COMP9444 Neural Networks and Deep Learning Term 2, 2025

Week 4 Tutorial Solutions

This page was last updated: 06/28/2025 14:07:43

1. **Softmax**

   Recall the formula for Softmax:

   $\text{Prob}(i) = \exp(z_i) \,/\, \Sigma_j \exp(z_j)$

   Consider a network trained on a classification task with three classes $1, 2, 3$.
   When the network is presented with a particular input, the output values are:

   $z_1 = 1.0,\ z_2 = 2.0,\ z_3 = 3.0$

   Suppose the correct class for this input is $\text{Class } 2$.
   Compute the following, to two decimal places:

   a. $\text{Prob}(i)$, for $i = 1, 2, 3$

   $$\begin{aligned}\text{Prob}(1) &= e^1/(e^1+e^2+e^3) = 2.718/30.193 = 0.09\\ \text{Prob}(2) &= e^2/(e^1+e^2+e^3) = 7.389/30.193 = 0.24\\ \text{Prob}(3) &= e^3/(e^1+e^2+e^3) = 20.086/30.193 = 0.67\end{aligned}$$

   b. $d(\log \text{Prob}(2))/dz_j$, for $j = 1, 2, 3$

   $$\begin{aligned}d(\log \text{Prob}(2))/dz_1 &= d(z_2 - \log \Sigma_j \exp(z_j))/dz_1 = -\exp(z_1)/\Sigma_j \exp(z_j) = -0.09\\ d(\log \text{Prob}(2))/dz_2 &= d(z_2 - \log \Sigma_j \exp(z_j))/dz_2\\ &= 1 - \exp(z_2)/\Sigma_j \exp(z_j) = 1 - 0.24 = 0.76\\ d(\log \text{Prob}(2))/dz_3 &= d(z_2 - \log \Sigma_j \exp(z_j))/dz_3 = -\exp(z_3)/\Sigma_j \exp(z_j) = -0.67\end{aligned}$$

   Note how in each case, the partial derivative is proportional to the difference between the estimated probability and its target value (1 for the correct class; 0 for the incorrect classes).

2. **Identical Inputs**

Consider a degenerate case where the training set consists of just a single input, repeated 100 times. In 80 of the 100 cases, the target output value is 1;
in the other 20, it is 0. What will a back-propagation neural network predict for this example, assuming that it has been trained and reaches a global optimum?
If the loss function is changed from Sum Squared Error to Cross Entropy,
does it give the same result?
(Hint: to find the global optimum, differentiate the loss function and set to zero)

When Sum Squared Error is minimized, we have:

$$\begin{aligned} E &= 80*(z-1)^2/2 + 20*(z-0)^2/2 \\ dE/dz &= 80*(z-1) + 20*(z-0) \\ &= 100*z - 80 \\ &= 0 \quad \text{when} \quad z = 0.8 \end{aligned}$$

When Cross Entropy is minimized, we have:

$$\begin{aligned} E &= -80*\log(z) - 20*\log(1-z) \\ dE/dz &= -80/z + 20/(1-z) \\ &= (-80*(1-z) + 20*z) / (z*(1-z)) \\ &= (100*z - 80) / (z*(1-z)) \\ &= 0 \quad \text{when} \quad z = 0.8, \quad \text{as before.} \end{aligned}$$

---

3. **Hidden Unit Dynamics**

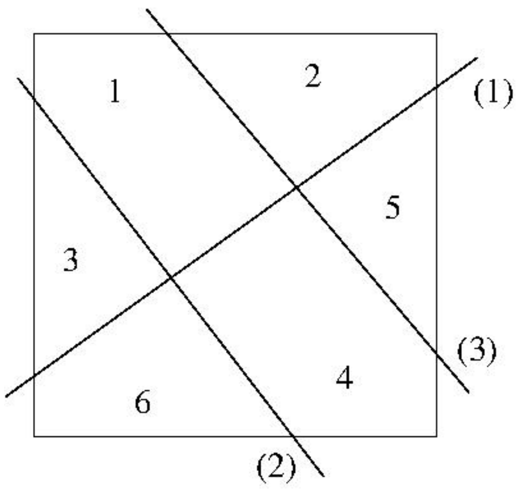
Consider a fully connected feedforward neural network with 6 inputs, 2 hidden units and 3 outputs, using tanh activation at the hidden units and sigmoid at the outputs. Suppose this network is trained on the following data, and that the training is successful.

Item	**Inputs**	**Outputs**
	123456	**123**
1.	100000	000
2.	010000	001
3.	001000	010
4.	000100	100
5.	000010	101
6.	000001	110

Draw a diagram showing:

a. for each input, a point in hidden unit space corresponding to that input, and

b. for each output, a line dividing the hidden unit space into regions for which the value of that output is greater/less than one half.



4. **Linear Transfer Functions**

Suppose you had a neural network with linear transfer functions. That is, for each unit the activation is some constant c times the weighted sum of the inputs.

a. Assume that the network has one hidden layer. We can write the weights from the input to the hidden layer as a matrix W^{IH}, the weights from the hidden to output layer as W^{HO}, and the bias at the hidden and output layer as vectors b^H and b^O. Using matrix notation, write down equations for the value O of the units in the output layer as a function of these weights and biases, and the input I. Show that, for any given assignment of values to these weights and biases, there is a simpler network with no hidden layer that computes the same function.

Using vector and matrix multiplication, the hidden activations can be written as:

$H = c * (b^H + I * W^{IH})$

(Following PyTorch conventions, we use matrix multiplication with the input activations on the left and the network weights on the right)

The output activations can be written as:

$$\begin{aligned} O &= c * [b^O + H * W^{HO}] \\ &= c * [b^O + c * (b^H + I * W^{IH}) * W^{HO}] \\ &= c * [(b^O + c * b^H * W^{HO}) + (c * I * W^{IH} * W^{HO})] \end{aligned}$$

Due to the associativity of matrix multiplication, this can be written as:

$O = c * (b^{IO} + I * W^{IO})$

where

$b^{IO} = b^{O} + c * b^{H} * W^{HO}$

$W^{IO} = \ c * W^{IH} * W^{HO}$

Therefore, the same function could be computed by a simpler network, with no hidden layer, using the weights b^{IO} and W^{IO}.

b. Repeat the calculation in part (a), this time for a network with any number of hidden layers. What can you say about the usefulness of linear transfer functions?

By removing the layers one at a time as above, a simpler network with no hidden layer could be constructed which computes exactly the same function as the original multi-layer network.
In other words, with linear activation functions, you don't get any benefit from having more than one layer.

5. **Group formation, finalising project, and finding dataset(s)**

   This Week's **checkpoint tasks** are as follows:

   a. Form a team for the group project, and enter the details into Moodle.
   b. Finalise the project topic based on the Project Description List in WebCMS.
   c. Familiarise yourself with the Project Marking Criteria available on WebCMS → Project → Project Marking Criteria
   d. Read the Project Description and access dataset(s) relevant to the project.
   e. Set up some form of communication channel to organize meetings, discuss about the project, keep track of other members' contributions, and make sure you are working as a team.
   f. Your tutor is the best person to contact if you have any questions related to the group project. Given they will be one of the assessors for the group project marking, it's important to seek their advice whenever you have a doubt on any aspects of the group project.