Artificial Intelligence – Logic and reasoning, neural networks and classification

Question 1: Crystal clear! (Logic problem):

(a) Expressing Madame Irma's six statements into First Order Logic (FOL)

- $\exists_x$ dog(x) $\wedge$ have(you, x)
- $\exists_x$ Robin(x) $\wedge$ buys(x, carrots)
- $\forall_x \forall_y$ (owns(x, y) $\wedge$ rabbit(y) $\rightarrow$ $\forall_z \forall_w$ (rabbit(w) $\wedge$ chases(z, w) $\rightarrow$ hates(x, z)))
- $\forall_x$ dog(x) $\rightarrow$ $\exists_y$ rabbit(y) $\wedge$ chases(x, y)
- $\forall_x$ (buy(x, carrots) $\rightarrow$ $\exists_y$ (owns(x, y) $\wedge$ (rabbit(y) $\vee$ grocery(y))))
- $\exists_x \exists_y \exists_z$ (owns(y, z) $\wedge$ hates(x, z) $\rightarrow$ $\neg$ date(x, y))

We use constants: Robin, c, me. Functions are: F(a) and G(b)

(b) Expressions to Conjunctive Normal Forms (CNFs)

- owns (me, c)
- buys_carrots (Robin)
- $\neg$ owns (x, y) $\vee$ $\neg$rabbit(y) $\vee$ $\neg$ rabbit(w) $\vee$ $\neg$ chases (z, w)) $\vee$ hates (x, z)
- $\neg$dog(a) $\vee$ rabbit(F(a))
- $\neg$dog(a) $\vee$ chases (a, F(a))
- $\neg$owns (x, y) $\vee$ $\neg$ hates(z, y) $\vee$ $\neg$ date(z, x)
- dog(c)
- $\neg$ buys_carrots (b) $\vee$ owns (b, G(b))
- $\neg$ buys_carrots(b) $\vee$ rabbit (G(b)) $\vee$ grocery(G(b))

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

- FOL: $\neg$ ($\exists_x$ grocery(x) $\wedge$ owns(Robin, x)) $\rightarrow$ $\neg$ date(Robin, me))
- Negate and convert to CNF:
- $\neg$ ($\exists_x$ grocery(x) $\wedge$ owns(Robin, x)) $\wedge$ $\neg$ $\neg$ date(Robin, me))
  - as $\neg$ (P $\rightarrow$ Q) = P $\wedge$ $\neg$ Q
- ($\forall_x \neg$ grocery(x) $\vee$ $\forall_x \neg$ owns(Robin, x)))) $\wedge$ date(Robin, me)
- ($\neg$grocery(y) $\vee$ $\neg$ owns(Robin, y)) $\wedge$ date(Robin, me)

Conclusion:

- $\neg$ grocery(y) $\vee$ $\neg$ owns(Robin, y)
- dates (Robin, me)

(d) (20 points) Based on all the previously created CNF (you should have at least 7 depending 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.

- date(Robin, you) $\neg$ hates(x7, x9) V$\neg$ own(x8,x9) V$\neg$ date(x7x28)
  {Robin/x7, you/x8}
  Result: $\neg$ hates(Robin, x9) $\neg$ owns(you,x9)

- $\neg$ hates(Robin, x9) V$\neg$ owns(you,x9) $\neg$ rabbit(x1)V$\neg$owns(x2,x1)V$\neg$ chases(x3,x4)V$\neg$ rabbit(x4) V hates(x2, x3)
  {Robin/x2, x9/x3}
  Result: $\neg$ rabbit(x1)V$\neg$ owns(Robin, x1)V $\neg$ chases(x9,x4)V$\neg$ rabbit (x4)V$\neg$ owns(you, x9)

- rabbit(x1) V ¬ owns(Robin, x1) V ¬ chases(x9, x4) V ¬ rabbit (x4) V ¬ owns(you, x9) owns(you, a)
  {a/x9}
  Result: ¬ rabbit(x1) V ¬ owns(Robin, x1) V ¬ chases(a,x4) V ¬ rabbit (x4)

- ¬ rabbit(x1) V ¬ owns(Robin, x1) V ¬ chases(a, x4) V ¬ rabbit (x4)
  ¬ dog(x5) Vchases(x5, f1(x5))
  {a/x5, f1(a) /x4}
  Result: ¬ rabbit(x1) V¬ owns(Robin, x1) V ¬ rabbit(f1(a)) V¬ dog(a)

- ¬ rabbit (x1) V ¬ owns(Robin, x1) V¬ rabbit (f1 (a)) V¬ dog(a) dog(a)
  Result: ¬ rabbit (x1) V ¬owns(Robin, x1) V ¬ rabbit (f1(a))

- ¬ rabbit(x1) V¬ owns (Robin, x1)V¬ rabbit (f1(a)) ¬ buybushel(x6)V ¬ rabbit(f2(x6)) V grocery(f2(x6))
  {f2(x6)/x1}
  Result: ¬ owns(Robin, f2(x6)) V ¬ rabbit (f1(a)) V ¬ buybushel(x6) V grocery(f2(x6))

- owns(Robin, f2(x6)) V ¬ rabbit(f1(a)) V ¬ buybushel(x6) V grocery(f2(x6)) buybushel(Robin)
  {Robin/x6}
  Result: ¬ owns(Robin, f2(Robin)) V ¬ rabbit (f1(a)) V grocery(f2(Robin))

- ¬ owns(Robin, f2(Robin)) V ¬rabbit(f1 (a)) V grocery (f2(Robin))
  ¬dog(x5) V rabbit(f1(x5))
  {a/x5}
  Result: ¬owns(Robin, f2(Robin)) V grocery (f2(Robin)) V dog(a)

- ¬ owns(Robin, f2(Robin)) V grocery(f2(Robin)) V ¬ dog(a) dog(a)
  Result: ¬ owns(Robin,f2(Robin))V grocery(f2(Robin))

Question 2: Lost in the closet (Classification):

(a) Loss function to use:

Since this is a classification problem the **categorical Cross Entropy Loss** would be the appropriate loss function to use. The cross entropy loss function measures the performance of the classification model which has output of a probability between 0 and 1.

Pseudocode	Math (number of classes)	

def CrossEntropy(yHat, y):
  if y == 1:
  return -log(yHat)
  else:
  return -log(1 - yHat)
	binary classification, M = 2	$-(y\log(p)+(1-y)\log(1-p))$

	Multi class classification, M>2	$-\sum c=1My_{o,c}\log(p_{o,c})$

(b) Using the Xavier initialisation, use ReLU as the activation function, a learning rate of 0.1 with the SGD optimiser. Training neural network over 50 epochs.

Over time, as the training and test accuracy improves, the training loss decreases. The loss decreases dramatically for the first few epochs, then gradually decreases over the following

epochs. The test accuracy is approximately 89 percent, while the training accuracy is approximately 98 percent.

Out[58]:

	Final Train Accuracy	Final Test Acurracy
0	0.9822	0.898

Out[59]: <matplotlib.legend.Legend at 0x7fdb6f7fbe80>

(c) Activation function: Tanh, Sigmoid and ELU. Learning rates: 0.001, 0.1, 0.5, 1, 10.

Problem faced with tanh and sigmoid function is that they saturate. The large values are snapped to 1.0 small values are snapped to either -1 or 0 respectively for tanh and sigmoid. Majorly, the functions are only sensitive to changes around the mid-point of their inputs, eg: 0.5 for sigmoid and 0.0 for tanh. After reaching this saturation, it becomes difficult for the learning algorithm to continue to adapt weights for improving the model. The ELU function produces more accurate results as it tends to converge cost to zero faster. The below table on the *left* side. It can be noticed that the final training and test accuracies for *elu* do not show much of the difference whereas the final training accuracy has saturated to 1.0 in case of *tanh* activation function. In the case of ReLU, 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. The same has been displayed in the graph on the right side with different learning rates : 0.001, 0.100, 0.500,1.000 and 10.000.

	activation	final_train_acc	final_test_accuracy
1	tanh	1.000000	0.9117
2	sigmoid	0.939467	0.9024
3	elu	0.994233	0.9032

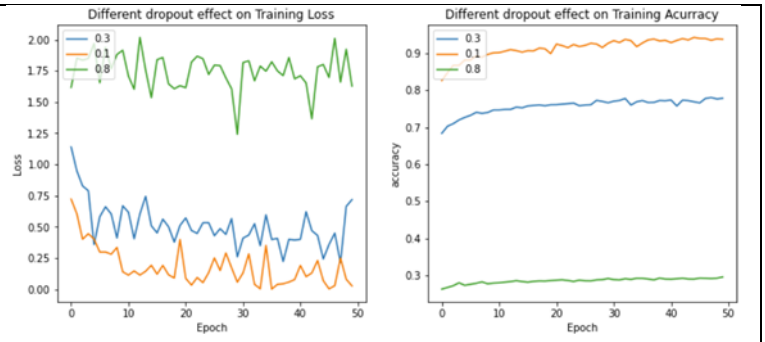
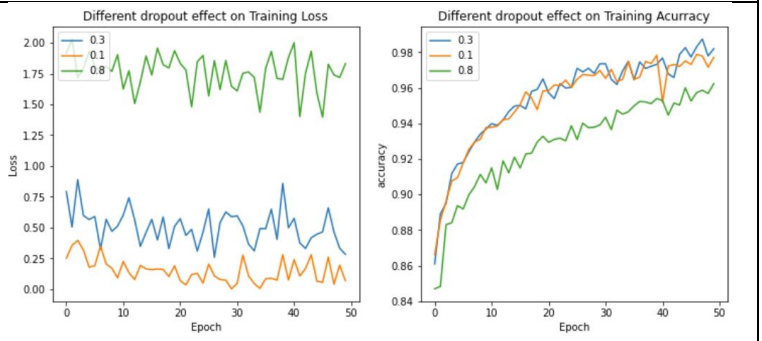
Table for Tanh, sigmoid and elu

	activation	learning_rate	final_train_acc	final_test_accuracy
4	relu	0.001	0.886917	0.8749
5	relu	0.100	0.975883	0.8960
6	relu	0.500	0.100000	0.1000
7	relu	1.000	0.100000	0.1000
8	relu	10.000	0.100000	0.1000

Experimentation with various learning rates

(d) Effect of dropout rate:

Dropout layer acts as a mask that nullifies the contributions made by some neurons towards next layer while leaving others modified. They are important in CNN to prevent overfitting on training data. In their absence, the initial batch of training samples will disproportionately influence learning in drastic way. Thus, preventing learning of features that tend to appear in the later batches only. As seen below are graphs with 3 different dropout rate: 0.3, 0.1 and 0.8 with their effects on the training loss and accuracy.

t_loss = df[-3:]['t_loss'] t_acc = df[-3:]['t_acc']	
t_loss= df[-6:-3]['t_loss'] t_acc = df[-6:-3]['t_acc']	

References:

1. https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e
2. https://towardsdatascience.com/machine-learning-part-20-dropout-keras-layers-explained-8c9f6dc4c9ab
3. https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html
4. https://www.baeldung.com/cs/ml-relu-dropout-layers