Medical Data Science
University of Tuebingen
WS 2023/2024
Atahan Özer - 6317973
Karahan Sarıtaş - 6661689

Assignment V

Problem 1

(a) The surge in popularity of deep learning can be attributed to two key advancements in the field: the improvement of computational power (with the introduction of GPUs and later TPUs) and the substantial increase in available data volumes by the early 2000s. Computational power, the growing abundance of available data, and the elimination of the manual requirement for feature extraction have collectively made deep learning preferable to other machine learning algorithms in application areas such as computer vision and natural language processing (NLP). Currently, deep learning algorithms are applied in lots of domains and tasks, including but not limited to, self-driving cars, object detection, generative AI, speech recognition, drug discovery, content-based recommendation systems, and medical diagnosis.

(b) Sparse coding is a way of learning how to represent the input data (representation learning) as a *sparse* linear combination of bases $\{\phi_1, \phi_2, ..., \phi_k\}$ such that most of the coefficients are equal to zero, hence earning the name "sparse" coding. Sparse coding can be trained using the reconstruction error between the actual input and the projection onto the new basis.

$$x_n = \sum_{k=1}^{K} a_{nk}\phi_k$$

Sparse coding can be employed to acquire "feature representations" of input data. For instance, it can be utilized for learning edges in image classification tasks. The resulting image representation, expressed in terms of the learned bases, can then be input into a machine learning algorithm (such as SVM) for classification.

(c) Autoencoders learn two functions: an encoder and a decoder. The encoder transforms the input data into (generally) a lower-dimensional space (latent representation). The decoder reconstructs the input data from the encoded latent representation while attempting to minimize the reconstruction error as much as possible.

There are different motivations for using encoders and decoders rather than simply using the image. Firstly, the encoder learns a mapping from the input image to a lower-dimensional space by extracting the important features (so that the decoder can revert it). This is a way of representation learning, and we can feed a machine learning algorithm (such as SVM) with the encoded data. Secondly, dimensionality reduction leads to a decrease in the training time of the algorithm. Thirdly, autoencoders can be used to filter out noise in the input image. If we do not impose any

constraint, both the encoder and decoder can simply learn an identity mapping to minimize the reconstruction error. One way to avoid learning an identity map is by applying an activation function like sigmoid to the output of the encoder Wx, where W is the learned weight matrix, and x is the input data. Furthermore, the encoder serves to avoid identity mapping by reducing the input to a lower dimension, as is usually the case.

(d) In Deep Learning, an activation function is a function applied to the output of a layer before progressing to the next layer. Activation functions are essential for introducing non-linearity into the network and determining whether the output of a neuron should be passed on to the next layer or zeroed out (e.g., ReLU). There are several activation functions such as Sigmoid $\sigma(x)$, $tanh(x)$, ReLU, and Leaky ReLU:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$
$$\text{ReLU} = \max(0, x)$$
$$\text{Leaky ReLU} = \max(\frac{x}{10}, x)$$

(e) Using a binary two-dimensional matrix, a protein contact map illustrates the interactions between all conceivable pairs of amino acid residues in a three-dimensional protein structure. If the distance between two amino acids falls below a predefined threshold, the corresponding matrix element is set to 1 (they are considered to be in contact); otherwise, it is set to 0.

(f) A 25-dimensional vector encapsulates the characteristics of each amino acid, encompassing 20 dimensions dedicated to evolutionary information (one for each amino acid type). Additionally, the vector incorporates three binary values signifying the anticipated secondary structure of the amino acid (α-helix, β-sheet, coil) and two binary values indicating the predicted accessibility of the amino acid within the three-dimensional structure (buried or exposed).

(g) Here, L represents the length of the amino acid sequence. To measure the performance of the algorithm, we only consider the contact pairs with top $\frac{L}{5}$ scores. Accuracy is calculated using the precision $= \frac{\text{TP}}{\text{TP+FP}}$ Considering only the top $\frac{L}{5}$ scores, allows a focused assessment of the accuracy of high-ranking predictions in comparison to the true contacts within a protein structure.

(h) To assess the dependence between random variables, one may calculate the correlation coefficient between them. Pearson correlation coefficient can be used to identify linear correlations between the variables. A non-negative result indicates a correlation, either positive or negative, between the variables. However, a result of zero does not lead to any conclusive evidence, as there might be a non-linear correlation as well. *Mutual information* is another methodology to measure the dependence between two random variables that are sampled simultaneously. In an intuitive sense, mutual information seeks to quantify the extent to which having information about one of these variables reduces the uncertainty linked to the other. Output is positive if the random variables are correlated and 0 if not. Mutual information is defined

as follows: (for discrete case)

$$\mathrm{MI}(X;Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}$$

For continuous case:

$$\mathrm{MI}(X;Y) = \int_{\Omega_x}\int_{\Omega_y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}dydx$$

(i) Spatial Pooling, also known as subsampling or downsampling, diminishes the dimensionality of individual feature maps while preserving crucial information. Various types of Spatial Pooling exist, including max, average, sum, and others. It's important to note that max pooling inherently involves a process with information loss as we are losing other pixels in the frame. However, it is used to (1) reduce the size of the representation (and therefore the training time), (2) reduce the number of parameters and computations in the network, (3) make the image more robust against small transformations, distortions and translations and (4) help us achieve almost a scale-invariant representation of the image.

Problem 2

(a) $p(c) = \sum_d p(d,c) = \sum_s \sum_d p(d,c,s)$

$$p(c) = \begin{cases} p(c=0) = p(0,0,0) + p(0,0,1) + p(1,0,0) + p(1,0,1) \\ p(c=1) = p(0,1,0) + p(0,1,1) + p(1,1,0) + p(1,1,1) \end{cases}$$

$$p(c) = \begin{cases} p(c=0) = 0.06 + 0.13 + 0.06 + 0.17 \\ p(c=1) = 0.04 + 0.23 + 0.04 + 0.27 \end{cases}$$

$$p(c) = \begin{cases} p(c=0) = 0.42 \\ p(c=1) = 0.58 \end{cases}$$

(b) $p(d) = \sum_c p(d,c) = \sum_s \sum_c p(d,c,s)$

$$p(d) = \begin{cases} p(d=0) = p(0,0,0) + p(0,0,1) + p(0,1,0) + p(0,1,1) \\ p(d=1) = p(1,0,0) + p(1,0,1) + p(1,1,0) + p(1,1,1) \end{cases}$$

$$p(d) = \begin{cases} p(d=0) = 0.06 + 0.13 + 0.04 + 0.23 \\ p(d=1) = 0.06 + 0.17 + 0.04 + 0.27 \end{cases}$$

$$p(d) = \begin{cases} p(d=0) = 0.46 \\ p(d=1) = 0.54 \end{cases}$$

(c) $p(d,c) = \sum_s p(d,c,s)$

$$p(d,c) = \begin{cases} p(d=0,c=0) = p(0,0,0) + p(0,0,1) \\ p(d=0,c=1) = p(0,1,0) + p(0,1,1) \\ p(d=1,c=0) = p(1,0,0) + p(1,0,0) \\ p(d=1,c=1) = p(1,1,0) + p(1,1,1) \end{cases}$$

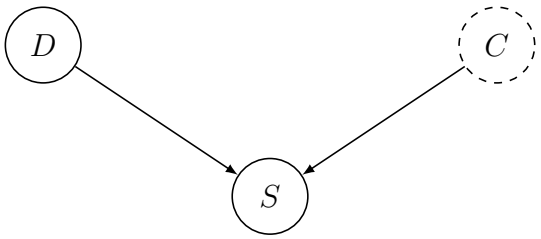
$$p(d,c) = \begin{cases} p(d=0,c=0) = 0.06 + 0.13 \\ p(d=0,c=1) = 0.04 + 0.23 \\ p(d=1,c=0) = 0.06 + 0.17 \\ p(d=1,c=1) = 0.04 + 0.27 \end{cases}$$

$$p(d,c) = \begin{cases} p(d=0,c=0) = 0.19 \\ p(d=0,c=1) = 0.27 \\ p(d=1,c=0) = 0.23 \\ p(d=1,c=1) = 0.31 \end{cases}$$

(d) $p(s|d,c) = \frac{p(d,c,s)}{p(d,c)}$

$$p(s|d,c) = \begin{cases} p(s=0|d=0,c=0) = \frac{0.06}{0.19} = 0.32 \\ p(s=0|d=0,c=1) = \frac{0.04}{0.27} = 0.15 \\ p(s=0|d=1,c=0) = \frac{0.06}{0.23} = 0.26 \\ p(s=0|d=1,c=1) = \frac{0.04}{0.31} = 0.13 \\ p(s=1|d=0,c=0) = \frac{0.13}{0.19} = 0.68 \\ p(s=1|d=0,c=1) = \frac{0.23}{0.27} = 0.85 \\ p(s=1|d=1,c=0) = \frac{0.17}{0.23} = 0.74 \\ p(s=1|d=1,c=1) = \frac{0.27}{0.31} = 0.87 \end{cases} \quad \forall(d^*,c^*)\ \sum_s p(s|d^*,c^*) = 1$$

(e) We cut the arcs coming into any nodes that were set by intervention:



We know that $p(d|\text{do}(c)) = p(d)$. Joint distribution $p(d,c,s)$ can be expressed as follows:

$$\begin{aligned} p(d,\text{do}(c),s) &= p(s|\text{do}(c),d) \cdot p(d|\text{do}(c)) \cdot p(\text{do}(c)) \\ &= p(s|\text{do}(c),d) \cdot p(d) \cdot p(\text{do}(c)) \end{aligned}$$

We already know $p(s|do(c),d) = p(s|c,d)$ and $p(d)$.

$$p(s,d,\text{do}) = \begin{cases} p(s=0,d=0,\text{do}(c=0)) = 0.32 \cdot 0.46 \cdot p(\text{do}(c=0)) = 0.1472 \cdot p(\text{do}(c=0)) \\ p(s=0,d=0,\text{do}(c=1)) = 0.15 \cdot 0.46 \cdot p(\text{do}(c=1)) = 0.069 \cdot p(\text{do}(c=1)) \\ p(s=0,d=1,\text{do}(c=0)) = 0.26 \cdot 0.54 \cdot p(\text{do}(c=0)) = 0.1404 \cdot p(\text{do}(c=0)) \\ p(s=0,d=1,\text{do}(c=1)) = 0.13 \cdot 0.54 \cdot p(\text{do}(c=1)) = 0.0702 \cdot p(\text{do}(c=1)) \\ p(s=1,d=0,\text{do}(c=0)) = 0.68 \cdot 0.46 \cdot p(\text{do}(c=0)) = 0.3128 \cdot p(\text{do}(c=0)) \\ p(s=1,d=0,\text{do}(c=1)) = 0.85 \cdot 0.46 \cdot p(\text{do}(c=1)) = 0.391 \cdot p(\text{do}(c=1)) \\ p(s=1,d=1,\text{do}(c=0)) = 0.74 \cdot 0.54 \cdot p(\text{do}(c=0)) = 0.3996 \cdot p(\text{do}(c=0)) \\ p(s=1,d=1,\text{do}(c=1)) = 0.87 \cdot 0.54 \cdot p(\text{do}(c=1)) = 0.4698 \cdot p(\text{do}(c=1)) \end{cases}$$

In total it sums up to $p(\text{do}(c=0))+p(\text{do}(c=1)) = 1$. Now, I assume that $p(do(c)) = p(c)$; in other words, we decide whether the chief surgeon operates with the same probability as before. Then the joint distribution table becomes the following:

d	$\text{do}(c)$	s	$p(d, \text{do}(c), s)$
0	0	0	0.062
0	0	1	0.131
0	1	0	0.040
0	1	1	0.227
1	0	0	0.059
1	0	1	0.168
1	1	0	0.041
1	1	1	0.272

During the calculations, we perform rounding, and therefore, there may be small errors in the comparison results. Nevertheless, these discrepancies are still sufficient for interpretation. The following table shows the differences between the first and the second tables:

d	$\text{do}(c)$	s	$p(d, \text{do}(c), s)$
0	0	0	+3.33%
0	0	1	+0.77%
0	1	0	-
0	1	1	-1.30%
1	0	0	-0.6%
1	0	1	-1.17%
1	1	0	+2.5%
1	1	1	+0.74%

The likelihood of the chief surgeon performing operations during the daytime ($p(d = 1, c = 1)$) appears to increase, while the probability of another surgeon operating at night ($p(d = 0, c = 0)$) also shows an upward trend. Conversely, the probability of the chief surgeon operating at night ($p(d = 0, c = 1)$) and another surgeon performing procedures during the daytime ($p(d = 1, c = 0)$) seems to decrease.

Problem 3

(a) Below you can find the screenshot of the final result on circle dataset.

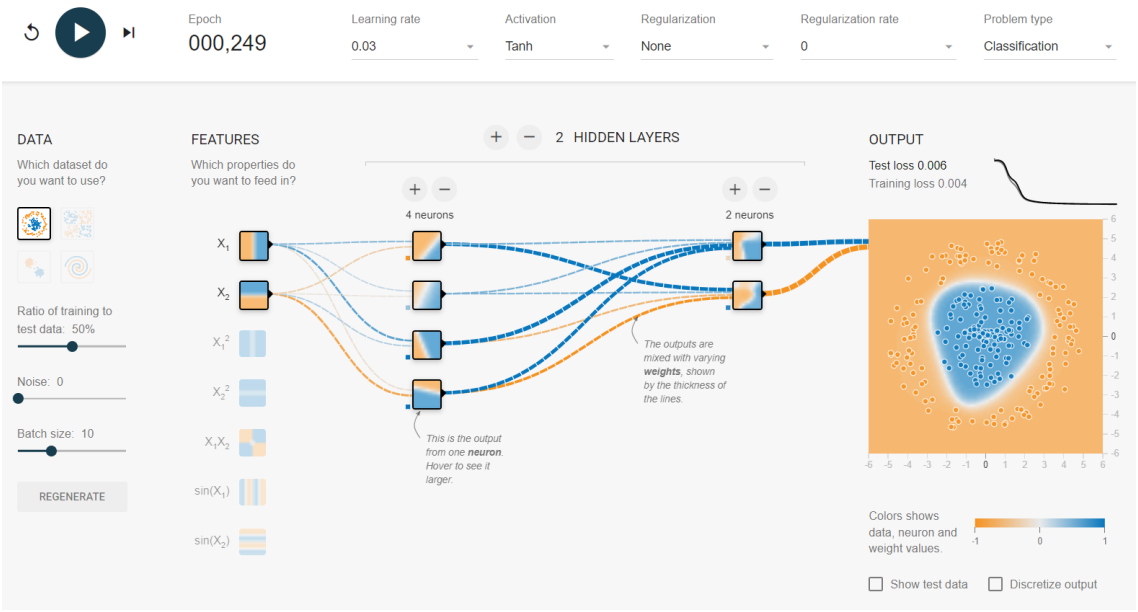


Figure 1: Results on the circle dataset

Parameters	Value
Learning Rate	0.03
Activation	TanH
Regularization	None
Features	X_1 and X_2
Number of Hidden Layers	2 (4, 2)
Ratio of training to test data	50%
Batch Size	10

(b) X_1 and X_2 are used as features. According to the final result, the neurons in the first layer learn straight lines that separate the plane into two parts. Then these lines are combined with different weights to produce two partial circles in the last layer. These partial circles are combined (with similar weights) to produce the final circle.

(c) Below you can find the screenshot of the final result on the spiral dataset.

Parameters	Value
Learning Rate	0.03
Activation	TanH
Regularization	None
Features	X_1, X_2, $\sin(X_1)$ and $\sin(X_2)$
Number of Hidden Layers	2 (5, 2)
Ratio of training to test data	50%
Batch Size	10

(d) This time we used X_1, X_2, $\sin(X_1)$ and $\sin(X_2)$ to learn the function which we expect to be highly non-linear. In the first hidden layer, we achieve non-linear functions with fluctuations. Here we can see that the weights of the trigonometric functions of

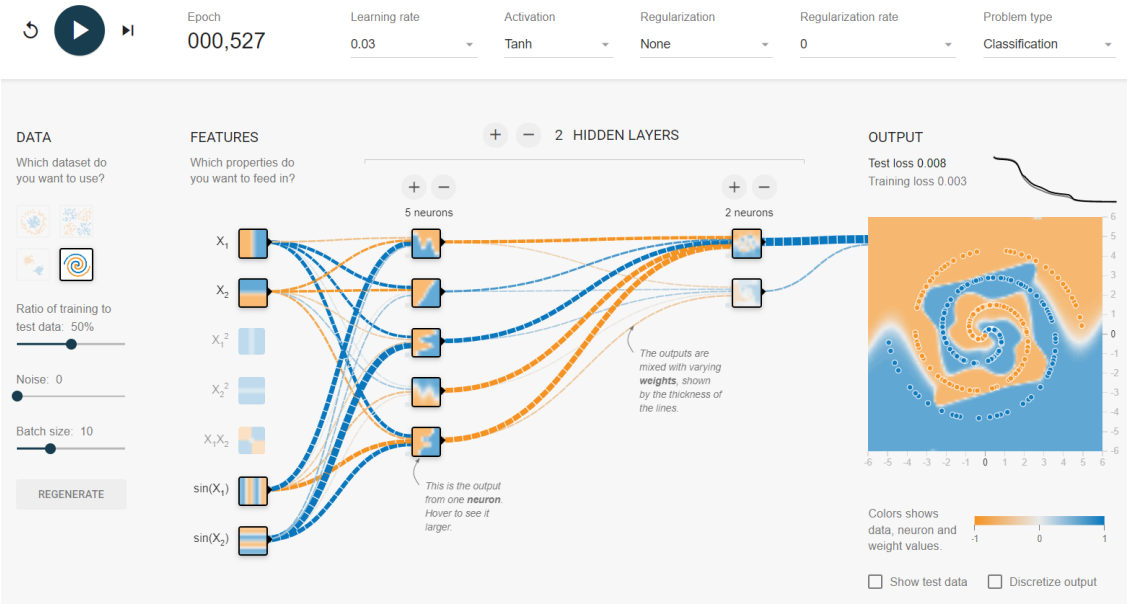


Figure 2: Results on the spiral dataset

the input data are way higher compared to the weights of X_1 and X_2. In the second layer, a spiral-like shape has already been attained. Interestingly, the output of the first neuron aligns more closely with the data distribution than the second output. As a result, the weight of the second output is disproportionately low compared to that of the first one, contributing very little to the final output. Despite the addition of only one extra neuron in the first hidden layer compared to the previous model, this time it took approximately 500 epochs to achieve a reasonable test loss in fitting the data.