Artificial Intelligence Assignment – 2

Palak Dhanadia (210748141)

School of Electronic Engineering and Computer Science, Queen Mary University of London, UK

ec21276@qmul.ac.uk

1 Crystal Clear!

Question 1.

A.

(a) First Order Logic (FOL)

(i) $\exists x\,(dog(x) \wedge owns(you, x))$

(ii) $buy(Robin)$

(iii) $\forall x\, \forall y\,(owns(x, y) \wedge rabbit(y) \rightarrow \forall z\, \forall w\,(rabbit(w) \wedge chases(z, w) \rightarrow hates(x, z)))$

(iv) $\forall x\,(dog(x) \rightarrow \exists y\,(rabbit(y) \wedge chases(x, y)))$

(v) $\forall x\,(buy(x) \rightarrow \exists y\,(owns(x, y) \wedge (rabbit(y) \vee grocery(y))))$

(vi) $\forall x\, \forall y\, \forall z\,(owns(y, z) \wedge hates(x, z) \rightarrow \neg date(x, y))$

B.

(b) Conjunctive Normal Form (CNF)

(i) $owns(you, C)$

(ii) $buys(Robin)$

(iii) $\neg owns(x_1, x_2) \vee \neg rabbit(x_3) \vee \neg chases(x_4, x_3) \vee hates(x_1, x_4)$

(iv) $\neg dog(b) \vee rabbit(F(b))$

(v) $\neg dog(b) \vee chases(b, F(b))$

(vi) $\neg buys(z) \vee owns(z, G(z))$

(vii) $\neg buys(z) \vee rabbit(G(z)) \vee grocery(G(z))$

(viii) $\neg owns(a_1, a_2) \vee \neg hates(a_3, a_2) \vee \neg date(a_3, a_1)$

(ix) $dog(C)$

{ Here the constants are C, you & Robin. F(b) & G(z) are functions. }

C.

(c) Conclusion into FOL, negate it & convert it to a CNF.

FOL:

$\neg(\exists x \; grocery(x) \wedge owns(Robin, x)) \rightarrow \neg date(Robin, you)$

Negate & convert to CNF:

$\neg(\exists x \; grocery(x) \wedge owns(Robin, x)) \wedge \neg . \neg \; date(Robin, you)$

as $\neg(P \rightarrow Q) = P \wedge \neg Q$

Conclusion:

(i) $\neg grocery(y) \vee \neg owns(Robin, y)$

(ii) $date(Robin, you)$

D.

The resolution has been written in this form (P v Q) ^ (R v not(Q)) −> (P v R). The unifiers are written.

(d) To prove: Madame Irma is right & that you should go to see Robin to declare to her your (logic) love.

$\{date(Robin, you)\} \wedge \{\neg owns(a_1, a_2) \vee \neg hates(a_3, a_2) \vee \neg date(a_3, a_1)\} \rightarrow \{\neg owns(you, a_2) \vee hates(Robin, a_2)\}$

$\{Robin/a_3\}$ & $\{you/a_1\}$

$\{\neg owns(you, a_2) \vee \neg hates(Robin, a_2)\} \wedge \{\neg owns(x_1, x_2) \vee \neg rabbit(x_2) \vee \neg rabbit(x_3) \neg chases(x_4, x_3) \vee hates(x_1, x_4)\} \rightarrow \{\neg owns(you, a_2) \vee \neg owns(Robin, x_2) \vee \neg rabbit(x_2) \vee \neg rabbit(x_3) \vee \neg chases(a_2, x_3)\}$

$\{Robin/x_1\}$ & $\{a_2/x_4\}$

$\{\neg owns(you, a_2) \vee \neg owns(Robin, x_2) \vee \neg rabbit(x_2) \vee \neg rabbit(x_3) \vee \neg chases(a_2, x_3)\} \wedge \{owns(you, C)\} \rightarrow \{\neg owns(Robin, x_2) \vee \neg rabbit(x_2) \vee \neg rabbit(x_3) \vee \neg chases(C, x_3)\}$

$\{C/a_2\}$

$\{\neg owns(Robin, x_2) \vee \neg rabbit(x_2) \vee \neg rabbit(x_3) \vee \neg chases(C, x_3)\} \wedge \{\neg dog(b) \vee chases(b, F(b))\} \rightarrow \{\neg owns(Robin, x_2) \vee \neg rabbit(x_2) \vee \neg rabbit(F(C)) \vee \neg dog(C)\}$

$\{C/b\}$ & $\{F(C)/x_3\}$

$\{\neg owns(Robin, x_2) \vee \neg rabbit(x_2) \vee \neg rabbit(F(c)) \vee \neg dog(c)\} \wedge \{\neg dog(b) \vee rabbit(F(b))\} \rightarrow \{\neg owns(Robin, x_2) \vee \neg rabbit(x_2) \vee \neg dog(c)\}$

$\{c/b\}$

$\{\neg owns(Robin, x_2) \vee \neg rabbit(x_2) \vee \neg dog(c)\} \wedge \{dog(c)\} \rightarrow \{\neg owns(Robin, x_2) \vee \neg rabbit(x_2)\}$

$\{\neg owns(Robin, x_2) \vee \neg rabbit(x_2)\} \wedge \{\neg buy(z) \vee rabbit(G(z)) \vee grocery(G(z))\} \rightarrow \{\neg owns(Robin, G(z)) \vee \neg buy(z) \vee grocery(G(z))\}$

$\{G(z)/x_2\}$

$\{\neg owns(Robin, G(z)) \vee \neg buy(z) \vee grocery(G(z))\} \wedge \{\neg grocery(y) \vee \neg owns(Robin, y)\} \rightarrow \{\neg buy(z) \vee \neg owns(Robin, G(z))\}$

$\{G(z)/y\}$

$\{\neg buy(z) \vee \neg owns(Robin, G(z))\} \wedge \{\neg buy(z) \vee owns(z, G(z))\} \rightarrow \{\neg buy(Robin)\}$

$\{Robin/z\}$

$\{\neg buy(Robin)\} \wedge \{buy(Robin)\} \rightarrow \{\}$

Hence Proved

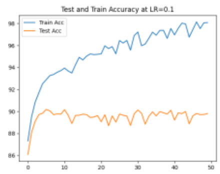
2 Lost in the Closet (Classification)

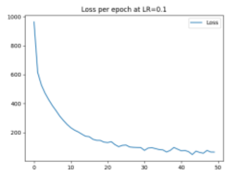
Question 2.

a) Because the task involves multiclass classification, the best loss function to utilise is categorical cross entropy, which is defined as $H(y, p) = -\Sigma_i y_i \log(p_i)$. This loss is combined with the output of the activation function of the last layer (predicted y) and the actual y. If the discrepancy between the projected and the real value is large, the loss will be significant. The loss function “CrossEntropyLoss()” is used in pytorch which already uses softmax internally. As a result, the final layer does not require an activation function.

b) The final stats (train and test) and graph are shown below:

Epoch [50/50], Loss: 65.2419, Train Accuracy: 98.08%, Test Accuracy: 89.81%





As the training and test accuracy increases the loss for the training reduces over the epochs. The loss reduces drastically for the first few epochs and then it reduces slowly over the next epochs. The training accuracy is 98% while the test accuracy is 89% approx.. This suggests overfitting of the model on the dataset.

c) Changing the activation function to tanh, Sigmoid and ELU:

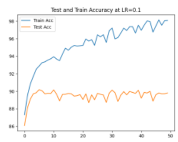
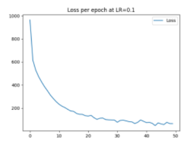
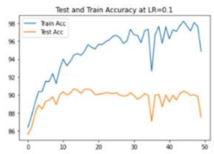
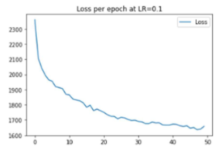
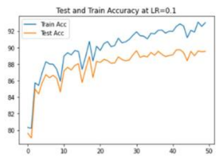
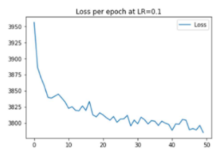
Activation Function	**Train Accuracy %**	**Test Accuracy %**	**Loss**
Tanh	99.92	90.80	0.8118
Sigmoid	93.79	90.24	292.1004
ELU	98.15	89.92	64.8841

From the observation we can see that tanh activation loss is very less as it is a steep activation function.

Learning Rate	**Train Accuracy %**	**Test Accuracy %**	**Loss**
0.001	88.82	87.75	599.9485
0.1	98.08	89.81	65.2419
0.5	89.68	84.92	648.4828
1	10	10.00	4330.5620
10	10	10.00	4636.2255

The loss for LR = 0.001 is high so it takes a lot more time for the model to converge. Higher number of epochs are required with very small learning rate. When the LR is 0.1, the train and test accuracy is at highest from the other LR and the loss is about 52. When LR is high the model starts to overshoot and can miss the optimal weights which results in increases the loss. Hence we see that, large LR results in unstable training and very small LR results in very slow training.

d)

Dropout	**Train Acc**	**Test Acc**	**Loss**	**Acc Graph**	**Loss Graph**
0.3	98.08	89.81	65.2419		
0.5	94.82	77.59	1657.6243		
0.9	93.16	89.25	3765.6582		

Adding dropouts is a regularisation approach that helps reduce the model's overfitting to the training set. As demonstrated above, it also raises the model's loss on the training set. While exercising, the dropout is used to freeze the weights or to let the knots fall. A higher abandon rate increases the likelihood of the node freezing. The value of the weight is 0 in a frozen node. The distance between the tensile accuracy (blue line) and the test accuracy (orange line) diminishes, as indicated in the graph above, and the tensile accuracy becomes comparable to the test accuracy. As a result, the model's overfitting is limited. A low dropout rate is consistent with the model, while a high dropout rate does not sufficiently adjust the model.

References

1) Convolutional Neural Networks (CNN) - Deep Learning Wizard