**1. ‘Adult’ Dataset from UCI**

Using some continuous and categorical data of a person such as age, education, race, job title etc., predict if a person’s annual income is more than $50,000 or not.

To build the corresponding classifier, we used “adult” data set (Blake and Merz 1998) from the UCI repository. (source: <http://archive.ics.uci.edu/ml/datasets/Adult>). [1]

Provided training and test datasets have 32561 and 16281 instances respectively. After removing the unknown and duplicate data, we have 30162 training instances and 15060 test instances. Both sets are properly labeled and have 14 predictor variables of categorical and numeric types. One of the interesting features of the dataset is that it is not well balanced. In both training and test dataset about 25% instances belong to the class of people with higher income (>50k) and about 75% are in the other class. However, the datasets are already shuffled for experiments.

**Neural Network Architecture**

We used an artificial neural network with two densely connected hidden layers with 100 nodes at each. There are 14 predictor variables at the input layer and output layer just outputs either 1 for income greater than $50000 or 0 for less income. We used ABAGAIL library to code the algorithm at focus.

**Random Hill Climbing**

Random hill climbing is a simple instance-based method. It starts at a random point as the current state. Then it compares the states of its neighboring instances. If the neighboring instances optimizes the function better, it makes an increment to that point as the current state. If no better optimization state found in the neighboring instances, it stops and mark it as the local maximum.

**Simulated Annealing**

It is a probabilistic method of finding the best solution. First, we randomly select an instance at a certain temperature T. Then we sample a neighboring instance. If the neighboring instance optimizes the function better, we move the current state to the neighboring instance. If the neighboring state is worse than the current state, we use the probability of acceptance to determine whether to move to that instance. Mathematically, Probability of acceptance = exp( (f(x\_neighbor) – f(x\_current))/T) . Then at every iteration, we decrease the temperature by a cooling factor ( T = CE \* T) and repeat the process for a fixed number of iteration.

Genetic Algorithm

Genetic Algorithms initially computes the fitness of all the individuals from a population of solutions. Then the most fitted individuals are selected as the parents for the crossover step. At the crossover step, the best individuals are paired up as parents and a random crossover point is selected to produce offspring solutions. Finally, at the mutation step, some of these offspring are randomly mutated to maintain diversity. This process is repeated until it reaches the convergence.

MIMIC

Mutual-Information-Maximizing Input Clustering (MIMIC) algorithm uses the knowledge of cost function at each iteration and a probability density estimate to find the best structure of the solution space, which in turn guides to the search to the optima.