Deep Learning Homework 1: Linear and Neural Network Classifiers

Group Members and Contributions:

107004	Diogo Martins Prata Pinto de Sousa
105855	Diogo Miguel Ferra dos Santos Pica
117194	Florian Huhnd

Florian Huhnd worked on Question 1. Diogo Sousa worked on Question 2. Diogo Pica worked on Question 3. We presented each other the results for each question and put the results together in this report. AI was used as described in each section.

Question 1: Handwritten Letter Classification with Linear Classifiers (35 points)

How AI was used in this section

Exercise 1 could easily be derived from the solutions of practical_02, I only needed a little help to convert the functions from explicitly taking the weights as an argument to taking the *self*-object.

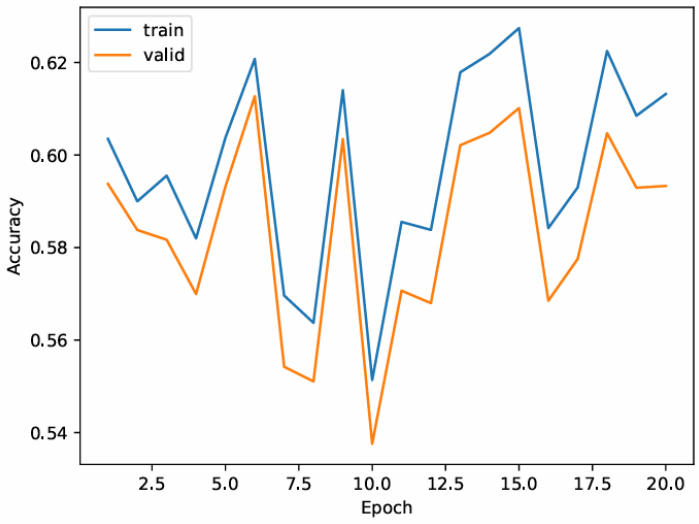
In exercise 2 I let AI create some helper functions to actually display the hand-written letters, as my first idea was to cut off unnecessary edges. It turned out that there were no pixels really unused, however the function still came in handy to show the downsampled images. The downsampling itself was implemented by AI as well, but I went through it line by line to make sure I understood what it was doing. The general structure of the solution still follows practical_03 and the lecture slides; AI was mainly used to debug and clean up code I had already written. For the grid search, I took my working code from the exercises (a) and (b) and let AI construct the grid search framework, which I adapted to my preferences later.

For exercise 3 I tried to stick as close to both the previous exercises as well as practical_04. Since they followed slightly different structures had some help implementing the results of the practical into the structure of the previous framework without losing functionality. Otherwise I mainly used AI to help me explain mistakes I made while trying to implement the methods by myself based on the practical lessons. I discarded some ideas supposed to improve speed or stability, as I felt they went far beyond the scope of what was taught in the lecture so far.

1. Perceptron (10 points)

The best validation accuracy occurred in epoch 6 with 61.27%, having a training accuracy of 62.08%. The test accuracy was 61.11%. Training took around 26 seconds on my device.

The following plot shows the train and validation accuracies over each epoch.

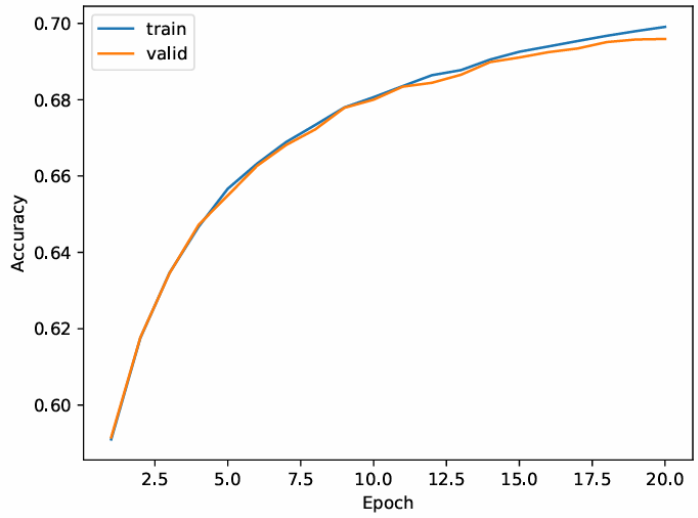


2. Logistic Regression (25 points)

(a) Baseline Logistic Regression with L2 Regularization (10 points)

The results stored in .json-file can be seen in the table below, followed by the plot of the train and validation accuracies over each epoch. It shows. That the accuracy increased with each epoch.

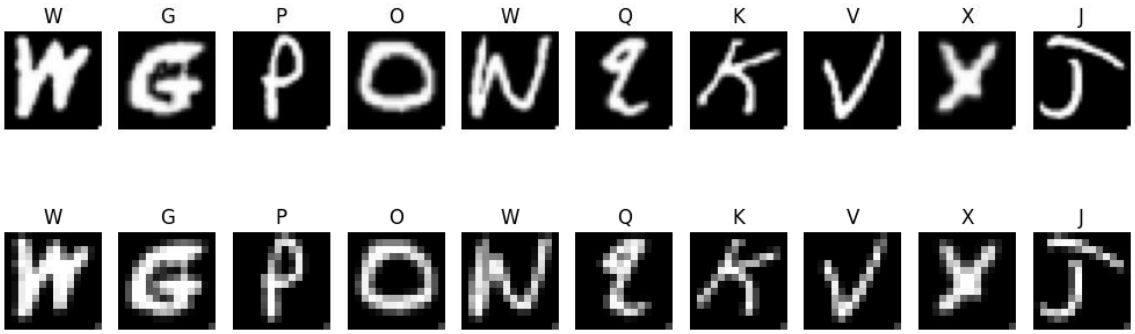
best_valid	selected_epoch	test	time
0.6959134615384616	20	0.6975	169.80031561851501



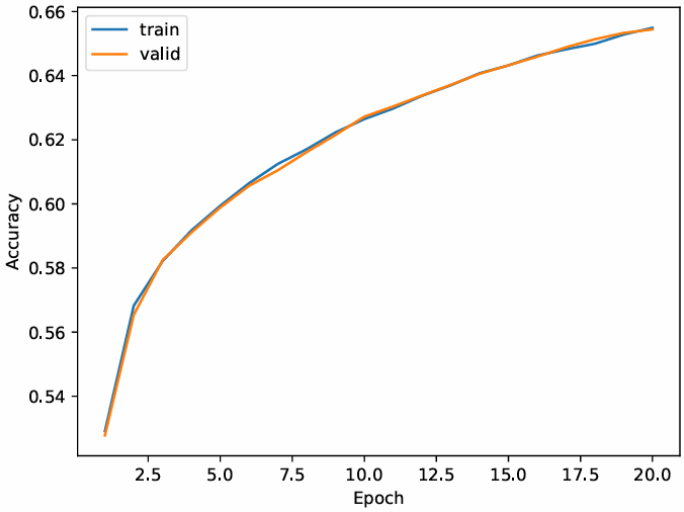
(b) Alternative Feature Representation (5 points)

Blurring the images by averaging the grey scale value of four adjacent pixels into one bigger cell reduces the feature size by factor 4 from 28x28 to 14x14, making it a lot more efficient without losing much of accuracy, as shown later. As stated above, I also thought about reducing feature size by removing the outer lines of pixels, but displaying the images showed me that there was not a lot to cut off. The idea of downsampling is pretty self-explaining so I did not take it from a specific reference, however I just recently watched this great explanation of what a convolution does by mathematician and Youtuber Grant Sanderson ("3 Blue 1 Brown"), which probably helped to immediately come up with this solution: But what is a convolution?

The following two images compare the regular 28x28 images (top) with the downsampled 14x14 images (bottom). The table and the plot below show the final results on the downsampled images. A comparison with the results from 2 (a) shows, that the code was twice as fast while still having a test accuracy of 65.21% (compared to the original test accuracy of 69.75%).



best_valid	selected_epoch	test	time
0.6544230769230769	20	0.6520673076923077	84.73830533027649



(c) Grid Search over Hyperparameters (10 points)

The following picture shows the output table of the grid search including the validation accuracies (best_valid_acc), test accuracies (test_acc) and the training time in seconds (time_seconds).

```
Grid search results:
    eta      l2    features  best_valid_acc  test_acc  time_seconds
0.00001 0.00001         raw         0.61750   0.61716     177.29950
0.00001 0.00001 downsampled         0.56471   0.56351      91.61737
0.00001 0.00010         raw         0.61755   0.61712     187.97646
0.00001 0.00010 downsampled         0.56476   0.56341     101.41750
0.00010 0.00001         raw         0.69712   0.69692     180.16277
0.00010 0.00001 downsampled         0.65428   0.65245      98.88763
0.00010 0.00010         raw         0.69702   0.69663     179.81706
0.00010 0.00010 downsampled         0.65385   0.65216     101.17302
0.00100 0.00001         raw         0.71995   0.72087     179.02185
0.00100 0.00001 downsampled         0.70976   0.71043      97.09031
0.00100 0.00010         raw         0.71904   0.72010     176.05964
0.00100 0.00010 downsampled         0.70846   0.70928      96.69972
```

The best configuration (judged by highest validation accuracy) is given by the following hyperparameters. The test accuracy achieved was 72.09%.

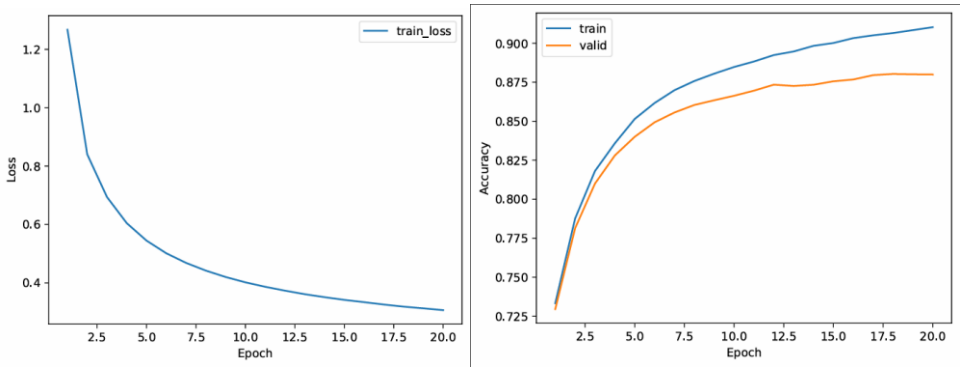
```
Best configuration (highest validation accuracy):
  eta: 0.00100
  l2: 0.00001
  features: raw
  Valid acc: 0.7200
  Test acc: 0.7209
```

3. Multilayer Perceptron

The results of the best performing checkpoint are presented in the table below.

best_valid	selected_epoch	test	time
0.8803365384615385	18	0.8775	426.5372622013092

The following two plots show the train loss and train and validation accuracies as a function of the epoch number.



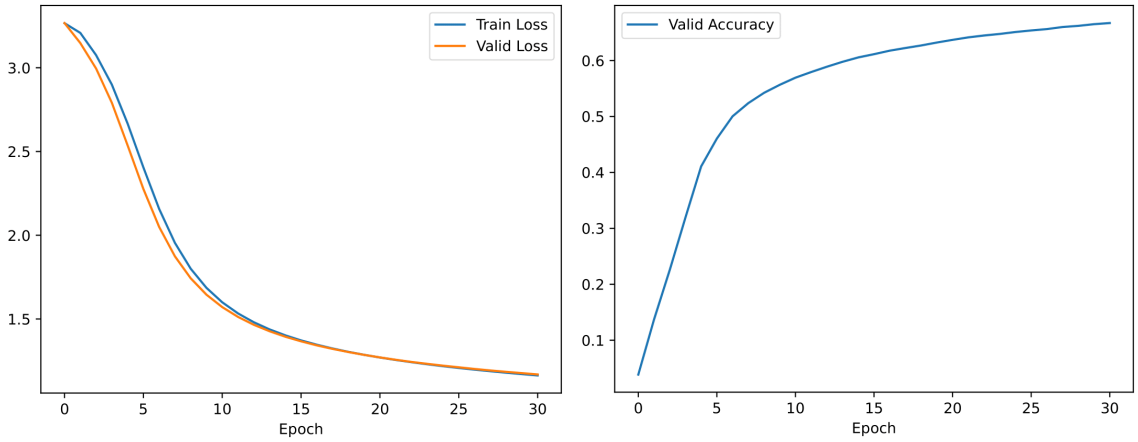
Question 2: PyTorch-Based Neural Networks (35 points)

How AI was used in this section

I used AI to assist with the parallelization of training tasks (2a)

1. FFN Implementation Warmup (5 points)

For this section, I based myself on practical class example. I then tested the code with the default arguments, obtaining the following results:



2. Infinite-Width One-Layer FFN (18 points)

(a)

For the grid search, I wrote a new version of the code, where I create all the configs by iteration and distribute tasks by the CPU cores. The parallelization saves runtime. I used the SGD optimizer and ReLU activation. For the unspecified parameters I chose the following values: Learning rates: 1e-4, 5e-4, 0.001, 0.01; Penalties: 0.0, 1e-5; Dropouts: 0.0 ,0.2.

Width	Learning Rates	Dropout	Penalty	Max validation accuracy
16	0.01	1.00E-05	0	0.746298075
32	0.01	1.00E-05	0	0.809134603
64	0.01	1.00E-05	0	0.845576942
128	0.01	1.00E-05	0	0.865769207

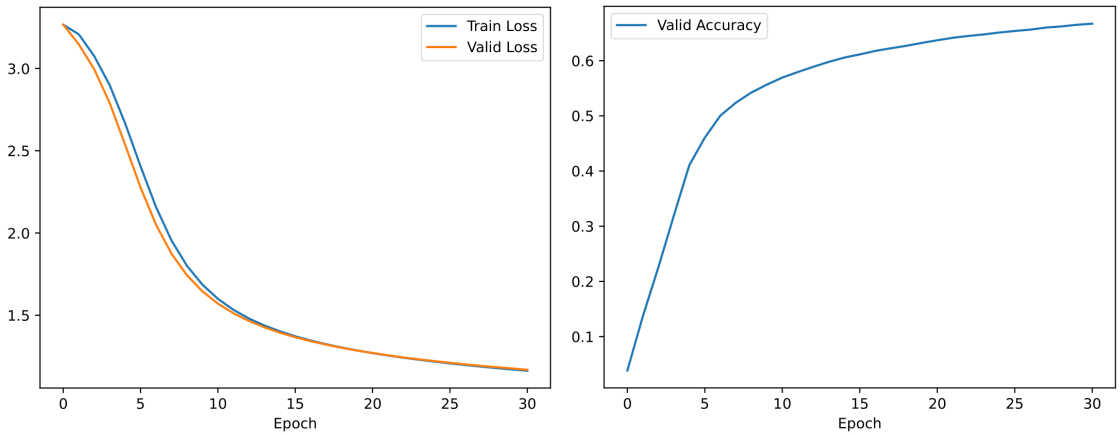
256	0.01	0	0.2	0.875817299

The best learning rate is 0.01 for every width. The best dropout and penalty are consistently 10^{-5} and 0, respectively, except for the last model (but even there the scores were very close).

It's clear that the validation accuracy improves as the layer width increases, but eventually converges. This behavior is further explained in section (c).

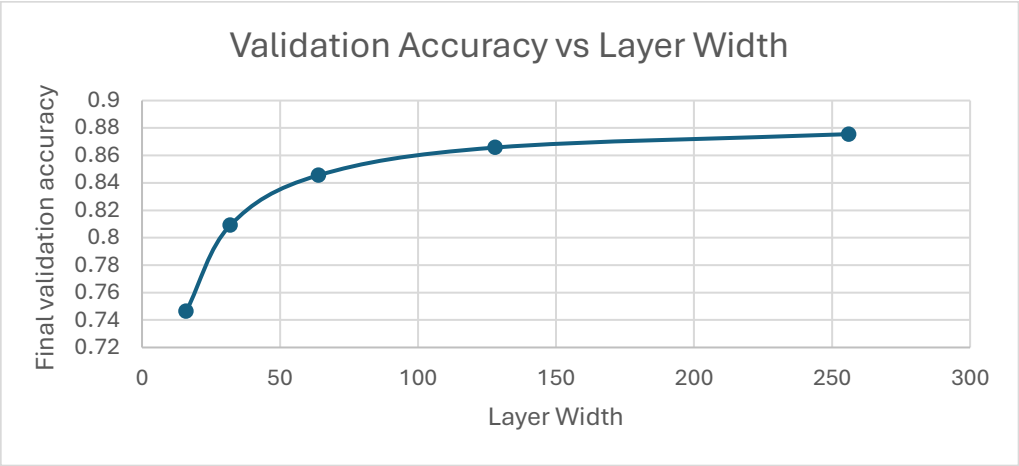
(b)

To get the plots for the best model, I ran the section 1 code with the flags: *-hidden_size 256 -learning_rate 0.01 -l2_decay 0.2*



The training and validation loss data looks healthy, dropping every epoch and still not having hit a plateau after 30 epochs. More epochs could bring the model closer to conversion. The high validation accuracy that was reached rules out underfitting. There is also no sign of overfitting, as the validation loss tightly tracks the training loss. The test accuracy was **87.53%**, which proves the generalization ability of the model, enabling it to perform well on unseen data.

(c)



As observed before, the validation accuracy rises, converging to ~88%. This model must learn the patterns in written letters, that requires some *memory* to capture them. For low widths neurons are shared by distinct features, leading to conflicts. The accuracy ceiling is related to the Bayes Error, which states that there is some irreducible error, which is inherent to the quality of the data (there are similar letter like 'u' and 'v' that can be indistinguishable for certain calligraphy variations). The Universal Approximation Theorem states that a sufficiently large network can approximate any continuous function to any desired degree of accuracy, therefore reaching the Bayes Error when approaching infinite width.

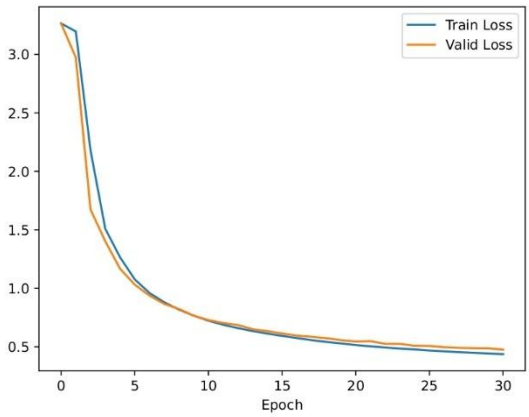
3. Effect of Network Depth (12 points)

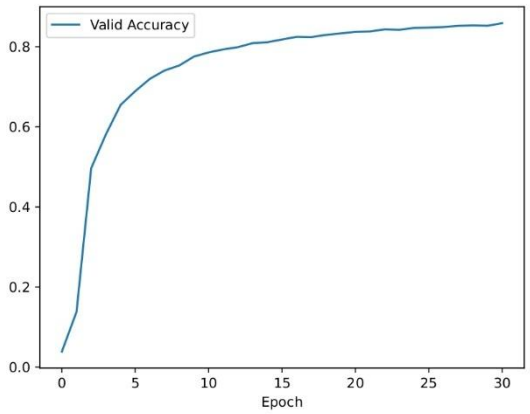
(a)

layers	Highest Validation Accuracy	Test Accuracy
1	0.807837	0.81
3	0.858798	0.85476
5	0.832212	0.831875
7	0.764423	0.763221
9	0.04476	0.043654

The accuracy increases for more than one layer, but quickly drops as we further increase past 5 layers.

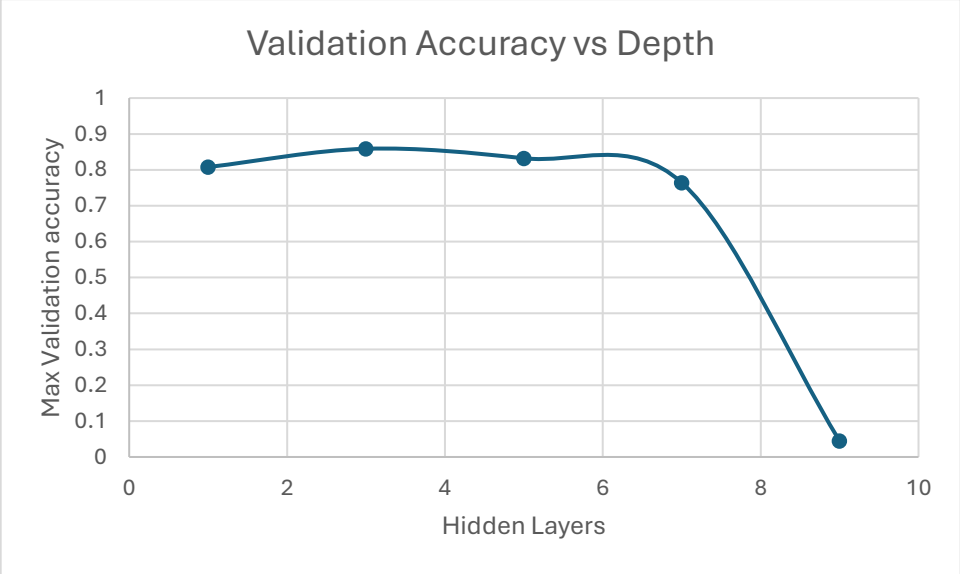
(b)





Just like in the previous exercise (2b), there are no signs of out underfitting or overfitting. This is evidenced by the closeness of training and validation curves, and the test accuracy of **87.04%**. This one, however, has converged faster.

(c)



As depth increases to 3 hidden layers, there is a significant rise in validation accuracy. This is because depth provides the ability to interpolate more complex features.

However, there is a point where adding more layers is not beneficial anymore, even causing the accuracy to eventually drop to 0. Having more layers decreases the trainability of the model, as earlier gradients are calculated with more multiplications,

shrinking when backpropagated through the layers. This is known as the Vanishing Gradient Problem.

Question 3: Normalized ReLU and Sparse Softmax Alternatives (30 points)

[This section involves mathematical proofs and closed-form derivations. Include your work below.]

How AI was used in this section

Exercise 1 and 2 AI was used to verify the proof, each mathematical step, and it to help simplify some mathematical operations.

For exercise 3, AI was used to clarify the mathematical steps and verify the results. It also helped generate Python code to plot the functions and their derivatives, ensuring the final expressions matched the expected behavior.

In exercise 4, AI was used to clarify mathematical derivation and verify the correctness of each differentiation step.

For exercise 5 AI was used to help mathematical operations and verify the results.

1. Proof that ReLU is Projection onto Non-negative Orthant (5 points)

In this question the objective is to prove that the ReLU activation function defined as

$$ReLU(z) \coloneqq \max(0,\, z)$$

is the solution to the optimization problem

$$arg \min_{y \geq 0} |y-z|^2$$

The optimization problem is constrained to $y \geq 0$. In this problem, we are trying to minimize the Euclidean norm. The Euclidean norm is separable across coordinates, in this case y and z, so the objective can be rewritten as

$$|y - z|^2 = \sum_{i=1}^{d} (y_i - z_i)^2$$

Knowing this the problem can be divided into d scalar subproblems with the form

$$\min_{y \geq 0} (y_i - z_i)^2.$$

The subproblem can be written as finding the solution y that minimizes the function $f(y) = (y - z)^2$. To solve this problem firstly we determine its derivative in respect to y

$$\frac{d}{dy} f(y) \;=\; 2(y - z)$$

And set it to 0

$$0 = 2(y - z)$$

By solving this equation, we get

$$y = z .$$

So, the solution that minimizes the subproblem is $y = z$, but we need to consider the constraint $y \geq 0$. For the previous constraint we have 2 cases, $z \geq 0$ and $z < 0$. When $z \geq 0$ the solution to the problem y is also ≥ 0 beacuse the solution is $y = z$, but when $z < 0$ the solution would be $y < 0$ which would violate the constraint, so the previous solution is not possible. The function $f(y)$ is increasing as the difference of z and y increases so to achieve the minimum value of $f(y)$ while not violating the constraint, y must be the smallest possible value in this case 0 so we get the following solution to the problem

$$y = \begin{Bmatrix} z, if\ z \geq 0 \\ 0, if\ z < 0 \end{Bmatrix}$$

Which is

$$y = \max(0, z) = ReLU(z).$$

Since the original problem is decomposed across d scalar subproblem we can define the solution to the problem as the solution to the subproblem which means

$$\underset{y \geq 0}{arg\ min} |y - z|^2 = ReLu(z).$$

2. ReLUMax and Scale Invariance (5 points)

Softmax, sparsemax and relumax are functions that are insensitive to adding a constant to z. This means substituting z by $z + c$ where $c \in \mathbb{R}$, does not influence the function output.

Softmax

$$soft\max_i(z + c) = \frac{e^{z_i + c}}{\sum_j e^{z_j + c}} = \frac{e^{z_i} e^c}{e^c \sum_j e^{z_j}} = soft\max_i(z)$$

Sparsemax

For any $p \in \Delta^{K-1}$ we have

$$|p - z - c|^2 = |p - z|^2 - 2c^T(p - z) + |c|^2 = |p - z|^2 + 1^T p = |p - z| + 1$$

With the previous equation we can conclude that

$$sparse\max(z - c) \coloneqq \underset{p \in \Delta^{K-1}}{\arg min} |p - z - c|^2 = sparse\max(z) \coloneqq \underset{p \in \Delta^{K-1}}{\arg min} |p - z|^2$$

Relumax

Using $z = z + c$, we calculate that $\max(z + c) = \max(z) + c$ and

$(z_i + c) - (\max(z) + c) + b = z_i - \max(z) + b$. With this we can conclude that

$$relu \max(z + c) = relu \max(z)$$

The softmax, sparsemax and relumax are functions that all approach the zero-temperature limit for this we want to solve $\lim_{T \to 0^+} f\left(\frac{z}{T}\right) = e_k$.

Softmax

$$softmax_i\left(\frac{z}{T}\right) = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$$

With $i \neq k$:

$$\frac{softmax_i(z/T)}{softmax_k(z/T)} = e^{(z_i - z_k)/T}$$

With $z_i - z_k < 0$ and $T \longrightarrow 0^+$, the exponent

$$\frac{(z_i - z_k)}{T} \to -\infty$$

So, the ratio tends to 0, with that we conclude

$$soft \max\left(\frac{z}{T}\right) \longrightarrow e_k.$$

Sparsemax

First, we start by applying the scaled vector αz were $\alpha = \frac{1}{T}$,

$$sparse \max_i(\alpha z) = [\alpha z_i - \tau(\alpha)]_+$$

Secondly, we choose the threshold such as:

$$\sum_i [\alpha z_i - \tau(\alpha)]_+ = 1$$

For large α the largest coordinates dominate. Using the previous statements and $j \neq k$ we get:

$$\alpha z_j - \tau(\alpha) = \alpha z_j - (\alpha z_k - 1) = \alpha\left(z_j - z_k\right) + 1$$

For the previous resolution we know $z_j - z_k < 0$ which means $\alpha(z_j - z_k) + 1 < 0$ for larger α, therefore those components are mapped to zero by the operator, and for large α we obtain we get:

$$sparse \max_k(\alpha z) = 1, \quad sparse \max_j(\alpha z) = 0 \quad (j \neq k)$$

With that previous result we can conclude that as the temperature goes to zero, sparsemax amplifies the differences between coordinates in a way that the highest entry dominates, and all others are set to zero and removed.

$$sparse \max(z/T) \to e_k$$

Relumax

Relumax is defined with b > 0 as

$$relu \max_b \left(\frac{z}{T}\right)_i = \frac{\left(\frac{z_i}{T} - \max\left(\frac{z}{T}\right) + b\right)^+}{\sum_j \left(\frac{z_j}{T} - \max\left(\frac{z}{T}\right) + b\right)^+}$$

Using the identity $\max\left(\frac{z}{T}\right) = \frac{\max(z)}{T}$

$$\frac{z_i}{T} - \max\left(\frac{z}{T}\right) + b = \frac{z_i - \max(z)}{T} + b$$

Case 1: $i = k$

Since $z_k = \max(z)$:

$$\frac{z_k - \max(z)}{T} + b = b > 0$$

After applying the ReLU function, the numerator tends to b.

Case 2: $i \neq k$

In this case $z_i - \max(z) < 0$ so:

$$\frac{z_i - \max(z)}{T} \longrightarrow -\infty$$

After applying the ReLU function, the numerator tends to 0.

Denominator

$$\sum_j \frac{z_j}{T} - \max\left(\frac{z}{T}\right) + b \ \to b$$

After knowing the denominator it's possible to conclude that

$$\lim_{T\to 0^+} r\,elu \max_b \left(\frac{z}{T}\right) = \left\{ \begin{matrix} 1, i=k \\ 0, i\neq k \end{matrix} \right\}$$

$$relu \max_b \left(\frac{z}{T}\right) \longrightarrow e_k.$$

In this part of the question, the objective is to prove that for every vector z exists a value b > 0 such that

$$relu \max_b(z) = sparse \max(z)$$

We start by writing both functions in the same format.

- **Sparsemax**

$$sparse \max_i(z) = (z_i - \tau)^+$$

$$\sum_i \quad (z_i - \tau)^+ = 1$$

- **Relumax**

$$relu \max_b(z)_i = \frac{(z_i - \max(z) + b)^+}{\sum_j \left(z_j - \max(z) + b\right)^+}$$

By writing both functions in the same format it's possible to identify that both functions have a constant shift b and τ so to make relumax and softmax match we need to find b in function of τ by solving the following equation

$$(z_i - \max(z) + b)^+ = (z_i - \tau)^+$$

After solving it we get $b \coloneqq \max(z) - \tau$.

After finding b we need to verify that

$$\sum_j \left(z_j - \max(z) + b\right)^+ = \sum_j \quad \left(z_j - \tau\right)^+ = 1$$

After this normalization and substituting it, we get the following equation:

$$relu \max_b(z)_i = \frac{(z_i - \max(z) + b)^+}{1} = (z_i - \tau)^+ = sparse \max_i(z)$$

Lastly, we need to verify that $b > 0$

$$(z_k - \tau)^+ > 0 \Rightarrow z_k - \tau > 0 \Rightarrow \tau < z_k \leq \max(z)$$

So $b := \max(z) - \tau > 0$.

After all the calculations it is possible to conclude that there is a positive b that for each z vector the equation is verified

$$relu \max_b(z) = sparse \max(z)$$

3. Closed-Form Analysis for K=2 (10 points)

Softmax

$$soft \max(z)_2 = \frac{e^t}{e^0 + e^t} = \frac{e^t}{1 + e^t}$$

$$\frac{dsoft \max(z)_2}{dt} = \frac{e^t}{(1 + e^t)^2}$$

Sparsemax

Defined as: $sparse \max(z)_i = \max(z_i - \tau, 0)$, $\sum_{k=1}^{2} \max(z_k - \tau, 0) = 1$

After derivations we obtain:

$$\text{sparsemax}(\mathbf{z})_2 = \begin{cases} 0, & t \leq -1, \\ \frac{t+1}{2}, & -1 < t < 1, \\ 1, & t \geq 1. \end{cases}$$

$$\frac{d}{dt}\text{sparsemax}(\mathbf{z})_2 = \begin{cases} 0, & t < -1, \\ \frac{1}{2}, & -1 < t < 1, \\ 0, & t > 1. \end{cases}$$

Relumax

Using the definition of relumax:

$$\text{relumax}_b(z)_i := \frac{\text{relu}\left(z_i - \max(z) + b\right)}{\sum_j \text{relu}\left(z_j - \max(z) + b\right)}$$

For the 2 classes we have:

$$u_1 = relu(z_1 - m + b) = relu(0 - m + b).$$

$$u_2 = relu(z_2 - m + b) = relu(t - m + b).$$

Which gives the following equation

$$relu\max b(z)_2 = \frac{u_2}{u_1 + u_2}$$

By solving the equation in for $t \leq 0$, $m = 0$ and $t \geq 0$, $m = t$ we obtain:

$$p_2(t) = \text{relumax}_b(z)_2 = \begin{cases} 0, & t \leq -b, \\ \dfrac{t+b}{2b+t}, & -b < t \leq 0, \\ \dfrac{b}{2b-t}, & 0 < t < b, \\ 1, & t \geq b. \end{cases}$$

$$p_2'(t) = \begin{cases} 0, & t \leq -b, \\ \dfrac{b}{(2b+t)^2}, & -b < t < 0, \\ \dfrac{b}{(2b-t)^2}, & 0 < t < b, \\ 0, & t \geq b. \end{cases}$$

4. Jacobian of ReLUMax (5 points)

To determine the Jacobian matrix of ReLUMax were $z \epsilon \mathbb{R}^K$ with $K = 2$, and all components of z are unique, so the maximizer is unique, we must first consider the post-normalized ReLU activation function

$$relu \max_b(z)_i = \frac{ReLU\left(z_i - \max_k(z_k) + b\right)}{\sum_{j=1}^{K} ReLU\left(z_j - \max_k(z_k) + b\right)}$$

To assist with the derivations of the function lets define some variables:

$$m \coloneqq \arg\max_k(z_k), M \coloneqq z_m, u_i \coloneqq z_i - M + b, r_i \coloneqq ReLU(u_i), Z \coloneqq \sum_{j=1}^{K} r_j$$

After defined the auxiliar variables we must now define the region where the output is different from 0

$$S \coloneqq \{i : u_i > 0.\}$$

With that it's now possible to rewrite the target function as

$$p_i \coloneqq relu \max_b(z)_i = \begin{Bmatrix} \frac{u_i}{Z} if i \in S \\ 0 \, f i \notin S \end{Bmatrix}$$

Now we are in condition of determine the entries of the Jacobian matrix $J_{ij} = \frac{\partial relu \max_b(z)_i}{\partial z_j}$

Case 1: If $j \notin S$

$$\frac{\partial p_i}{\partial z_j} = 0, \forall i$$

Case 2: If $j \in S, \; j \neq m$

$$\frac{\partial p_i}{\partial z_j} = \begin{cases} \frac{1 - p_j}{Z}, & i = j, \\ -\frac{p_i}{Z}, & i \in S, \; i \neq j, \\ 0, & i \notin S. \end{cases}$$

Case 3: If $j = m$

$$\frac{\partial p_i}{\partial z_m} = \begin{cases} p_m(|S| - 1), & i = m, \\ p_i(|S| - 1) - \frac{1}{Z}, & i \in S,\ i \neq m, \\ 0, & i \notin S. \end{cases}$$

With the general case already defined, we can now determine the specific Jacobian matrix for $K = 2$, $z = (z_1, z_2)$ with $z_1 \neq z_2$.

Case 1: Both coordinates give a ReLU output different from 0

Start by assuming $z_1 > z_2$, which gives us $m = 1$, $M = z_1$ and $u_1 = b > 0$, $u_2 = z_2 - z_1 + b$. If $u_2 > 0$, this means $z_1 - z_2 < b$, so both ReLUs have an output different from 0 which means $S = \{1,2\}$, $Z = u_1 + u_2 = 2b + z_1 - z_2$, $p_1 = \frac{u_1}{Z} = \frac{b}{Z}$, $p_2 = \frac{u_2}{Z} = 1 - p_1$.

With this is now possible to calculate the Jacobian matrix entries giving us the following matrix

$$J(z) = \frac{1}{Z}\begin{pmatrix} p_1 & -p_1 \\ p_2 - 1 & 1 - p_2 \end{pmatrix}$$

If we substitute, $p_2 = 1 - p_1$ we get the simplified matrix

$$J(z) = \frac{p_1}{Z}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}, \qquad p_1 = \frac{b}{2b + z_2 - z_1}.$$

If $z_2 > z_1$the matrix is the same, but with p_2 instead and $z_1 - z_2$ instead.

Case 2: Only the maximum z or none give an output different from 0

This case is possible when $z_1 - z_2 > b$, $u_2 < 0$ or vice versa which gave ReLU only positive in the maximum value giving us $J(z) = (0)_{2\cdot 2}$.

5. Modified Cross-Entropy Loss for ReLUMax (5 points)

When we are using the standard cross-entropy with softmax it isn't a problem because softmax is always positive, which makes the logarithm always defined and finite, with the loss function convex, on the other hand when softmax its substituted by relumax or sparsemax the loss functions becomes non convex, because both relumax and sparsemax allow for 0 values which would make the logarithm undefined .

The second part of the exercise asked to calculate the gradient of the loss function

$$L(z) = -\log \quad \frac{relu \max_b(z)_i + \in}{1 + K \in}$$

To help with the calculations let's define:

$$p(z) \coloneqq relu \max_b(z) \ \in \mathbb{R}^K,\ p_i \coloneqq relu \max_b(z_i)$$

With this we can now define the perturbed relumax probability:

$$\tilde{p}_i = \frac{\in + p_i(z)}{1 + K \in}$$

And the loss

$$L(z) = -\log\tilde{} p_i(z) = -\log\left(\frac{\in + p_i(z)}{1 + K \in}\right) = -\log\big(\in + p_i(z)\big) + \log(1 + K \in)$$

For the gradient we only need to consider the first term, $-\log\big(\in + p_i(z)\big)$, because the second term is constant, $\log(1 + K \in)$. We are now in condition to determine the gradient by calculating the derivative in function of z of the loss function, which gives us the following result

$$\nabla_z L = -\frac{1}{\in + p_i(z)} \cdot \frac{\partial p_i(z)}{\partial z_j} = -\frac{1}{\in + elu \max_b(z)_i} \cdot J_i(z)$$

Section 3 References:

In International conference on machine learning, pages 1614–1623. PMLR.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. Cambridge, MA: MIT Press.

Andersen, A. (2023). *Projection onto nonnegative orthant, rectangular box and polyhedron*. Version April 2, 2023. Available at https://angms.science/doc/CVX/Proj_nonnegBoxpoly.pdf