ECS759P Artificial Intelligence: Coursework 2

Razhan Hameed 210667222

December 17, 2021

Contents

1 Logic Problem

1.1 Express Madame Irma's six statements into First Order Logic (FOL).

You have a dog.

$$\exists x(DOG(x) \land OWN(You, x))$$

Robin buys carrots by the bushel.

$$Buy(Robin)$$

Anyone who owns a rabbit hates anything that chases any rabbit.

$$\forall x \forall y(OWN(x, y) \land RABBIT(y) \to \forall z \forall w(RABBIT(w) \land CHASE(z, w) \to HATES(x, z)))$$

Every dog chases some rabbit.

$$\forall x(DOG(x) \to \exists y(RABBIT(y) \land CHASE(x, y)))$$

Anyone who buys carrots by the bushel owns either a rabbit or a grocery store.

$$\forall x(BUY(x) \to \exists y(OWN(x, y) \land (RABBIT(y) \lor GROCERY(y))))$$

Someone who hates something owned by another person will not date that person.

$$\forall x \forall y \forall z(OWN(y, z) \land HATES(x, z) \to \neg DATE(x, y))$$

1.2 Translate the obtained expressions to Conjunctive Normal Forms (CNFs)

1. $\exists x(DOG(x) \land OWN(You, x))$
   - $(DOG(D) \land OWN(You, D))$
     Note: No more changes applicable

$$(DOG(D) \tag{1}$$

$$OWN(You, D)) \tag{2}$$

2. Buy(Robin)
   - CNF:

$$Buy(Robin) \tag{3}$$

3. $\forall x \forall y(OWN(x, y) \land RABBIT(y) \to \forall z \forall w(RABBIT(w) \land CHASE(z, w) \to HATES(x, z)))$
   - $\forall x \forall y(\neg(OWN(x, y) \land RABBIT(y)) \lor \forall z \forall w(\neg(RABBIT(w) \land CHASE(z, w)) \lor HATES(x, z)))$
   - $\forall x \forall y(\neg OWN(x, y) \lor \neg RABBIT(y)) \lor \forall z \forall w(\neg RABBIT(w) \lor \neg CHASE(z, w)) \lor HATES(x, z)))$
   - $\forall x \forall y(\neg OWN(x, y) \lor \neg RABBIT(y)) \lor \forall z \forall w(\neg RABBIT(w) \lor \neg CHASE(z, w)) \lor HATES(x, z)))$
   - CNF:

$$(\neg OWN(x, y) \lor \neg RABBIT(y)) \lor (\neg RABBIT(w) \lor \neg CHASE(z, w)) \lor HATES(x, z))) \tag{4}$$

4. $\forall x(DOG(x) \to \exists y(RABBIT(y) \land CHASE(x, y)))$
   - $\forall x(\neg DOG(x) \lor \exists y(RABBIT(y) \land CHASE(x, y)))$
   - $\forall x(\neg DOG(x) \lor \exists y(RABBIT(y) \land CHASE(x, y)))$
   - $\neg DOG(x) \lor (RABBIT(R) \land CHASE(x, R))$
   - $(\neg DOG(x) \lor (RABBIT(R)) \land (\neg DOG(x) \lor CHASE(x, R))$
     Note: There are two clauses

$$(\neg DOG(x) \lor (RABBIT(R)) \tag{5}$$

$$(\neg DOG(x) \lor CHASE(x, R)) \tag{6}$$

5. $\forall x(BUY(x) \to \exists y(OWN(x, y) \land (RABBIT(y) \lor GROCERY(y))))$
   - $\forall x(\neg BUY(x) \lor \exists y(OWN(x, y) \land (RABBIT(y) \lor GROCERY(y))))$
   - $\neg BUY(x) \lor (OWN(x, f(x)) \land (RABBIT(A) \lor GROCERY(A)))$
   - CNF:

$$\neg BUY(x) \lor (OWN(x, f(x)) \land RABBIT(f(x))) \lor (OWN(x, f(x)) \land GROCERY(f(x))) \tag{7}$$

6. $\forall x \forall y \forall z(OWN(y, z) \land HATES(x, z) \to \neg DATE(x, y))$
   - $\forall x \forall y \forall z(\neg(OWN(y, z) \land HATES(x, z)) \lor \neg DATE(x, y))$
   - $\forall x \forall y \forall z(\neg OWN(y, z) \lor \neg HATES(x, z)) \lor \neg DATE(x, y))$
   - CNF:

$$\neg OWN(y, z) \lor \neg HATES(x, z)) \lor \neg DATE(x, y) \tag{8}$$

1.3 Transform Madame Irma's conclusion into FOL, negate it and convert it to a CNF.

If the person you are looking for does not own a grocery, she will not date you.
First-Order Logic

$$((\neg\exists x(GROCERY(x) \wedge OWN(Robin, x))) \rightarrow \neg DATE(Robin, You))$$

Negation:

$$\neg[((\neg\exists x(GROCERY(x) \wedge OWN(Robin, x))) \rightarrow \neg DATE(Robin, You))]$$

CNF Conversion:

- $\neg[((\neg\exists x(\neg GROCERY(x) \vee \neg OWN(Robin, x))) \vee \neg DATE(Robin, You))]$
- $[((\neg\neg\exists x(\neg GROCERY(x) \vee \neg OWN(Robin, x))) \wedge \neg\neg DATE(Robin, You))]$
- $(\neg GROCERY(G) \vee \neg OWN(Robin, G))) \wedge DATE(Robin, You)$

$$(\neg GROCERY(G) \vee \neg OWN(Robin, G)) \tag{9}$$

$$DATE(Robin, You) \tag{10}$$

1.4 Based on all the previously created CNF (you should have at least 7 de- pending on how you split them), prove that Madame Irma is right and that you should go to see Robin to declare to her your (logic) love.

Proof by contradiction:

- Resolve 3, 7 with unifier (Robin/x)

$$(OWN(Robin, f(Robin)) \wedge RABBIT(f(Robin))) \vee (OWN(Robin, f(Robin)) \wedge GROCERY(f(Robin))) \tag{11}$$

- Rewrite 9

$$\neg(GROCERY(G) \wedge OWN(Robin, G))) \tag{12}$$

- Resolve 11, 12 with unifier (f(Robin)/G):

$$(OWN(Robin, f(Robin)) \wedge RABBIT(f(Robin))) \tag{13}$$

- Resolve 4, 13 with unifier (Robin/x) and (f(Robin)/y):

$$(\neg RABBIT(w) \vee \neg CHASE(z, w)) \vee HATES(Robin, z))) \tag{14}$$

- Resolve 1, 5 with unifier (D/x):

$$RABBIT(R) \tag{15}$$

- Resolve 1, 6 with unifier (D/x):

$$CHASE(D, R) \tag{16}$$

- Resolve 1, 15, and 16 with unifier (R/w) and (D/z):

$$HATES(Robin, D) \tag{17}$$

- Resolve 8, 17 with unifier (Robin/x) and (D/z):

$$\neg OWN(y, D) \vee \neg DATE(Robin, y) \tag{18}$$

- Resolve 10, 18 , with unifier (You/y):

$$\neg OWN(You, D) \tag{19}$$

- Resolve 2, 19 :

$$\perp \qquad\qquad False \tag{20}$$

We have reached a contradiction which says I do not own a dog which from the first premise we know I own a dog, there we completed the proof of our goal, hence I can go to robin and prove her my logic/love.

2 Classification

2.1 Given the problem, what is the most appropriate loss function to use?

The dataset consists of 10 classes which mean it is a multi-class classification problem. Therefore we decided not to use loss functions best suited for regression problems such as Mean-Squared Error. There are a few suitable loss functions for the given problem:

- Kullback Leibler Divergence Loss
- Negative Log-Likelihood
- Cross-entropy

The KL divergence is used when using models that learn to estimate a more complex function than just a multi-class classification problem. For this problem, the model can be interpreted as predicting integer values 0, 1 .. 9 each class is assigned a unique integer value. Cross-entropy is suited best for this type of problem because it's probabilistic since we have ten classes. We need the probability for each class to measure the error between two probability distributions; In our case, the ground truth and the predicted probability distribution.

2.2 Create and train a Convolutional Neural Network

Based on the given configuration: SGD with learning rate of 0.1, weight initialization using the Xavier initialisation and ReLU as an activation function, after 50 epochs we get the following values:
Train accuracy: 98.91 and **Test accuracy**: 90.6

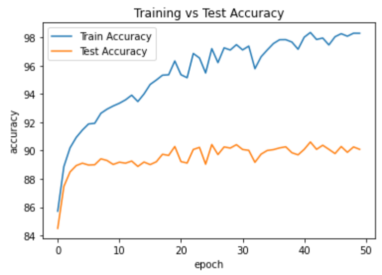

Figure 1: Training and testing accuracy with 0.1 learning rate

When the first epoch is completed, the loss is 988. However, after the network runs for 50 epochs, the loss decreases to 26. The network fits the data perfectly based on the returned loss, but 50 epochs may be too much since it oscillates between 20 and 30 for the last five epochs. In other words, the model over trains converges before finishing all the 50 epochs. We can observe that the test accuracy stabilizes over the 50 epochs based on the plot. The gap between test and training accuracy increases, and thus we can state the model is overfit.

Note: *Some of the code (i.e. training loop and evaluation is adopted from the lab solutions.*

2.3 Change the activation function to Tanh, Sigmoid and ELU. Provide only the final classification accuracy. Keeping ReLU, use 5 different learning rates: 0.001, 0.1, 0.5, 1, 10. What do you observe? Explain.

I built a helper function *build˜network()* to be able to use the same network architecture with different activation functions. The function takes one argument: the activation function name passes it to the model class, applies the Xavier initialization weights, and returns the network.

The table below shows loss value, training accuracy, and test accuracy for different activation functions:

Loss function	Loss	Training Accuracy	Test Accuracy
Tanh	0.29	100.0	91.4
Sigmoid	389.43	92.46	90.28
ELU	90.66	96.658	89.58

The model is obviously overfitting with Tanh. Sigmoid is good in terms of training accuracy, but the loss is noticeably large. The test accuracy is not as high as the other two with ELU. The below table contains experimentation with different learning rates using the ReLU activation function.

Learning rate	Loss	Training Accuracy	Test Accuracy
0.001	567.09	89.57	88.02
0.1	31.45	98.24	90.36
0.5	Nan	10	10
1	4330.56	10	10
10	Nan	10	10

When training the network with a small learning rate of 0.001, the network learns the problem, but it takes way longer to get to good accuracy. When the learning rate is 0.1, the model performs better compared to the other learning rates; we get the smallest loss in this case. However, when we increase the learning rate to 0.5, the loss values initially decreases for a few epochs but eventually diverge into infinity. Therefore, python returns a not a number (Nan) value. For learning rate 1 the loss does not diverge into infinity, but the loss remains at a very high value of 4330. For all the three learning rates of 0.5, 1, and 10, training and test accuracy get stuck at 10 and never updated.

2.4 Now, add a dropout of 0.3 rate on the second fully connected layer. What is the impact of dropout on the performance? Provide the plot for training and test after each epoch. What happens if you decrease or increase the dropout rate?

Dropout is a regularization technique, and it works by probabilistically removing neurons to a layer. It reduces over fitting and improves the generalization of the model. When we apply a dropout rate of 0.3 to the second layer, we see the training accuracy slightly increases from 98.24 to 98.65, and test accuracy increase a bit in return (90.36 to 91.03), but the loss value remains the same. Suppose we set the dropout probability to a very high value, e.g. 0.9, both training and testing performance decrease to 96.52 and 89.4, respectively. This happens because, with a high value of dropout, we neutralize too many neurons. In other words, more neurons are inactivated. A too high dropout may lead to under fitting, but in our case, since we are only applying dropout to one layer, the high value compensates for the other layers without dropout. With lower dropout values, e.g. 0.05, the model performs as it did without dropout due to a small number of neurons being ignored. Too low dropout values cannot have a regularization effect.

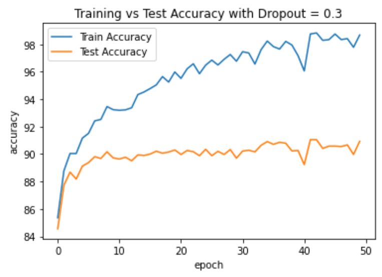

Figure 2: Training and testing accuracy, Dropout = 0.3