Exercise 1 Answers:

Theoretical Part

1.

a. Let $f(x)$ and $g(x)$ be linear functions:

$f(x) = Ax + b \quad g(x) = Mx + C$

$f(g(x)) = f(Mx + C) = A(Mx + C) + B$

$f(g(x)) = AMx + AC + B$

$\alpha = AM$

$\beta = AC + B$

$f(g(x)) = \alpha x + \beta$

which is a linear function.

b. Let $f(\vec{x}) = A\vec{x} + \overrightarrow{\alpha}$, $g(\vec{x}) = B\vec{x} + \vec{\beta}$ be affine transformations.

$f(g(\vec{x}) = A(B\vec{x} + \vec{\beta}) + \overrightarrow{\alpha}$

$f(g(\vec{x})) = AB\vec{x} + A\vec{\beta} + \overrightarrow{\alpha}$

Since A and B are invertibles, AB is invertible:

$(B^{-1}A^{-1})(AB) = B^{-1}(A^{-1}A)B = B^{-1} * I * B = B^{-1}B = I$

$AB * (B^{-1}A^{-1}) = I$

Therefore, we conclude that $f(g(\vec{x}))$ is an affine transformation.

2. Gradient descent:

a. Gradient descent update equations:

Minimizing x: $\frac{\partial f}{\partial x} 10(x-1)^2 + (y+1)^2/10 = 20(x-1) = 20x - 20$

Minimizing y: $\frac{\partial f}{\partial y} 10(x-1)^2 + (y+1)^2/10 = \frac{y+1}{5}$

b. A possible learning rate is achieved when |A| < 1 where $x^{k+1} = Ax^k + B$ for x,y. If $|A| \geq 1$ a global minimum will never be reached, as the function is divergent.

Therefore, for x the learning rates are:

$$x^{k+1} = x^k - \theta(\Delta_x f) = x^k - \theta(20x^k - 20)$$

$$x^{k+1} = x^k(1 - 20\theta) - 20\theta$$

$$1 - 20\theta < 1 \iff 20\theta > 0 \iff \theta > 0$$

$$20\theta - 1 < 1 \iff 20\theta < 2 \iff \theta < \frac{1}{10}$$

For y the learning rates are:

$$y^{k+1} = y^k - \theta(\Delta_y f) = y^k - \theta(y^k + 1)/5$$

$$y^{k+1} = y^k(1 - \frac{1}{5}\theta) - \frac{1}{5}\theta$$

$$1 - \frac{1}{5}\theta < 1 \iff \frac{1}{5}\theta > 0 \iff \theta > 0$$

$$\frac{1}{5}\theta - 1 < 1 \iff \frac{1}{5}\theta < 2 \iff \theta < 10$$

The maximal learning rate possible is $\frac{1}{10}$ and any learning rate above that will result with a gradient descent method that won't get to a converging solution.

c. The value in the minimization term that determines the maximal learning rate for a variable is the coefficient of that variable in the term.

For example, for variable x, the term $x^k - \theta(\mathbf{20}x^k - 20)$ indicates that the maximal learning rate is determined by the coefficient 20.

d. The variable that takes the longest to converge in the optimization process is the one with the higher maximal learning rate, as a higher rate typically requires adjustments to slow down convergence for stability. In this case, variable y takes longer to converge because it's maximal learning rate is higher than that of x. This is reflected in the term $y^k - \theta(\frac{\mathbf{1}}{\mathbf{5}}y^k - \frac{1}{5})$, where the coefficient $\frac{1}{5}$ is the determining factor for the maximal learning rate. Thus y's convergence is intentionally slowed down to allow x to converge as fast as possible.

3. We will define the loss as:

   Calculate circular loss(deg_{pred}, deg_{true}):

   $$loss = deg_{pred} - deg_{true}$$

   $$loss = (loss + 180)\ \%\ 360 - 180$$

   return $abs(loss)$

4.

   1.

   $$\frac{\partial}{\partial x} f(x+y, 2x, z) = \frac{\partial f}{\partial (x+y)} \cdot \frac{\partial (x+y)}{\partial x} + \frac{\partial f}{\partial (2x)} \cdot \frac{\partial (2x)}{\partial x} + \frac{\partial f}{\partial z} \cdot \frac{\partial z}{\partial x} = \frac{\partial f}{\partial (x+y)} + 2\frac{\partial f}{\partial (2x)}$$

   2. $f_1(f_2(\ldots.f_n(x)))) = f_1'(f_2(\ldots.f_n(x))) * f_2(f_3(\ldots.f_n(x)))'$

      Iteratively we calculate

      $f_2(f_3(\ldots.f_n(x)))' = f_2'(f_3(\ldots f_n(x)) * f_3(f_4(\ldots f_n(x)))'$

      $$f_1(f_2(\ldots.f_n(x)))) = \prod_{i=1}^{n} f_i'(f_{i+1}(\ldots f_n(x))$$

3. $f_1(x, f_2(x, f_3(\ldots f_{n-1}(x, f_n(x)))' = \frac{\partial f_1}{\partial x} + \frac{\partial f_1}{\partial f_2} * \frac{\partial f_2}{\partial x} + \frac{\partial f_1}{\partial f_2} * \frac{\partial f_2}{\partial f_3} * \frac{\partial f_3}{\partial x} \ldots + \frac{\partial f_1}{\partial f_2} * \ldots \frac{\partial f_n}{\partial x}$

$$= \sum_{i=1}^{n} \frac{\partial f_i}{\partial x} \prod_{j=1}^{i-1} \frac{\partial f_j}{\partial f_{j+1}}$$

4. $f(x + g(x + h(x)))' = f'(x + g(x + h(x))) * (x + g(x + h(x)))'$

   $(x + g(x + h(x)))' = 1 + g'(x + h(x)) * (1 + h'(x))$

   $f(x + g(x + h(x)))' = f'(x + g(x + h(x))) * (1 + g'(x + h(x)) * (1 + h'(x)))$

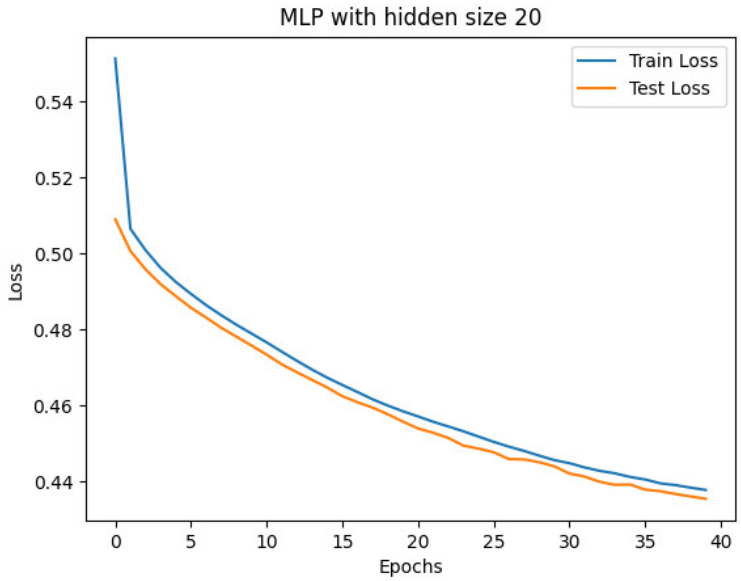
Programming Task

Question 2:

A. The 9-mers of amino acids will be encoded into a 20-dimensional vector, using one-hot encoding.
   We have counted each of the amino acids, added them all together and created this 20-dimensional vector.
   For example if the amino acids were : ARNEACEFM, the 20 dimensional vector will look like this: {2,1,1,0,1,2,0,…,0}, where the 20 dimensional vectors represent {Count of A, Count of R, …Count of Y, Count of V}.
   This will allow us to represent the amino acids in a way that the model can understand.

B. The network's input dimension under this representation will be 20, as each amino acid is represented by a 20-dimensional vector.
   This dimension pose a training problem, under-fitting in particular, as the size of the hidden layers is relatively small, which may result in a model

that is not able to learn the data well. We concluded that the network has faced an under-fitting problem as both the train and test losses are relatively high. If the problem was over-fitting, we should have seen a big difference between the train and test loss, which is not the case here.



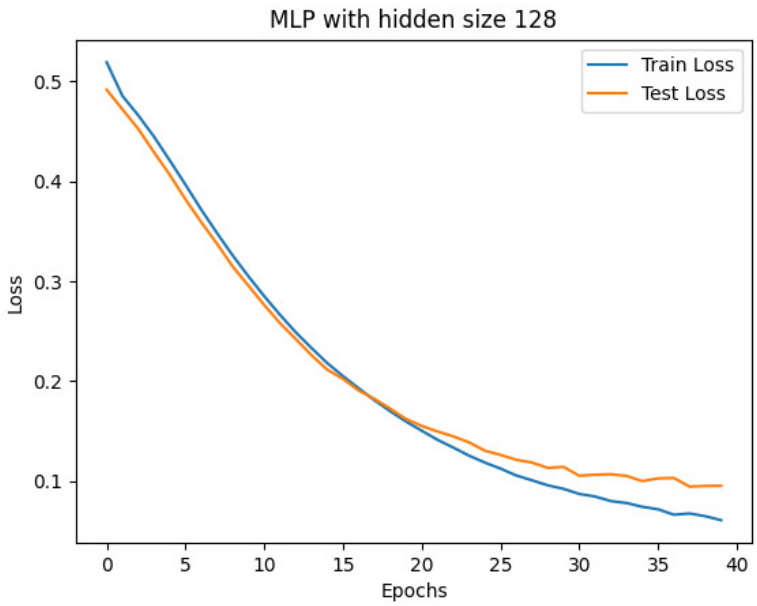
Plot 1 - MLP With hidden size of 20

Accuracy: 79.129

C. A network architecture that avoids the problem seen in B is to increase the size of the hidden layers.

Such network architecture will have a larger number of neurons in the hidden layers, which will allow the model to learn the data better.

For the comparison we have used a hidden layer size of 128.

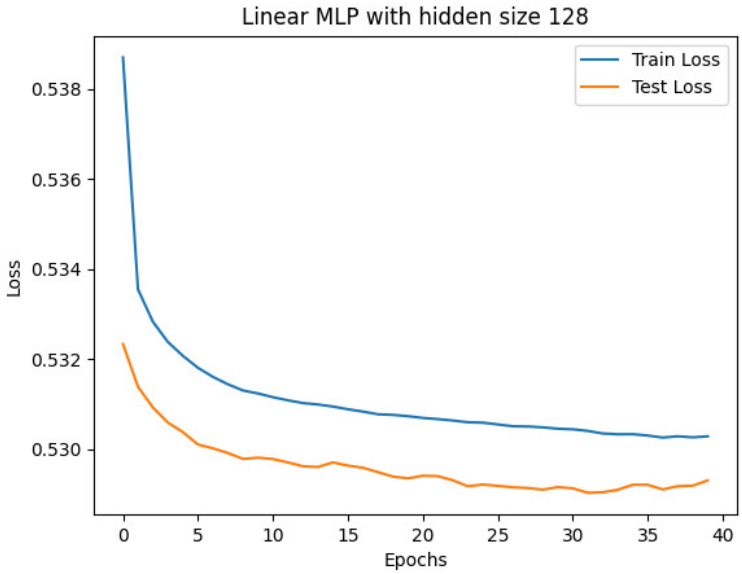


Plot 2 - MLP with hidden size of 128

Accuracy: 97.409

D. The results of the model without the non-linear activation function are worse than the model with the activation function. This is because the activation function allows the model to learn complex patterns in the data, which is not possible in a linear model.

The model with the activation function has a better performance than the linear model.



Plot 3 - MLP without non-linear operators

Accuracy: 73.620

E. The top 3 detectable peptides are :
[('SWMESEFRV', 1.0), ('PFAMQMAYR', 1.0), ('IPFAMQMAY', 0.9999971389770508)]