REPORT

Question 1:

- **Provide a normalization over the given data.**

  I have used **Min-Max** normalization technique over the given data set.

$$\forall x \frac{x - min(\forall x)}{max(\forall x) - min(\forall x)}$$

Question 2:

- **Implement the backpropagation algorithm to compute gradients. Run gradient descent on a cross-entropy loss function. Use exactly the following network configuration and report the accuracy on test.csv within report.pdf.**
  - **1 Hidden layer**
  - **100 Hidden units**
  - **0.001 Learining rate**
  - **No regularization**
  - **Batch size : 100**

  I have used the following configuration for the network (described below):

INPUT NEURON	37 (36 FEATURES + 1 BIAS)
HIDDEN LAYER	1
HIDDEN NEURONS	100
OUTPUT NEURONS	3 (ONE HOT ENCODING OF 3 CLASSES)
LEARNING RATE	0.001
REGULARIZATION	**NO**
ACTIVATION FUNCTION	SIGMOID
GRADIENT DESCENT VARIANT	MINI BATCH OF SIZE 100

Result: Test Set Mean F Score --> 0.51500 (Source : kaggle)

Question 3:

I. **Start with an initial learning rate and decay it with time by evaluating performance on a validation set (randomly sampled from the training data, constituting 10% of the original training set). Explain how you set your learning rate.**

For solving the problem of decaying of learning rate with time duration, I used the variation of time based decay method, instead of decaying the learning rate with number of iteration in the former method, I reduced it by the fraction of drop in the validation error (**cross emtropy loss**) from minimum validation error occurred till now.

NOTE : Learn rate is decayed only when a drop in validation errror is seen

min_loss = Minimum validation loss seen till current epochs
valid_loss = Current epoch validation loss

$$learn_{rate} = \frac{learn_{rate}}{(1+|(min_{loss} - valid_{loss})|)}$$

I started with initial **learn_rate = 0.1**

Configuration of the network on which the result is taken:

INPUT NEURON	37 (36 FEATURES + 1 BIAS)
HIDDEN LAYER	1
HIDDEN NEURONS	100
OUTPUT NEURONS	3 (ONE HOT ENCODING OF 3 CLASSES)
LEARNING RATE	0.1
REGULARIZATION	**YES** ($\lambda = 3$)
ACTIVATION FUNCTION	SIGMOID
GRADIENT DESCENT VARIANT	MINI BATCH OF SIZE 100
VALIDATION-SET	**10% OF TRAINING SET** (RANDOMLY TAKEN)

RESULT:

Train : Loss = 0.473803768 **Accuracy= 0.6658680**
Valid : Loss = 0.483203134 **Accuracy= 0.6546875**
Test Accuracy : 0.48600 **...(Source: Kaggle)**

II. **Vary the learning rate, number of hidden layers, number of nodes in each hidden layer and λ and record both the training/validation losses and training/validation accuracies. Include a table structured as follows within report.pdf that records these numbers. This table should have at least five rows and should include your most successful runs.**

S.No.	Learning Rate	Num/size of hidden layers	λ	Training loss	Training accuracy	Validation loss	Validation accuracy	Test F-Score (Kaggle)
1.	0.1	2(100,100)	4	0.43304071	0.705173611	0.43465259	0.70125	**0.50266**
2.	0.005	2(200,200)	4	0.39843171	0.726458333	0.41648905	0.711875	**0.48300**
3.	0.01	3(150,150,150)	15	0.44450174	0.706840277	0.46146342	0.6925	**0.53433**
4.	0.001	3(100,100,100)	10	0.46473313	0.6778125	0.48018130	0.659375	**0.48966**
5.	0.0	3(200,200,200)	2	0.41193493	0.70892361	0.42859600	0.6978125	**0.47833**
6.	0..01	4(100,100,100, 100,)	15	0.45600845	0.67857638	0.44829181	0.6859375	**0.52700**
7.	0.001	4(200,200,200, 200)	10	0.47738775	0.656319444	0.47311003	0.6671875	**0.51100**

Question 4:

- **To learn more about how activation functions influence performance, implement at least two different activation functions for the nodes in the hidden layers. Describe how they behave by reporting performance on test.csv within report.pdf with using these different activation functions.**

1. **Tanh:**

$$\tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

Tanh as a hyperbolic function is better than the sigmoid function, as the range of it is bigger than siomoid which is from [-1,1]. The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph.

The function is **differentiable.**

Its derivative:-

$$\tanh'(x) = (1 - \tanh^2(x))$$

2. ReLu:

$$Relu(x) = \begin{matrix} x = 0, if\ x < 0 \\ x = x, if\ x \geq 0 \end{matrix}$$

The ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.

The function is differentiable except at point 0.

Its derivative,

$$Relu'(x) = \begin{matrix} x = 0, if\ x < 0 \\ x = 1, if\ x \geq 0 \end{matrix}$$

3. Softplus:

$$Softplus(x) = \log(1 + e^{x})$$

Outputs produced by sigmoid and tanh functions have upper and lower limits whereas softplus function produces outputs in scale of (0, +∞). **That's the essental difference.**

The function is differentiable.

It's derivative,

$$Softplus'(x) = \frac{1}{1 + e^{-x}} = sigmoid(x)$$

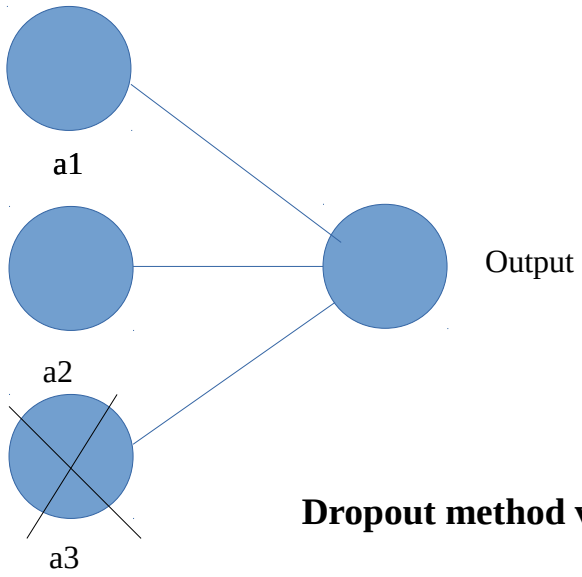
Result: Test Set Mean F Score is given along with accuracy on validation-set and training-set.

Activation Functiom	Learning Rate	Num/size of hidden layers	λ	Training loss	Training accuracy	Validation loss	Validation accuracy	Test F-Score (Kaggle)
TANH	0.001	3(100,100,100)	10	0.47513541	0.70302083	0.4928550	0.6846875	**0.50466**
	0.001	2(300,300)	10	0.62847272	0.69065979	0.6144809	0.6940625	**0.48900**
RELU	0.005	3(100,100,100)	4	0.40617853	0.72135416	0.4200231	0.710625	**0.46033**
	0.005	2(300,300)	4	0.40377603	0.72246527	0.4065594	0.7215632	**0.43633**
SOFTPLUS	0.005	3(100,100,100)	4	0.41564286	0.715	0.4206334	0.716875	**0.46600**
	0..005	2(300,300)	15	0.42695146	0.70291666	0.4073041	0.7215625	**0.48766**

Question 5:

- **Optimize the neural network to perform as well as possible on the classification task using any techniques you deem fit.**

DROPOUT METHOD :



Dropout method visualization

With dropout, we remove some neurons randomly. So there will be iterations in which neuron 3 of penultimate layer (say L-1) is removed viz., making a3 = 0, and therefore only the output of neuron 1 and neuron 3 will be forward. This will generate non-zero gradients for neuron 1 and neuron 2, and it will gradually start doing better than random guess. Then neuron 3 will also update its function to have a non-trivial linear combination of neurons 1 and 2.

Configuration of the network on which the result is taken:

INPUT NEURON	37 (36 FEATURES + 1 BIAS)
HIDDEN LAYER	3
HIDDEN NEURONS	200
OUTPUT NEURONS	3 (ONE HOT ENCODING OF 3 CLASSES)
LEARNING RATE	0.001
REGULARIZATION	**NO**
ACTIVATION FUNCTION	SIGMOID
GRADIENT DESCENT VARIANT	MINI BATCH OF SIZE 100
DROPOUT METHOD	0.02 (2% of neuron of each layer is dropped out)

RESULT : TEST SET MEAN F-SCORE**:->0.50900** **(SOURCE: Kaggle)**