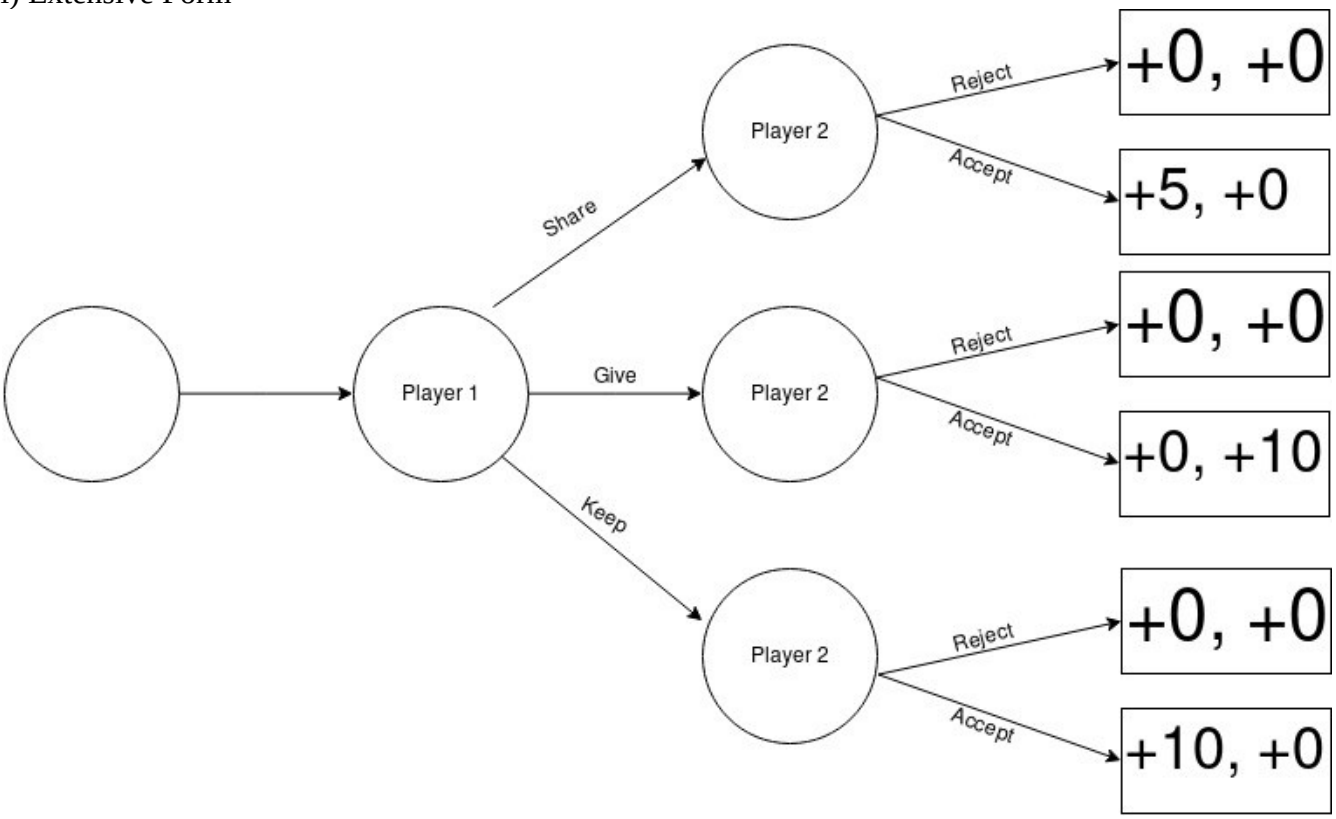
COSC-4368 AI

Problem Set 2

James Richardson
1555520

Problem 7

i) Extensive Form



ii) Normal Form

Player 1 Player 2	Accept	Reject
Share	+5, **+5**	+0, +0
Give	+0, +10	+0, +0
Keep	+10, +0	+0, +0

iii)
Pure Nash Equilibria: (Keep, Accept), (Keep, Reject)
Mixed Nash Equilibria: All of the other states

Problem 8

a)
Q-Learning Equation used:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

SARSA equation used:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma\, Q(s',a') - Q(s,a))$$

i)
q(b,2) = 0 + 0.5(2 + 1 * max(1, -0.5) – 0) = 1.5
q(b, 1) = -0.5 + 0.5(0 + 1 * max(0, 1) - -0.5) = 0.25
q(b, 3) = 1 + 0.5 * (-1 + 1 * max(1.5, -0.5)) = 1.25
q(b, 2) = 2 + 0.5 * (1 + 1 * max(1, 0.25) – 2) = 2.0

	Value
q(a, 1)	1
q(b, 1)	0.25
q(a, 2)	-0.5
q(b, 2)	2.0
q(a, 3)	0
q(b, 3)	1.25

ii)
q(b, 2) = 0 + 0.5(2 + 1 * (-0.5) – 0) = 0.75
q(b, 1) = -0.5 + 0.5 * (0 + 1 * 1 - - 0.5) = 0.25
q(b, 3) = 1 + 0.5 * (-1 + 1 * 0 – 1) = 0.0
q(b, 2) = 0.75 + 0.5 * (2 + 1 * (0.25) – 0.75) = 1.5

	Value
q(a, 1)	1
q(b, 1)	0.25
q(a, 2)	-0.5
q(b, 2)	1.5
q(a, 3)	0
q(b, 3)	0.0

b) Since SARSA considers the actual action that will be taken in the next state rather than the best action that could be taken, we will get a better idea of the value of a particular action in terms of our current policy. We don't wait a particular option as better just because we COULD make a good move in that state.

c) With expected SARSA we consider the expected value of our next state rather than applying the policy to the next state and determining the next move.
Expected SARSA would be nice when the policy is random, or dynamic in some way, and you can't actually determine what the action in the next state will be. If the policy is non random and the value of the action that will be selected in the next state is fluctuating, Expected SARSA would be better because we would get a constant average rather than a fluctuating value.

d) With reinforcement learning you don't exactly know what is right and what is wrong. We have relative ideas of how good things are based on their surroundings. In a supervised learning example, such as classification, something is either classified as one thing or another. In reinforcement learning we don't know the value off a particular state, this is something that must be determined over time.

Its comparable to the difference between someone telling you exactly why you got a problem wrong on the homework vs receiving a grade and trying different methods of solving that problem while incrementally shooting for a better score.

Problem 9

a) Back Propagation is used in neural networks that have a hidden layer; hidden layers can only occur in neural networks with more than 2 layers.

In training a neural Network, the goal is to reduce the error to a particular value. This is done by adjusting the weight constants that link the nodes of different layers together. For the layers at the beginning and end of the Network, this is easy because they are exposed. To determine the error at a particular hidden layer of the neural network and thus how to adjust the weights to reduce this error, Back Propagation is used.

Back Propagation will 'propagate' the error 'back' into the hidden layers.

b) The derivative of the input to the outer node, the weight between the outer node and inner node as well as the error of the outer node.

c) Deep Neural Networks can both determine the features of a data set as well as classify those features. Rather than first having to create a function which extract features and then using those features in a traditional Neural Network, a Deep Neural Network can do all that work on its own.

d) The second term minimizes the testing error. The ξ corresponds to the error itself while the C in front of the Sum is a constant that specifies how much we care about the margin.

The advantage of using the soft margin approach is that the function does not have to me mapped into a higher dimension, making the model less complex and thus more generalizable. Generally with SVMs, if there is not a clear line that could act as the Maximum Margin Separator and separate the two classes, a Kernel function would would be used to map the data into a space such that the two classes are able to be linearly separated.

Using the soft margin in this case allows us to accept data of the wrong class on the wrong side of the Maximum Margin Separator, though with some penalty, meaning that a kernel function does not have to be used to map the data into a higher dimensional space.

e) With an SVM, depending on which side of the Margin a point falls determines which class it will be identified as. The dotted lines are known as the support vectors and they determine a 'safe' distance form the margin for point to definitively be identified as that particular class. So any case in which a point falls beyond the support vectors will have some error value and will be less likely to be classified correctly.

The lower ξ is, the higher the the more correct that the point was classified correctly. It will tell us how wrong our answer is.

Problem 10

For this problem I originally trained 8 models in total, 4 that were Support Vector Machines and 4 that were Neural Networks. The Support Vector Machine was implemented using Sci-Kit Learn's *SVC* object and the Neural Network was implemented using their *MLPClassifier*. The default parameters were used unless they were explicitly specified in the table.

After working through my theory regarding the performance of the models though, I decided to try some additional parameters to support my theory.

The patient id as well as the classifier were removed prior to normalizing the data using sklearn's *normalize* function. Then all models were ran through 10 Fold Cross Validation using Sci-Kit Learn's *cross_validate* function. This function outputs a dictionary with each of the calculated scores; each of the entries from this dict point to a list of size 10 (since 10 Fold was used). The test score entry from this dict was used to calculate the 'Average Test Score' and 'Max Test Score' columns of the following tables.

Support Vector Machine

Kernel	Gamma	Average Test Score	Max Test Score
Linear	0.001	0.8415932071558204	0.8771929824561403
Linear	0.5	0.8415932071558204	0.8771929824561403
Poly (degree 3)	0.001	0.628516333938294	0.631578947368421
Poly (degree 3)	0.5	0.6655183216662346	0.6964285714285714

Neural Network (one hidden layer of size 100)

Activation Function	Learning Rate	Average Test Score	Max Test Score
Logistic	0.1	0.911993345432547	0.9473684210526315
Logistic	0.5	0.628516333938294	0.631578947368421
Tanh	0.1	0.8906911675741076	0.9310344827586207
Tanh	0.5	0.6759301270417423	0.9107142857142857

The Neural Network with a lower learning rate, regardless of the activation function, had a significantly higher accuracy than any of the State Vector Machine results. Both of the Neural Networks that used a higher learning rate performed pretty badly overall, though better than the Support Vector Machine with the polynomial kernel. The Neural Network using a Logistic Activation Function and a learning rate of 0.1 performed the best overall, and the Support Vector Machine with Polynomial Kernel and Gamma of 0.001 tied for worst overall with the Neural network with Logistic Activation function and Learning Rate of 0.5.

The best explanation for the behavior of the Support Vector Machine performance is that **the data was already linearly separable** and trying to separate it with a degree 3 polynomial just complicated things and made it more difficult to separate. I ran this example again with a Polynomial kernel of degree 2 and degree 4 in an attempt to show that the higher the degree of the polynomial, the more difficult to separate the data.

Support Vector Machine

Kernel	Gamma	Average Test Score	Max Test Score
Poly (degree 2)	0.001	0.628516333938294	0.631578947368421
Poly (degree 2)	0.5	0.7198470313715324	0.7586206896551724
Poly (degree 4)	0.001	0.628516333938294	0.631578947368421
Poly (degree 4)	0.5	0.6319948578342408	0.6491228070175439

We do see in fact that the lower the degree, the better the overall score seems to be. Though even with this considered, the Support Vector machine that used the linear kernel performed better, further pointing towards the conclusion that the data was already linearly separable. For a low Gamma value however, the Error Function appears to get stuck at a local minimum. Interestingly this is the same local minimum that the worst Neural Network example got stuck at.

If we look at the equation for the polynomial kernel, we can get an idea of why the lower gamma gets us stuck at a local minima.

polynomial: $(\gamma\langle x, x'\rangle + r)^d . d$

Gamma is the weight for the dot product. According to sklearn's documentation of Support vector machines, "`gamma` defines how much influence a single training example has." The higher gamma is, the smaller the variance is. This implies that the data is clumped closely together. We are overshooting a better minimum since we are assuming our variance is high.

Neural Network with Single Hidden Layer of Size 30

Activation Function	Learning Rate	Average Test Score	Max Test Score
Logistic	0.1	0.9208279318987124	0.9649122807017544
Logistic	0.5	0.628516333938294	0.631578947368421
Tanh	0.1	0.924149814190649	0.9649122807017544
Tanh	0.5	0.7969870797683865	0.9473684210526315

We can see that when the number of neurons in the hidden layer is equal to the number of features in the data set, the accuracy is significantly higher (especially for a lower learning rate). This probably lead to each neuron in the hidden layer being able to activate for a single particular feature. The outer layer was then able to determine based on which features were activated which class the particular example should be classified as.

The reason that tanh performs better than the sigmoid (logistic) function is because the derivative of the tanh function has significantly higher gradients and could push the error more in the right (lower) direction.

Problem 11

a. There is a least one corrupt politician in San Antonio.

C(x) is the predicate meaning x is corrupt
S(x) is the predicate meaning x lives in San Antonio

The universe contains, and is limited to, all politicians.

$\exists x[C(x) \wedge S(x)]$

There exists some politician such that the politician is corrupt and the politician is from San Antonio.

b. Every man loves a least one woman.

L(x, y) is the predicate meaning x loves y
M(x) is the predicate meaning x is a man
W(y) is the predicate meaning y is a woman

The universe contains, and is limited to, all male and female humans.

$\forall x \exists y [M(x) \wedge W(y) \wedge L(x, y)]$

For all people x, there exists some person y, such if x is a man and y is a woman then x loves y.

c. Neither red frogs nor green frogs eat grass.

R(x) is the predicate meaning x is red
G(x) is the predicate meaning x is green
E(x) is the predicate meaning x eats grass

The universe contains, and is limited to, all frogs.

$\forall x [[R(x) \wedge G(x)] \rightarrow \neg E(x)]$

For all frogs, if the frog is red or the frog is green then the frog does not eat grass.

d. There are at least 2 yellow houses in Harris County that have been flooded.

Y(x) is the predicate meaning x is yellow
H(x) is the predicate meaning x is in Harris County
F(x) is the predicate meaning x has been flooded
N(x, y) is the predicate meaning x and y are not the same entity

The Universe contains, and is limited to, all houses in Texas.

$\exists x \exists y [N(x, y) \wedge Y(x) \wedge Y(y) \wedge H(x) \wedge H(y) \wedge F(x) \wedge F(y)]$

There exists some house x and there exists some house y such that x and y are not the same house, x is in Harris County, Is Yellow and has been flooded and y is in Harris County, Is Yellow and has been flooded.

e. There is a dogcatcher in Texas who caught at least one dog in every county in Texas.

D(x) is the predicate meaning x is a dogcatcher
U(x) is the predicate meaning x is a county in Texas
V(x, y) is the predicate meaning x has caught at least one dog in y

The universe is the set of all dogcatchers and all counties in Texas

$\exists x \forall y [[D(x) \wedge U(y)] \rightarrow V(x, y)]$

There exists some dogcatcher or county in Texas x such that for all dogcatchers and counties in Texas y, if x is a dog catcher and y is a county in Texas then x has caught at least one dog in y.