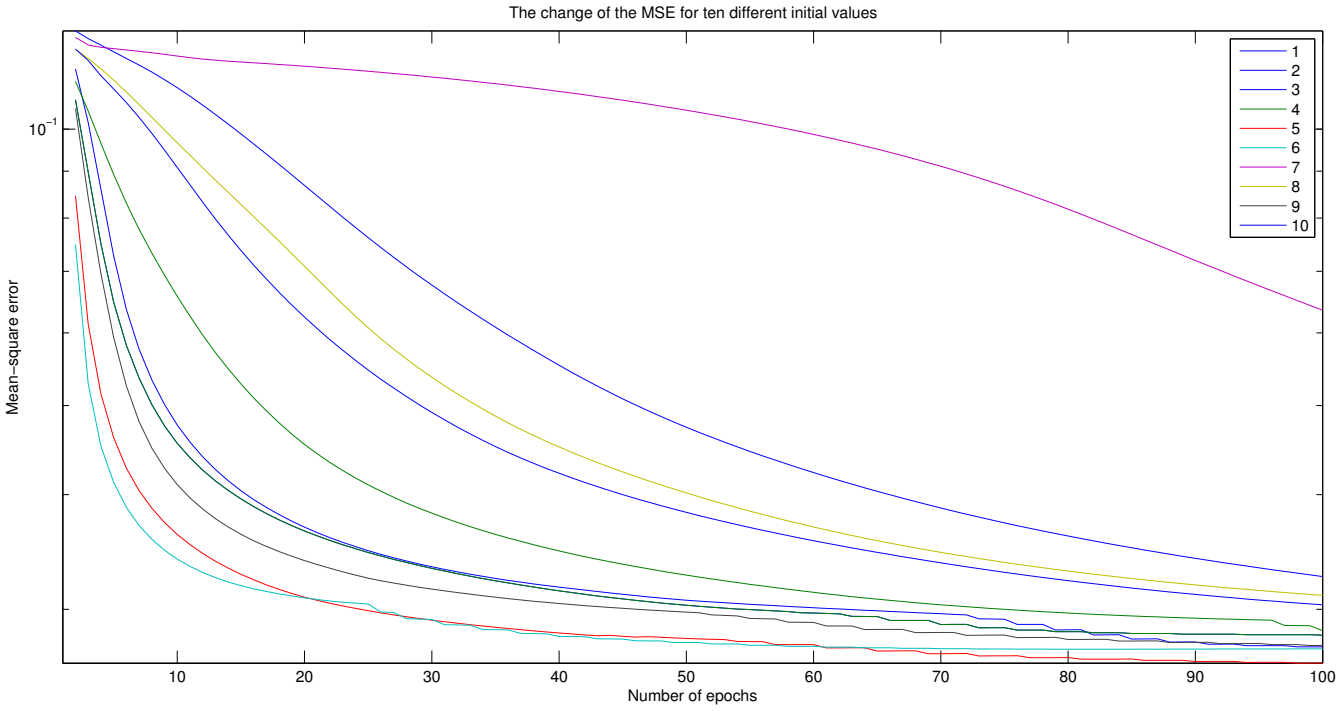


6. We gebruiken ongeveer 50% van de data voor training, 25% voor validatie en 25% als testset. De data lijkt aardig random gesorteerd te zijn, het is dus niet nodig om te zorgen dat er ongeveer even veel van elke class in elke set aanwezig is.

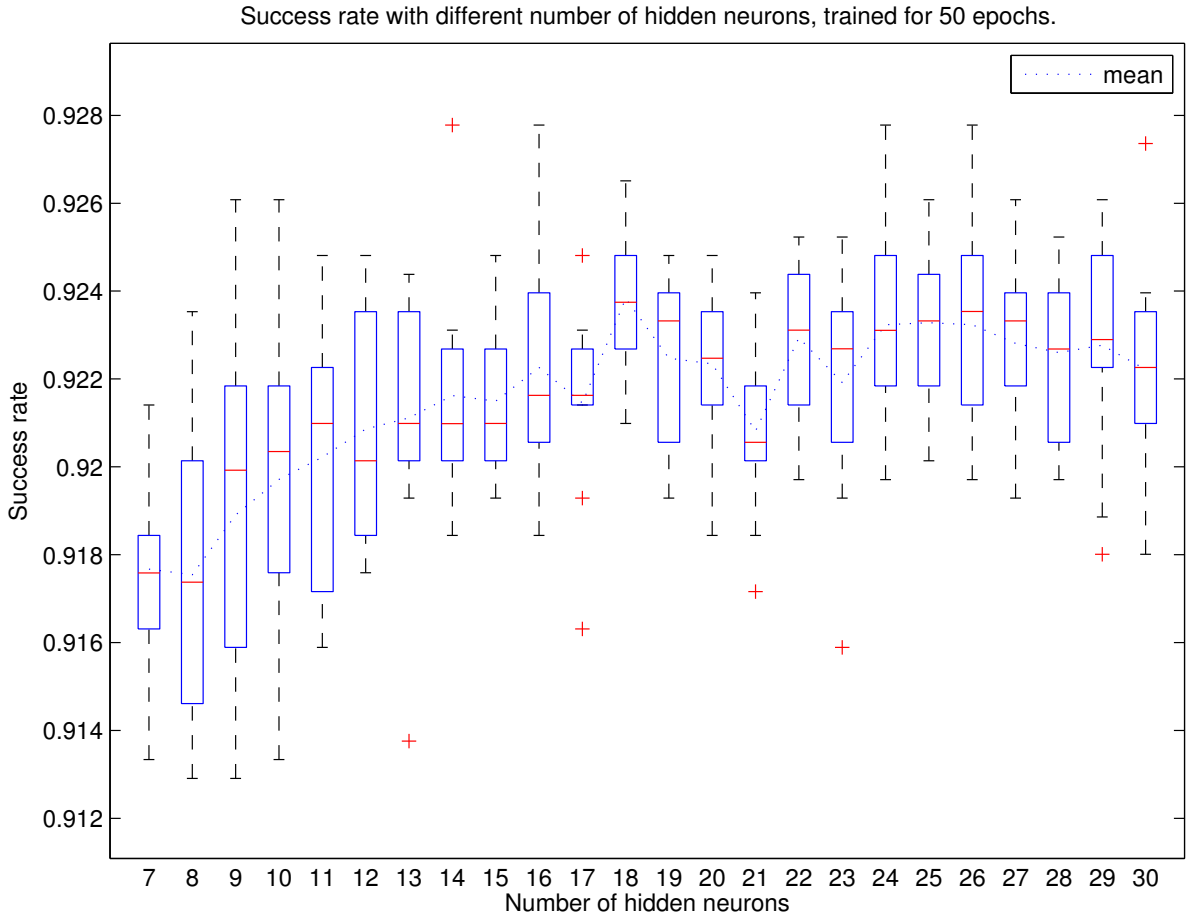
7. Door te kijken naar de *Mean square error* (MSE) op te trainingset, en op de validatieset. Zolang ze beide dalen, wordt het netwerk beter. Als de MSE op de trainingsset daalt, maar die op de validatieset gaat stijgen is er sprake van overfitting: het netwerk wordt beter in het herkennen van ruis in de trainingsset, maar niet beter in het herkennen van de algemene patronen.

8. Op het moment dat het netwerk overtrained dreigt te raken, dus wanneer de MSE niet langer daalt en en begint te stijgen, wordt de training gestopt, zodat het nog wel geschikt is voor andere data dan de trainingsset. Daarnaast is het handig voor de snelheid van het netwerk te stellen dat wanneer de MSE van het netwerk over tien epochs niet meer dan 0.1% verschilt, ook het trainingsproces te beëindigen. Gebeurt dit niet, dan kan het nog erg lang doorgaan met optimaliseren, zonder dat er in het uiteindelijke resultaat verschil merkbaar is.

9. Wanneer het netwerk tienmaal getraind wordt, komt hij iedere keer redelijk op dezelfde waarde voor de MSE uit. Slechts eenmaal laat hij een totaal andere curve zien. Dit komt waarschijnlijk doordat de geïnitialiseerde waardes van de gewichten en vergelijkingswaarden ver van een lokaal optimum liggen. Aan de andere lijnen valt te zien dat de meeste lokale optima dicht bij elkaar in de buurt liggen.



10. We hebben 23 verschillende netwerken, met 7 - 30 *hidden neurons* 10 keer getraind. De succes-ratio van die netwerken hebben we verwerkt in de volgende boxplots: Hieruit blijkt ondermeer dat de

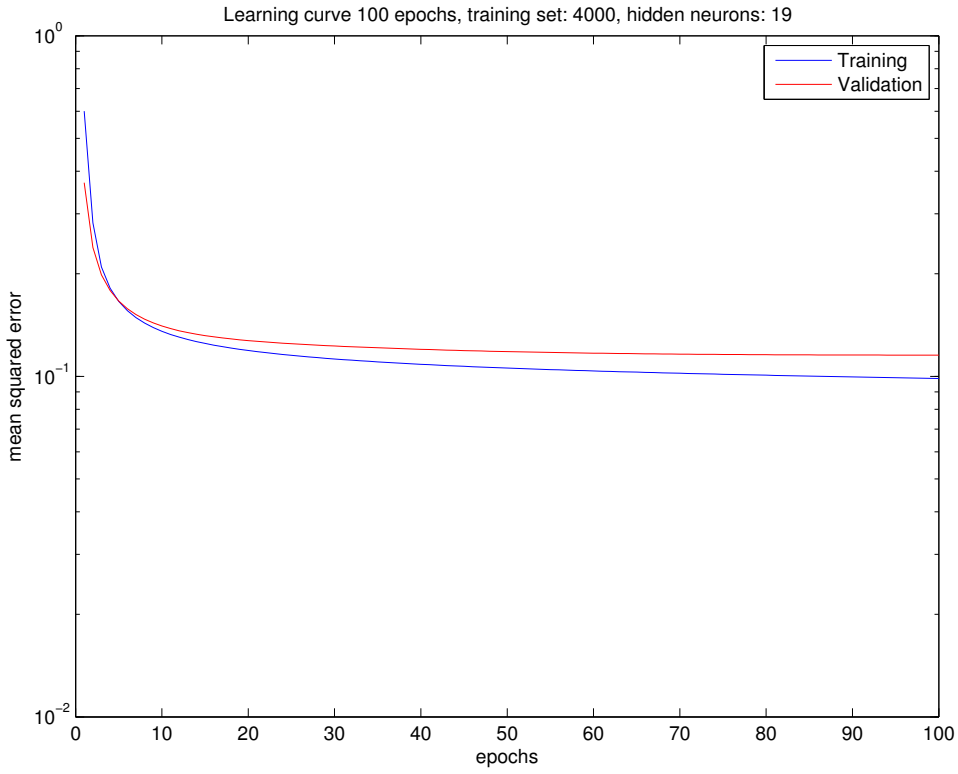


performance ook bij een klein aantal *hidden neurons* nog best goed is. Naast dit plaatje hebben we nog gekeken naar kleinere aantallen en zelfs met 4 *hidden neurons* wordt in gemiddeld 85% van de gevallen nog een correcte indeling in klassen gemaakt.

Verder is duidelijk dat het op een gegeven moment geen zin meer heeft om nog meer *hidden neurons* toe te voeggen, het resulterende netwerk heeft er geen significant betere performance door.

Wat ook interessant is om bij grotere aantallen neuronen te kijken naar de gewichten w_{ij} en w_jk. Dat kan bijvoorbeeld met `imagesc`. Er blijkt dan dat een deel van de neuronen bijna niet meer mee doet in de berekening van de output. Helaas is er in dit verslag geen ruimte voor plaatjes hiervan, laat staan voor een animatie van de plaatjes gedurdende de training.

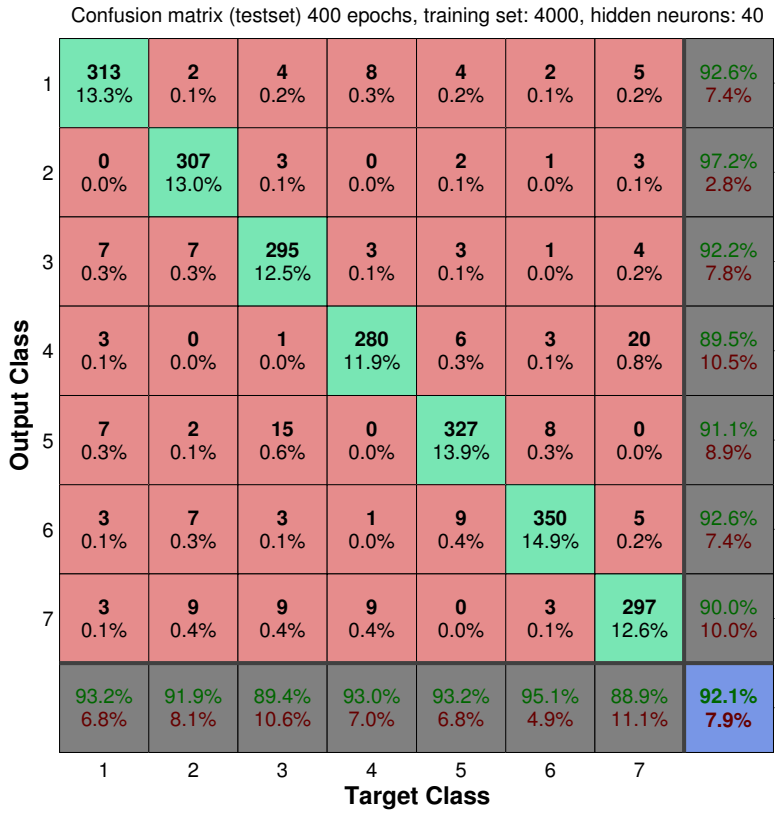
11. Hier een afbeelding van een netwerk met 19 *hidden neurons*. In de voorgaande boxplot kwam 18 hidden neurons beter uit de bus, maar dit was de beste prestatie uit een groot aantal runs en het netwerk wat we gebruikt hebben om de class van de set `unknown.txt` te bepalen.



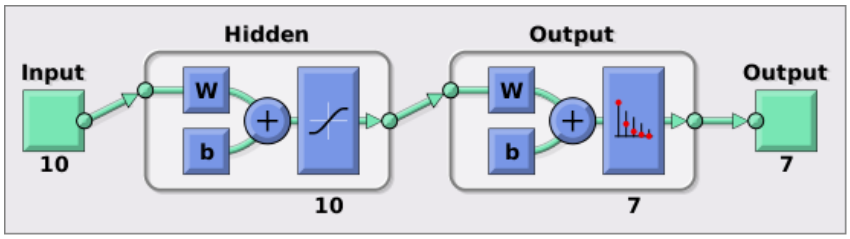
12. De *success rate* van de testset is 0.91 $\pm$0.01. Dit is vergelijkbaar met die van de validatieset. Het netwerk wordt echter wel geoptimaliseerd met behulp van de *success rate* van de validatieset door de training te stoppen op het moment dat deze weer stijgt. Dit heeft ook een positief effect op de *success rate* van de test set, maar het kan net buiten het optimum liggen. Dit is echter verwaarloosbaar in vergelijking met de verschillen tussen verschillende trainingsrondes.

13. Een confusionplot gemaakt met `plotconfusion` (zie volgende pagina). Op de y-as staan de output-classes zoals ons netwerk produceert. Op de x-as staat de verwachtingswaarde. Een juist voorspelde waarde staat dus op de diagonaal, in de groene cellen.
Verder is er ook uit af te lezen dat in dit geval het netwerk 20 keer een 4 voorspelde, terwijl het een 7 had moeten zijn. Dat is een duidelijke uitschieter waar verder voornamelijk fouten onder de 10 staan.

14. Geüpload naar blackboard als `5_classes.txt`



15. Met `nprtool` is het binnen een paar minuten mogelijk om een netwerk te maken wat er zo uitziet:



16. De performance van dit netwerk is erg goed, en de training is ook erg snel klaar. Ook met 10 hidden neurons worden succes-percentages van rond de 92% gehaald.

    Tijdens het maken van het eigen neurale netwerk had ik het idee dat het gebruiken van de sigmoid-activatiefunctie wellicht voor de output-layer niet handig was. Dit netwerk gebruikt een softmax activatiefunctie in de output-layer, wellicht ook een reden van betere prestaties.

    De veel hogere trainingsefficientie is waarschijnlijk verklaarbaar door bijvoorbeeld implementatie in correcte matrix-algebra en het toevoegen van het aanpassen van de *learning rate*, het toevoegen van een momentum-term in de delta-rule (Negnevitsky, equation 6.17).