International Journal of Hybrid Information Technology
Vol.8, No.8 (2015), pp.175-184
http://dx.doi.org/10.14257/ijhit.2015.8.8.17

Hierarchical Reinforcement Learning Based on KNN Classification Algorithms

Shanhong Zhu[1,2], Weipeng Dong[1] and Wei Liu[2]

[1]*School of Computer and Information Engineering, Xinxiang University, Henan, China*
[2]*InternationalSchool of Software, Wuhan University, Wuhan, China*
zshazhu@163.com

Abstract

In recent years, machine learning is increasingly becoming an important field of computer science. A new method using KNN classification algorithm identifies the layered boundary to find subgoal condition, to automatic classifying of large state space, reaches the dimension reduction of state space, and on the basis of generated subspace classifying to structure subtasks, and then realizes the hierarchical learning tasks automatically. In autonomous system, Agent assigns to their task through interaction with the environment, using hierarchical reinforcement learning technology can help the Agent in the large, complex environment to improve learning efficiency. Through the experimental results the effectiveness of the proposed algorithm is demonstrated. The goal of this paper is to provide a basic overview for both specialists and non-specialists to how to decide a good reinforcement learning algorithm for classification.

Key Words: *KNN Classification algorithm; Reinforcement learning; Classifying Option*

1. Introduction

Reinforcement learning is put forward by Minsky for the first time in the 1950s, the problem to be solved is: an agent can sense the surrounding environment, how to choose by learning to choose reach the goal of the optimal action. When the agent is to make each activity in the environment, it will give appropriate reward or punishment, to show whether the results of the state are correct or not. For example, when agent is training for a chess game, game victory gives positive returns, but the game failed gives negative returns, the other is zero returns [1]. Agent's task is to learn from the indirect learn and delayed return, so that the subsequent action can produce the largest cumulative returns. Reinforcement learning is the mapping study of intelligent body state from the environment to the action, getting the biggest reward value of the accumulation of action from the environment. This method is different from supervised learning techniques which can inform what behavior through positive cases and counterexamples, it is to find the optimal behavior strategy through trial error.

1.1. KNN Classification Analysis

K nearest neighbors (KNN) classifier is a classical supervised method in the field of machine learning, based on statistical data. In 1967, Cover and Hart had proved the KNN classification deviation approximates to optimal Bayesian classifier [2]. The KNN classifier is widely used in such areas as text classification, pattern recognition, image processing and spatio-temporal classification.

The decision-making of the KNN algorithm is to find test sample *k* nearest or most similar training samples in the feature space, and then the test sample is assigned to a majority vote of its *k* nearest neighbors. Essentially, it is still based on statistical method.

ISSN: 1738-9968 IJHIT

In the real world, most of the practical data does not obey the typical model (such as Gaussian mixture, linearly separable, *etc*). Non parametric algorithms like KNN happen to deal with the situation. If domain classes of test sample sets has more cross or overlap, KNN is the more appropriate method.

The advantages of KNN algorithm are as follows: (1)There is no explicit training phase or it is very minimal. Once the data is loaded into memory, it began to classify. This means the training phase is pretty fast. (2)The data set has been labeled, that is to o say the selected neighbors of test sample have been classified correctly. (3)Independent of any data distribution, it only needs to be adjusted or assigned an integer parameter *k*.

The disadvantages of KNN algorithm are as follows: (1) It has a costly testing phase. The cost is in terms of both time and memory. More memory is needed to store all training data [3]-[4]. (2) It is highly dependent of number of samples. Along with the number of training samples increases, the classifier performance will be better and better. (3) Classification accuracy is affected because of the property is equivalent to the weight. (4) It is difficult to select an appropriate *k* value. Small or large value of *k* has different characteristics.

1.2. Q-learning

Reinforcement learning (RL) is a kind of method to obtain the optimal strategy through the Agent to trial and error. Standard is the discrete time MDP model with a limited set of state set S and operating set A, at any moment t, the Agent, from the current state of $st \in S$, chooses next executable action A (st), to obtain the immediate reward value*r*t and take the value function of a state-action as the target function. The purpose of reinforcement learning is to search the optimal strategy of $\pi^*: S \rightarrow A$, which maximizes the value function.

Q-learning is a kind of independent learning environment model proposed by Watkins, and its beauty is: $Q(s_t, a_t)$ value of Current state-action sums up all the needed information, to determine a discount of the total reward value of the optimal sequence which is executed:

$$Q(s_t, a_t) = r_t + \gamma \max\{ Q(s_{t+1}, a_{t+1}) \mid a_{t+1} \in A$$

Where, γ is discount factor. In Q-learning, getting a reward signal will update Q value at a time:

$$Q(s_t, a_t) = \begin{cases} (1 - \eta_k) Q_{k-1}(s_t, a_t) + \eta_k (r_t + \gamma V(s_{t+1})) & s = s_t, a = a_t \\ Q_{k-1}(s_t, a_t) & other \end{cases}$$

Where, k is the number of iterations, η_k is the learning efficiency, to control learning speed.

1.3. Option

In order to build the hierarchical learning system, we make the introduction of semi-markov process (SMDP). Between state s and execution action a has a time interval τ.Option is a macro Q-learning proposed by Sutton in 1999. Learning task is abstracted into several options, and will make these options as a special kind of "action" added to the original action focus. We use a triple (I_i,π_i,β_i) to denote an Option. $I_i \in S$ is state set for entry, $\pi_i$is internal strategy, and $\beta_i$is termination condition. Internal strategy includes the strategy $\pi_{ia}: I_i \rightarrow A_i$ defined on the original action and the strategy $\pi_{io}: I_i \rightarrow O_i$ defined on Option, so (I_i,π_{io},β_i) forms the layered Option.

According to the optimal strategyπ_{io}*, the implemented state-action value function is defined as:

$$V_o^*(s_t) = \max_{o \in O_s} \left[R(s_t, o) + \sum_{s_{t+1}} P(s_{t+1} \mid s, o) V_o^*(s_{t+1}) \right]$$

According to the optimal strategyπio*, the implemented state-Option value function is defined as:

$$Q_o^*(s_t, o_t) = R(s_t, o_t) + \sum_{s_{t+1}} P(s_{t+1} \mid s_t, o_t) \max_{o \in O_{s_{t+1}}} Q_o^*(s_{t+!}, o_{t+1})$$

The Option is updated only at the end of an Option in the Option study, the corresponding update formula is:

$$Q_{k+1}(s_t, o_t) = (1 - \eta_k) Q_k(s_t.o_t) + \eta_k \left[r_t + \gamma^{\tau} \max_{o \in O_{s_{t+1}}} Q_k(s_{t+!}, o_{t+1}) \right]$$

2. Hierarchical Reinforcement Learning Method based on KNN Classification

Using the Option algorithm has been shown to effectively improve the learning speed of the current task. Based on the existing Option method, this paper proposes a hierarchical reinforcement learning method based on KNN classification, better realizes the automatic layered learning task [5]. In the start of the study, each Option contains only one entrance state, and executes flat strategy. After multiple learning cycles which are fully detected, using KNN classification algorithm to classify state space, at the same time generates each Option under the classifying space, and completes the internal strategy in the learning process of learning, to realize automatic layering. Based on the improved KNN classification Option, automatic generation algorithm is as follows:

1) Initialization of the input data, the space coordinates of the detected state are as input data object.

2) Initialize the parameters of KNN classification.

3) Set population number N, the classifying center is encoded by real number, and randomly produces initial population. It's important to note the populations of the selection, if too small, the performance of genetic algorithm will become very poor or not to find out the solution of the problem, if too big, it can increase the amount of calculation and make growth convergence time. It is more appropriate generally between 30 and 160.

4) Decode initial populations, calculate the fitness of each string of genes.

5) The largest individual fitness individuals, namely the best individuals in the population are unconditionally copied to the next generation of new populations, then make the parent species selection, crossover and mutation genetic operators et al, so as to produce the next generation of new species of other N-1 genes. Wheel method is usually used as the selected method, the choice of electing big fitness gene string is big, and thus be heredity to the next generation, on the contrary, the choice of small fitness gene string is small, which will be eliminated. Crossover and mutation is to generate new individual genetic operator, crossover rate is too big, which will lead that high fitness gene string structure is quickly destroyed, too small makes search stalled, it generally takes 0.25 ~ 0.75.Mutation rate is too big, which will lead that genetic algorithm is a random search, too small will not produce new individual, it usually takes 0.01 ~ 0.2.

6) If reaching the set number of generation, it returns the best gene string, and ends the algorithm. Otherwise, it returns to 3) to continue to the next generation.

7) Each classifying removes all the state set besides adjacent state as the Option of launching state set, for each classifying boundary condition (export), the termination

condition the Option is set to 1, random to set up the Option internal initial strategies.

8) End of the algorithm.

In learning initial stage, Option is not formed, according to the standard of Q learning method agent detects and learns in the state space. After the classifying, on the state of each subset through experience makes playback to produce internal learning policyset, generates a new Option. By the KNN classification algorithm to classify the state space and create the Option, the whole algorithm KNN main process is as follows:

a) Observe the current state.

b) Call Option value function updating algorithm.

c) If meets the end of the study conditions, it turns i, or turns d.

d) If the Option has been generated, then it turns a, or e.

e) Record the environmental information.

f) If not present new state N cycle in a row, then turns g, or turns a.

g) Call Option automatically generated algorithm based on improved KNN CLASSIFICATION.

h) Turn a.

i) End.

Among them, the step c refers to learning end conditions which have a variety of setting methods, such as to the expected demand learning cycle error et al. The whole KNN algorithm flow chart is shown as Figure 1:

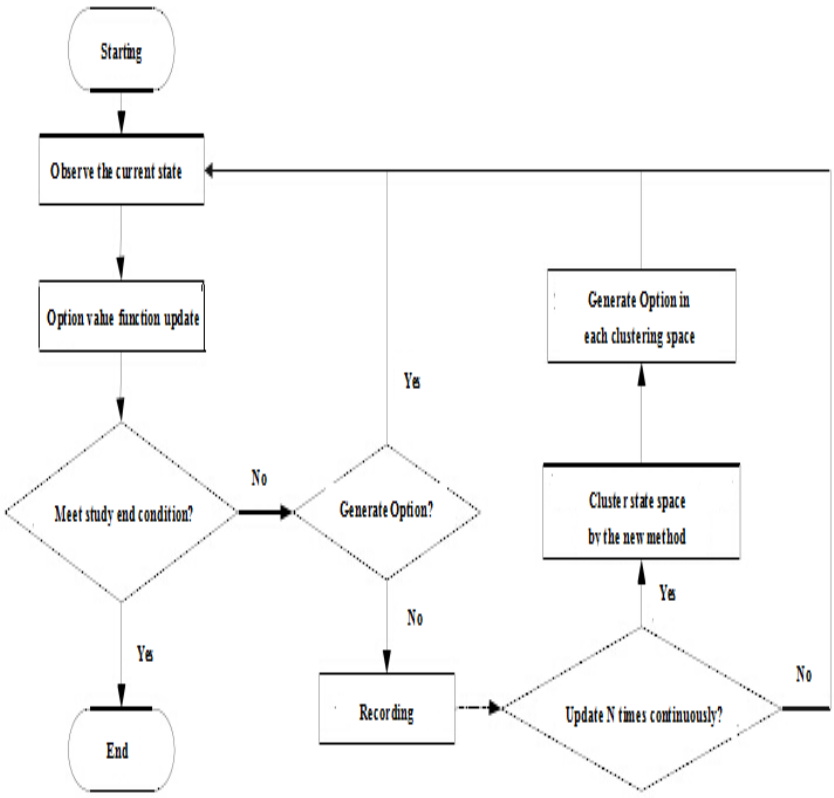

Figure 1. The Whole KNN Algorithm Flow Chart

In its original form, the nearest neighbor classifier is perhaps the simplest pattern classification algorithm, the classifier assigns any input feature vector to the class indicated by the label of the nearest vector in the reference sample. More generally, the *k* nearest neighbors classifier maps any feature vector to the pattern class that appears most frequently among the *k* nearest neighbors. The algorithm steps can be described as follows:

Algorithm: Simple KNN pseudo code

Procedure: Find class labels.

Input: *k*, the number of nearest neighbors; *D*, the set of test sample; *T*, the set of training sample.

Output: *L*, the label set of test sample.

1: readDataFile(TrainingData)
2: readDataFile(TestingData)
3: L={}
4: for each d in D and each t in T do
5: Neighbors(d) ={}
6: if |Neighbors(d)|<k then
7: Neighbors(d)=Closest(d, t)$\cup$ Neighbors(d)
8: endif
9: if |Neighbors(d)| $\geq$ k then
10: break
11: L =testClass(Neighbors(d)) $\cup$ L
12: endfor

Notes:

Neighbors(d) return the k nearest neighbors of d
Closest(d, t) return the closest elements of t in d
testClass(S) return the class label of S

Generally, the training phase is executed once, and the classification phase is executed many times until all test samples are classified.

2.2. Different Parameter for KNN

The classification efficiency mainly depends on three factors: k value-the number of neighbors, distance metric and decision rule. We can change these parameters to improve classification efficiency.

2.2.1. K-value Selection

Small or large value of k has different effects (refer with: Table 1), Heuristic techniques such as cross-validation can help in selecting a good value for k.

Table 1. Features of Small or Large Value of *k*

Small value	**Large value**
Over-fit	Generalization
Negative effect of noise	Reduce negative effect of noise
Create distinct class boundaries	Create distinct class boundaries

Generally, the k value is a relatively small integer. If $k = 1$, that is nearest neighbor algorithm, the label of object's nearest neighbor is assigned to it.

2.2.2. Distance Metric

Different distance metrics can be used when calculating distance between the object points.

Minkowski Distance: It can be universally described as Minkowski metric which is given in Equation 1.

$$dist(X,Y) = \left(\sum_{1}^{n} \left| x_i - y_i \right|^p \right)^{1/p} \quad (1)$$

Where different values of $p \geq 1$ result in different commonly-used metric.

Manhattan Distance: When $p = 1$, that is Manhattan distance (Equation 2). It is the absolute distance total between two points on the standard coordinate system. The correlation of different features is not taken into account here.

$$dist(X,Y) = \sum_{i=1}^{n} \left| x_i - y_i \right| \quad (2)$$

Euclidean Distance: When $p = 2$, that is Euclidean distance (Equation 3).It is the most common metric used in KNN.

$$dist(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad (3)$$

Euclidean distance measures an absolute distance between various points in a multidimensional space. At the same time, Euclidean metric need to ensure dimension indicators in the same level because of distance calculating based on the absolute values of various dimension characteristics. This metric should be used when the different features are not strongly correlated.

Chebyshev Distance: When $p \to \infty$, Equation 1 is represented as:

$$dist(X,Y) = \lim_{p \to \infty} \left(\sum_{i=1}^{n} \left| x_{i\_} y_i \right|^p \right)^{1/p} = \max \left| x_i - y_i \right| \quad (4)$$

Mahalanobis Distance: Since Euclidean distance cannot ignore differences in metric indicators, the data need to be standardized before using the Euclidean distance. That can derive another distance measure-Mahalanobis distance. Let the vectors x and y be two input samples of the same distribution with the covariance matrix P. The Mahalanobis distance between sample x and y is defined as:

$$dist(X,Y) = \sqrt{(x-y)^T \sum^{-1} (x-y)} \quad (5)$$

This metric should be used when the different features are strongly correlated. The covariance matrix $\sum$ represents this correlation.

2.2.3. Decision Function

The KNN classifies each unlabeled example by the majority label among its k-nearest

neighbors in training set. Its performance thus depends crucially on the distance metric used to identify nearest neighbors [7, 8]. However, the principle based on maximum voting easily leads to equal votes of neighbors, that's draw phenomenon. In addition, if the quantity of sample is unbalance, such as one class samples is amount, the others is minority, which cause the samples of the amount class are majority in k nearest neighbors when a test sample is coming. In reality, especially in small sample cases such as curse dimensionality and existing outliers, the sample quantity is little which usually leads to a dramatic degradation of classification accuracy. Thus different weighting methods are be used.

The goal of weighted functions is to make the distant neighbors have less effect on the majority vote than the closer neighbors. Most commonly used weighted functions of KNN are attribute weighted and instance weighted.

3. The Experimental Simulation and Analysis

In order to assess the performance of the proposed method, it is tested under the environment of the taxi [9]. As shown 5 * 5 grid environment in Figure 2, it contains a taxi and a passenger. Specify the location of the four special B, G, R and Y, passengers can choose any position of the four to get on the bus, and then get off at any position. The mission of the taxi is to the initial position to passengers, takes him to the destination and puts him down. Each grid position on the taxi has six original actions: North, South, East, West, Pickup and Putdown, the probability of each movement is performed at 0.8. When the taxi arrives to the initial position of passengers the Pickup action is effective, and will gain + 10 bonus to perform. At the same time when the taxi has passengers and on the target position, the Putdown actions are effective, performing bonus of + 20, but performing other actions is - 1.

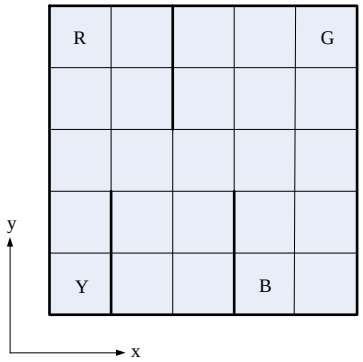

Figure 2. The Taxi Problem

To reflect whether the proposed new algorithm performance is good, the experimental results are compared with the Q-learning and SCC. Three lines in Figure 3 are respectively the taxi experimental results to solve the problems by Q-learning algorithm, SCC algorithm and the Ant-O algorithm. The ant algorithm to optimize system parameter is set tont=200, nk=10, α =0.98, ρ =0.98. Experiment has two stages, in the initial learning stage, the Agent is traversal in the environment, and records the transformation process. Then based on the information transition diagram, obtains subgoals. Next, the Agent uses the Option learned in the previous stage, to continue the interaction with the environment, learns the optimal strategy.

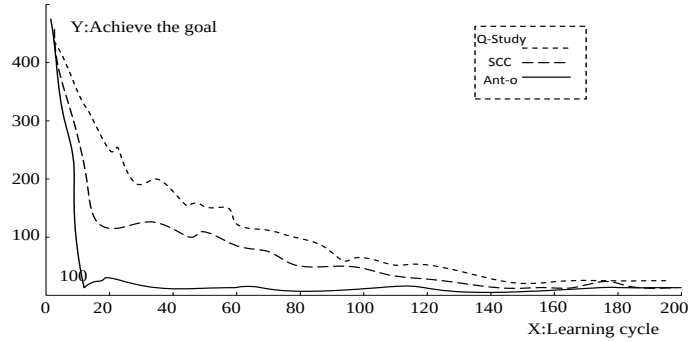

Figure 3. Algorithm Performance Comparison

Experimental results are shown in figure 3, with the goal state steps taken to depict algorithm performance, steps in continuous cycle reaches an average of more than 40 steps [10]. These three methods are used to a critical state, it effectively navigates interaction the weighted graph, which is a useful subgoal. It can be seen from the diagram, steps traverse gradually reduce with the increase of learning cycle, curve is gradually convergence, and three methods are ultimately to find the optimal path. However Ant-O algorithm proposed in this paper in a relatively short period of learning, makes search faster convergence to the optimum solution, shortens the time to learn. Compared with the Q-learning algorithm, the Ant-O introduces orderly directed and weighted graph, which is arranged on the shortest path pheromone concentration.

4. Conclusion

In essence KNN classification is a supervised learning method based on statistical data. Firstly, we present the general procedure of classification algorithm. There are three factors which affect classification performance, that is the number of neighbor, distance measure and decision rule. We have analyzed different Parameter of KNN, such as different distance function. Then we have summarized several important issues and recent developments of classification problems. These include building index structure to find nearest neighbor, reducing dimension of high-dimensional dataset, and being integrated into SVM or neural network. We do not discuss problems such as feature selection, dimension, class number, which are other important and difficult problem of classification. Specifically, the Option of KNN algorithm, through the combination of the ACO study method, using edge roughness find bottlenecks, makes hierarchical state space. Tasks are divided into multiple sub tasks, division of space, decrease the learning complexity, under the premise of not to affect the solution quality of the algorithm, significantly speed up the solution.

Acknowledgments

This project is supported by National Natural Science Foundation of China (Grant No. 51375001, 51309045).

References

[1] O. Imsek and A. G. Barto, “Using relative novelty to identify useful temporal abstractions in reinforcement learning”, Proc of the 21st International Conference on Machine Learning, New York: ACM Press, **(2004)**.
[2] B. Digney, “Learning hierarchical control structures for multiple tasks and changing environments”, The 5th international conference on the Simulation of Adaptive Behavior, **(1998)**, pp. 321-330.
[3] S. Mannor and I. Menache, “Dynamic abstraction in reinforcement learning via classifying”, The 21st international conference on Machine learning, **(2004)**, pp. 71-78.

[4] S. J. Kazemitabar and H. Beigy, "Automatic discovery of subgoals in reinforcement learning using strongly connected components", ICONIP 1(5506), **(2008)**, pp. 829-834.
[5] N. Taghizadeh, "Autonomous skill acquisition in reinforcement learning based on graph classifying", Islamic Republic of Iran, Sharif University of Technology, **(2011)**.
[6] D. Erik and P. Gilbert, "Scaling Ant Colony Optimization with Hierarchical Reinforcement Learning Partitioning", The 10th annual conference on Genetic and evolutionary computation, **(2008)**, pp. 25-32.
[7] S. Berchtold, D. Keim and H. Kreigel, "The X-tree: an index structure for high dimensional data", In proceedings of the international conference on very large data based (VLDB), **(1996)**.
[8] E. Oliveira and P. Marques Ciarelli, "A Comparison Between a KNN bases Approach and a PNN Algorithm for a Multi-Label Classification Problem", Eighth International Conference on Intelligent Systems Design and Applications, **(2008)**, pp. 628-634.
[9] A. K. Jain, R. P. W. Duin and J. Mao, "Statistical Pattern Recongnition: A Review", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 1, **(2000)**, pp. 4-37.
[10] G. P. Zhang, "Neural Networks for Classification: A Survey", IEEE Transactions on Systems, Man and Cybernetics, vol. 30, no. 4, **(2000)**, pp. 451-462.

Author

Shanhong Zhu is currently a Ph.D. candidate at the School of software engineering at Wuhan University in China. And also lecturer in Xinxiang University. Her research interests include system dynamics and multimedia technology.

