ECS759P Artificial Intelligence: Coursework 2

Name : Allu Vadakemulanjanal Babu

Student Number : 220576404

Crystal clear (Logic problem)

Question 1

1. ∃ x (dog(x) ∧ Owns(You, x))
2. BuysCarrotsBushel(Robin)
3. ∀x (∃y (Owns(x, y) ∧ Rabbit(y)) →(∃z (∃w (Cat(w) ∧ Chases(z, w)))))
4. ∀ x Dog(x) → ∃y (Rabbit(y) ∧ Chases(x, y))
5. ∀ x (BuysCarrotsBushel(x) → ∃y (Owns(x, y) ∧ (Rabbit(y) ∨ Grocery(y))))
6. ∀ x ∀ y (∃z(Hates(x, z) ∧ (y, z)) → ¬ Dates(x, y))

Question 2

The sentences expressed in First Order Logic are translated into Conjunctive Normal Forms using the rules.

1. Dog(a)

   Owns(You, a)
2. BuysCarrotsBushel(Robin)
3. ¬ Rabbit(x1) ∨¬ Owns(x2, x1) ∨¬ Chase(x3, x4) ∨¬ Rabbit(x4) ∨ Hates(x2, x3)
4. ¬ Dog(x5) ∨ Rabbit(f1(x5))

   ¬ Dog(x5) ∨ Chases(x5, f1(x5))
5. ¬ BuysCarrotsBushel(x6) ∨ Owns(x6, f2(x6))

   ¬ BuysCarrotsBushel(x6) ∨ Rabbit(f2(x6)) ∨ Grocery(f2(x6))
6. ¬ Hates(x7, x9) ∨ ¬ Owns(x8, x9) ∨¬ Dates(x7, x8)

Question 3

Madam Irma’s conclusion “If the person you are looking for does not own a grocery, she will not date you.” is first translated into FOL.

¬∃ x (Grocery(x) ∧ Owns(Robin, x)) → ¬Dates(Robin, You)

The FOL expression is now converted to CNF in the steps below:

(¬ (¬∃x(Grocery(x) ∧ Owns(Rabbit, x)) → ¬ Dates(Robin, You)))	Negate before conversion
¬(¬¬∃ x (Grocery(x) ∧ Owns(Robin, x))) ∧ ¬¬Dates(Robin, You)	Rewrite double negation
∀x(¬(Grocery(x) ∧ Owns(Robin, x))) ∧ Dates(Robin, You)	Minimise negation
∀x(¬Grocery(x) ∨ ¬ Owns(Robin, x)) ∧ Dates(Robin, You)	De Morgan’s Law
¬Grocery(x) ∨ Owns(Robin, x) ∧ Dates(Robin, You)	Skolemise

¬Grocery(x) ∨ Owns(Robin, x)

Date(Robin, You)

Question 4

The expressions obtained in question 2 are added to Knowledge Base. With Knowledge base and the expression obtained in question 3, we are trying to prove that Dates(Robin, You) is true. We are trying to prove it by proof by contradiction. Initially we assume that the expression we are trying to prove is not true. With the help of the Knowledge Base we then prove that the assumption is not possible.

Note : Unifiers are written in green

Date(Robin, You)

¬ Hates(x7, x9) ∨ ¬ Owns(x8, x9) ∨¬ Dates(x7, x8)

{Robin/ x7, you/x8}

Result : ¬ Hates(Robin, x9) ∨¬ Own(You, x9) -(1)

¬ Hates(Robin, x9) ∨¬ Own(You, x9)

¬ Rabbit(x1) ∨¬ Owns(x2, x1) ∨ Chases(x3, x4) ∨¬ Rabbit(x4) ∨Hates(x2, x3)

{Robin/x2, x9/x3}

Result : ¬ Rabbit(x1) ∨ ¬ Owns(Robin, x1) ∨¬ Chases(x9, x4) ∨¬ Rabbit(x4) ∨¬ Owns(You, x9)-(2)

¬ Rabbit(x1) ∨ ¬ Owns(Robin, x1) ∨¬ Chases(x9, x4) ∨¬ Rabbit(x4) ∨¬
Owns(You, x9)

Owns(You, a)

{a/x9}

Result : ¬ Rabbit(x1) ∨ ¬ Owns(Robin, x1) ∨¬ Chases(a, x4) ∨¬ Rabbit(x4) -(3)

¬ Rabbit(x1) ∨ ¬ Owns(Robin, x1) ∨¬ Chases(a, x4) ∨¬ Rabbit(x4)

¬ Dog(x5) ∨ Chases(x5, f1(x5))

{a/x5, f1(a)/x4}

Result : ¬ Rabbit(x1) ∨ ¬ Owns(Robin, x1) ∨¬ Rabbit(f1(a)) ∨¬ Dog(a) -(4)

¬ Rabbit(x1) ∨ ¬ Owns(Robin, x1) ∨¬ Rabbit(f1(a)) ∨¬ Dog(a) Dog(a)

Result : ¬ Rabbit(x1) ∨ ¬ Owns(Robin, x1) ∨¬ Rabbit(f1(a))-(5)

¬ Rabbit(x1) ∨ ¬ Owns(Robin, x1) ∨¬ Rabbit(f1(a))

¬ BuysCarrotsBushels(x6) ∨ Rabbit(f2(x6)) ∨ Grocery(f2(x6))

{f2(x6)/x1}

Result : ¬ Owns(Robin, f2(x6))∨ Rabbit(f1(a)) ∨ BuysCarrotsBushels(x6) ∨ Grocery(f2(x6)) -(6)

Owns(Robin, f2(x6))∨ Rabbit(f1(a)) ∨ BuysCarrotsBushels(x6) ∨ Grocery(f2(x6))

BuysCarrotsBushels(Robin)

{Robin/x6}

Result : ¬ Owns(Robin, f2(Robin)) ∨ Rabbit(f1(a)) ∨ Grocery(f2(Robin)) -(7)

¬ Owns(Robin, f2(Robin)) ∨ Rabbit(f1(a)) ∨ Grocery(f2(Robin))

¬ Dog(x5) ∨ Rabbit(f1(x5))

{a/x5}

Result : ¬ Owns(Robin, f2(Robin)) ∨ Grocery(f2(Robin)) ∨¬ Dog(a) -(8)

¬ Owns(Robin, f2(Robin)) ∨ Grocery(f2(Robin)) ∨¬ Dog(a) Dog(a)

Result : ¬ Owns(Robin, f2(Robin)) ∨ Grocery(f2(Robin)) -(9)

¬ Owns(Robin, f2(Robin)) ∨ Grocery(f2(Robin))

¬ BuysCarrotsBushels(x6) ∨ Owns(x6, f2(x6))

{Robin/x6}

Result : Grocery(f2(Robin)) ∨¬ BuysCarrotsBushels(Robin) -(10)

Grocery(f2(Robin)) ∨¬ BuysCarrotsBushels(Robin)

BuysCarrotsBushels(Robin)

Result : Grocery(f2(Robin)) -(11)

Grocery(f2(Robin))

¬ Grocery(x10) ∨¬ Owns(Robin, x10)

{f2(Robin)/x10}

Result : ¬ Owns(Robin, f2(Robin)) -(12)

¬ Owns(Robin, f2(Robin))

¬ BuysCarrotsBushels(x6) ∨ Owns(x6, f2(x6))

{Robin/x6}

Result : ¬ BuysCarrotsBushels(Robin) -(13)

¬ BuysCarrotsBushels(Robin)

BuysCarrotBushels(Robin)

{Robin/x6}

Result : {}

The statement 13 contradicts the CNF statement BusCarrotsBushels(Robin). This shows that Madam Irma was right.

Lost in Closet(Classification)

Question 1

The loss function depends on the predictions that our model is trained upon. We have a multi class classification problem here. Therefore, Categorical Cross Entropy Loss is the most appropriate loss function.

$$L_{CE} - \sum_{i=1}^{n} t_i log(p_i)$$

where

t_i = Ground truth score for each class i

p_i=Softmax probability for the i th class

Cross Entropy Loss function for multi class classification calculates the average of the differences between the ground truth and the predicted probability distributions.

Question 2

The train and test accuracy obtained after 30 epochs are 99.21 and 90.84.

Plot of the train and test accuracy per epochs is given in the Figure 1. Both the training and test accuracy increases with epochs. But after a few epochs the increase in test accuracy is less than the increase in the training accuracy. Training accuracy is higher than test accuracy. During training we fit the data with the expected output. However during testing the model developed during training is applied to the test data. Therefore it makes sense that the training accuracy is higher than the test accuracy. The difference between training and test accuracies is around 8 percent.
Therefore the model created is a good fit for the data.

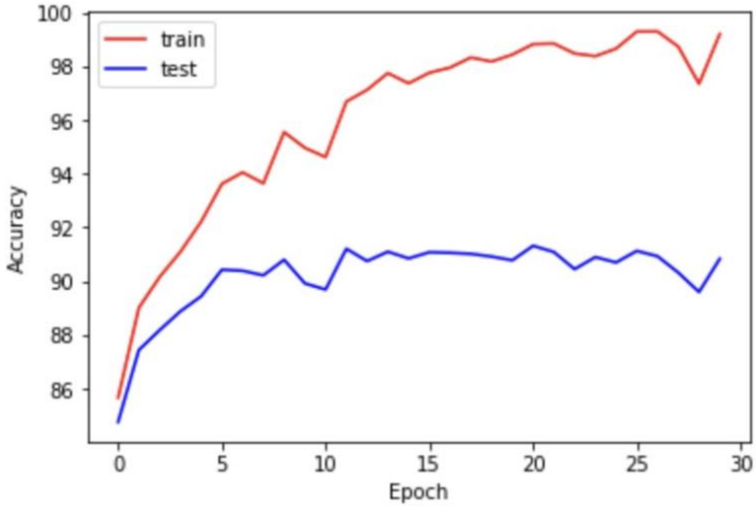

Figure 1 : Plot of the accuracy of training and test sets per epochs for ReLU activation function with learning rate = 0.1

Plot of the accuracy of the train loss per epochs is given in Figure 2. Loss function decreases with each epochs. For the first few epochs, the loss reduces drastically then it reduces gradually.

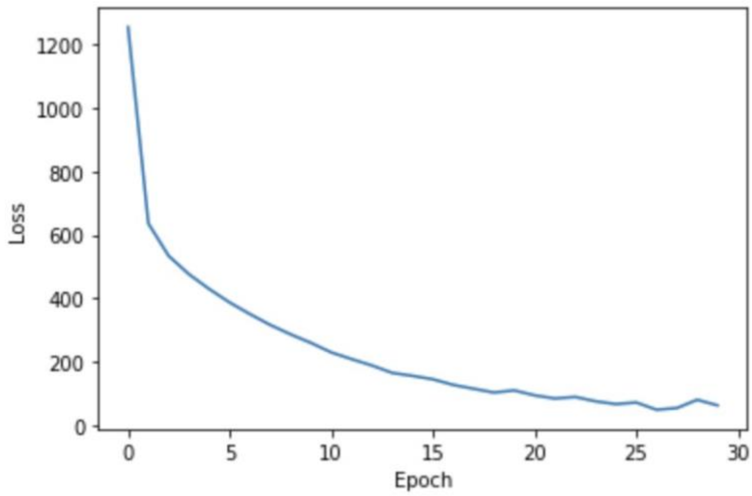

Figure 2 : Plot of the training loss per epochs for ReLU activation function with learning rate = 0.1

Question 3

The training and test accuracies obtained with Tanh, Sigmoid and ELU activation functions are given in the Table 1.

Activation Function	Train Accuracy	Test Accuracy
Tanh	100	91.67
Sigmoid	89.99	88.23
ELU	99.3	98.41

Table 1: Train and Test accuracies of the model with different activation functions for learning rate = 0.1

Higher Noise is observed when ELU activation function was used in training and testing of the model

Training Accuracy with Tanh activation function was found to be the highest. Since the derivative of the Tanh function is a higher value, gradient descent takes a larger step in each epoch and loss function converges faster. So the training accuracy of Tanh function being higher than other activation function is as expected. The derivative of the ELU activation function has an extra alpha as compared to ReLU activation function so ELU function takes larger step in gradient descent and converges faster. Thus for the same epochs accuracy of the model with ELU activation function is higher than ReLU activation function. The derivative of the sigmoid activation function is smaller therefore it will take smaller steps in gradient descent and the loss converges to minimum slowly. Therefore for the same epoch compare to other activation functions the accuracy of the sigmoid activation function will be the smallest.

The plots for training and test accuracies with epochs and the plot of the training loss with epochs for different activations functions is given below:

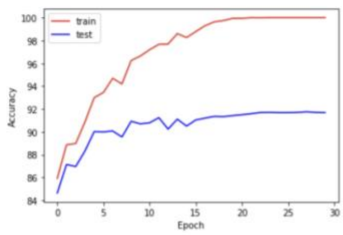

Figure 3 : Plot of the accuracy of training and test sets per epochs for Tanh activation function with learning rate = 0.1

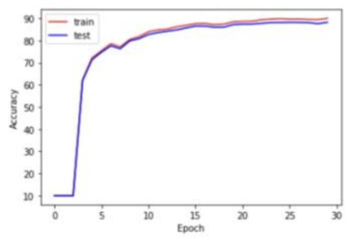

Figure 5 : Plot of the accuracy of training and test sets per epochs for Sigmoid activation function with learning rate = 0.1

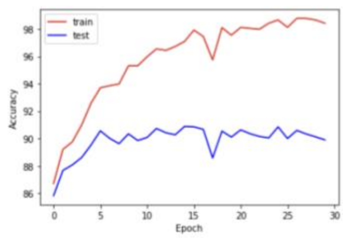

Figure 7 : Plot of the accuracy of training and test sets per epochs for ELU activation function with learning rate = 0.1

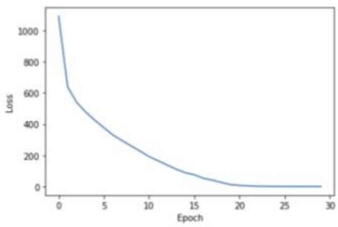

Figure 4 : Plot of the training loss per epochs for Tanh activation function with learning rate = 0.1

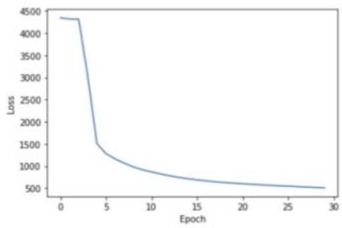

Figure 4 : Plot of the training loss per epochs for Sigmoid activation function with learning rate = 0.1

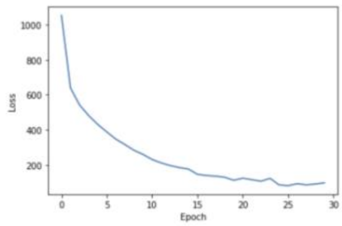

Figure 8 : Plot of the training loss per epochs for ELU activation function with learning rate = 0.1

<u>Question 4</u>

The training and test accuracies obtained with different learning rates are given in the Table 2.

Learning Rate	Train Accuracy	Test Accuracy
0.001	88.03	86.83
0.1	99.21	90.84
0.5	10.0	10.0
1	10.0	10.0
10	10.0	10.0

Table 2: Train and Test accuracies of the model with different learning rates for ReLU activation function

With learning rate 0.001 the step taken by gradient descent in each iteration is very less and the classifier does not converge to the minimum value. The result of the model is not improving with epochs. Therefore the accuracy of the model is very less. When the learning rate ≥0.5, gradient descent take very large step with each iteration and the loss instead of converging diverges and the accuracy of the model is very less. With learning rate 0.1 the model is converging and the step taken by gradient descent in each iteration is neither too small not too large. Therefore the accuracy of model with learning rate = 0.1 is the highest.

The plots for training and test accuracies with epochs and the plot of the training loss with epochs for different learning rate is given below:

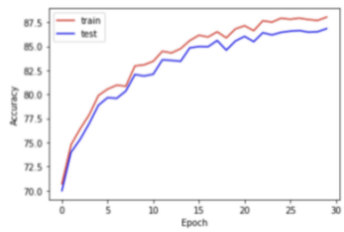

Figure 9 : Plot of the accuracy of training and test sets per epochs for ReLu activation function with learning rate = 0.001

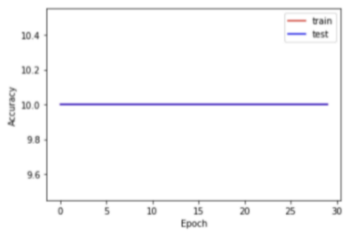

Figure 11 : : Plot of the accuracy of training and test sets per epochs for ReLu activation function with learning rate=0.5

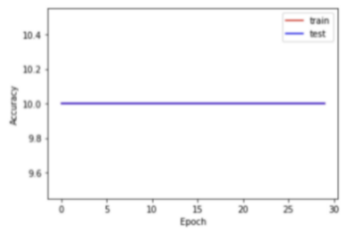

Figure 12 : Plot of the accuracy of training and test sets per epochs for ReLu activation function with learning rate = 1

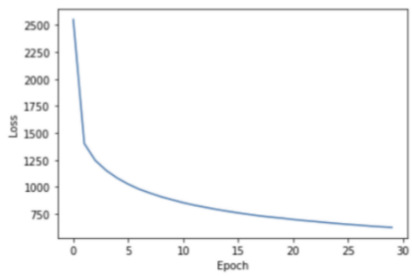

Figure 10 : Plot of the training loss per epochs for ReLu activation function with learning rate = 0.001

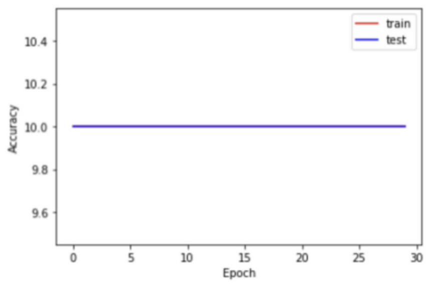

Figure 13 : Plot of the accuracy of training and test sets per epochs for ReLu activation function with learning rate = 10

Question 5

The train and test accuracy of the model with ReLU layer and learning rate = 0.1 with Dropout is 98.05 and 90.04.

When a dropout layer is added to the second fully connected layer during training some of the neurons of the second fully connected layer will be deactivated. Since some neurons will be deactivated each neuron will learn without depending on other neurons. This will improve the generalisation of the model and thus helps reduce overfitting. Comparison of Figure 1 and Figure 14 shows that the distance between test and train accuracy of the model with dropout is less than the model without dropout. The plot of the accuracy of the training and test set per epochs and the plot of the training loss per epochs is given below:

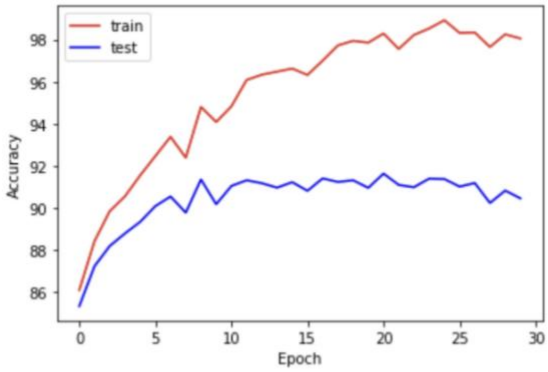

Figure 14 : Plot of the accuracy of training and test sets per epochs for ReLu activation function with learning rate = 0.1 and Dropout

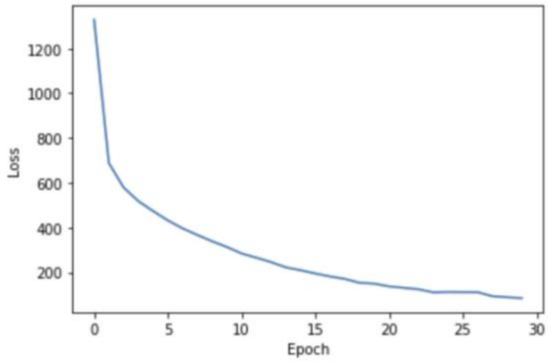

Figure 15 : Plot of the training loss per epochs for ReLu activation function with learning rate = 0.1 and Dropout