n-dimensional matching algorithm

Asked 15 years, 9 months ago Modified 10 years, 8 months ago

Viewed 5k times

5

Looking for some advice here. Does anyone know a good place to start looking into matching algorithm in a n-dimensional space. For example, any dating site out there must be using some sort of algorithm to match 2 people. What I have read is that we can map characteristics of a person in a n-dimensional array with a point system for each characteristic. Once we have all (available) characteristics of a person, we can represent this person in a point within a n-dimensional array. Then, to match 2 person would be as simple as finding the shortest distance between 2 point in this n-dim array. Does anyone has any reference in implementation of these kind of algorithm? What's the best language to write these kind of stuff in?

algorithm matching n-dimensional

Share

Improve this question

Follow

asked Mar 24, 2009 at 15:41

Herman

3,024 ● 5 ● 38 ● 49

5 Answers

Sorted by: Highest score (default)

6

If you want to find the closest match for one person, Bentley & Shamos published a multi-dimensional divide-and-conquer method: Divide-and-conquer in O(N log N) time: [Divide-and-conquer in multidimensional space]() in Proceedings of the eighth annual ACM symposium on Theory of computing 1976. If you can't get a copy [this]() may also be helpful.

However for your example application actually finding the nearest neighbour doesn't seem to be the biggest problem - much trickier is mapping inputs into dimensions. For example if one dimension is "likes animals", what value do you give to someone who likes dogs & cats but can't stand horses? What about someone who loves horses, thinks dogs are OK, is annoyed by cats and is ambivalent about goldfish?

Share Improve this answer Follow

edited Mar 30, 2009 at 17:02

answered Mar 24, 2009 at 15:59

danio
8,655 ● 6 ● 49 ● 57

Good point on matching people into dimensions. Maybe something like a scale per dimension, ie: somebody who likes cats+dogs but hates horses would get +1/+1/-1, ie: +1 as a score in that dimension, or something along those lines.
– Reed Copsey Mar 24, 2009 at 16:02

@danio: You can always break a single "likes animals" dimension into separate "likes dogs", "likes cats", etc. dimensions. – j_random_hacker Mar 25, 2009 at 3:14

1

First off, pick the language you are most familiar with. The algorithms for handling this are fairly straightforward, and should work in any modern language. (As long as there is some concept of array, and potentially a matrix library, you should be fine.) I've implemented many of these in C, C++, and C# before, but seen implementations in python, vb.net, etc.

Depending on what you are trying to do, there are a couple of options.

That being said, what you want to do depends on your goals. If you just want to find the best match, you can use simple distance calculations (ie: sqrt of sum of squares for each dimension/property in the n-dimensional array), optionally weight each properties distance, and use the closest point.

If you want to group together people, you'll want to look at clustering algorithms. For data like this, I would suspect that some form of K-Means clustering or fuzzy c-means clustering would work the best.

Share Improve this answer Follow

answered Mar 24, 2009 at 15:50

Reed Copsey
564k ● 80 ● 1.2k ● 1.4k

1

How about following solution.

Assume the users are U1, U2, U3, U4, U5.... Un. Attributes are A1, A2, A3, A4, A5..... Am

Store these as

A1 - U1, U2, U3... A2 - U4, U6, U7.... A3 -

Profile attribute is the index and stores all users. Now, if a new User comes, see the its attributes and for those attributes, find common people. number of times a person exists in these lists - higher ranking.

Share Improve this answer Follow

answered May 7, 2012 at 5:46

Thoughtful Monkey
668 ● 2 ● 10 ● 24

0

The process you mention is known as k-nearest neighbor, with k=1. It's the most intuitive approach for finding similar vectors.

http://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm

Share Improve this answer Follow

answered Apr 1, 2009 at 11:48

rjprins

k-nearest neighbor is a classification algorithm, not what Herman is looking for. – Fifi Oct 3, 2019 at 15:58

0

What you describe with your example is not n-dimentional matching, but rather bipartite matching of nodes with multiple features. (You will need to provide a function that will, given two persons compute this distance). There should be very efficient algorithms for that. In n-dimentional matching you would try to match nodes from more than two sets (in your example, suppose you could cut people to body, soul, and music preferences, then recombine them to make new persons. Then the n-dimentional matching would cut the people apart, and recombine them so that the new persons engineered would make really nice couples :D) Here is the wikipedia article for 3-dimentional matching, which is np-complete.

Also, as noted by another, if your goal is not to match persons in pairs, but rather find compatible groups, you should consider clustering them to groups. This can be done with e.g. Unsupervised Learning

Share Improve this answer Follow

edited Apr 2, 2014 at 13:52

answered Apr 2, 2014 at 13:45

ntg

14k ● 9 ● 80 ● 103