Learning and Computational Finance and Economics
Answers to Test 2009-10

Answer 1:

(a) [Book work]

* Accuracy: true positive + true negative divided by total number of cases.
* Precision: true positive divided by true positive + false positive
* Recall: true positive divided by true positive + false negative

(b) There is no such thing as “the best way” to measure performance in machine learning. [Marks will be given to answers that point this out.]
The choice of measure depends on the application.
A weighted sum of them could be a good measure. So as ROC, which is covered in textbooks. [Marks will be given to any of these suggestions, as long as they are justified.]

(c) [Book work]

* Function set: A tree is a function. The Function set is the set of functions that can be employed by a tree. They are the internal nodes of a tree. Example: {+, -, *, /}
* Terminal set: The terminal set is the set of variables and sometimes decisions that are considered. They are the leaf nodes of a tree. Example: {x1, x2, x3}
* Crossover operator: pick a random point in each of the parent trees, and exchange the subtrees.

(d) Each of the functions takes two arguments. For a tree of depth 3, there is a maximum of 6 internal nodes and 8 leave nodes. There are 4 possible functions and 3 possible terminals. So there are 4^6 * 3^8 combinations.

Answer 2:

(a) [Book work]
Two players bargain to share a pie. They make alternative offers. Their utilities drop exponentially over time. This motivates them to accept an offer as soon as possible.

(b) A player's optimal strategy depends the other player's strategy. Subgame equilibrium is the optimal strategy by both players, given their belief of the opponents' strategies. It was derived recursively by Rubenstein: To calculate Player A's optimal strategy, one has to solve the subproblem of Player B's strategy had B rejected A's initial offer (whatever that is). This in turn can be calculated by solving A's optimal strategy in the third round should A reject B's offer in the second round. The subproblems can be solved recursively till infinity, which has a fixed point when both players' utilities drop to a limit.
Rubenstein assumed complete information, i.e. each player know both discount rates, and know that the opponent knows such.
Rubenstein also assumed perfect rationality by both players.

(c) To apply genetic programming, one has to represent solutions.
Solutions are represented by trees in genetic programming.
Each tree is a strategy.
Populations of trees are maintained, one representing strategies by Player A, the other by Player B.
Each strategy in Player A is evaluated through playing it against the players in the population for Player B.
The populations are evolved using genetic programming operators, which may vary.
This approach is attractive because
(i) It does not assume perfect rationality as Rubenstein did.
(ii) It is a general method which can be applied to a wide range of situations
(iii) Computing subgame equilibrium requires less effort.

(d) Given an offer, say, 0.75, one could consider all possible subgame equilibrium for the opponent. Each counter offer by the opponent must be evaluated through the enumeration of all counter-counter offers, and so on. This has to be calculated till the utilities drop to their limits (which is called the "discount factors" by Rubenstein), in which case the optimal strategies can be calculated.
It is not feasible because of combinatorial explosion. The number of combinations to evaluate is 99 to the power n (99^n), where n is the number of rounds considered. 99^n grows exponentially as n grows. Since one has to solve subgame equilibrium till the limit, n is typically large, and therefore it is infeasible to consider all combinations.

Answer 3:
This is an open question. Marks will be awarded to any sensible and relevant points. Some of the valid points include:
* This is a Boolean decision.
* This is a supervised learning problem in machine learning. You are trying to classify cases into safe (i.e. the clients will pay back the loan) default (i.e. the clients will not pay back the loan).
* You have to set your targets. In this case, the target is a Boolean decision: approve or reject the loan application.
* To tackle this as a machine-learning problem, you have to collect data.
* The main question is what data to be collected. This requires expertise.
* This is a classification problem. You need to identify the variables that contribute to the classification.
* Your variables could be numerical, enumerative or Boolean.
* Is learning possible? You need to evaluate whether there exists a function that maps the variables (of your choice) to the decision.
* Variables that you need are factors that you would consider in a manual approach. For example, you may consider the borrower's credit rating, cash flow,
* You need to convert those considerations into variables. This is not always straightforward. For example, for cash flow, simply supplying the program with the lists of revenues and outlets may not help. It would be more promising if they could be converted into some indicators.