EM optimisation

It is necessary to optimise calculation of distances for EM and projections onto faces for MapGeometry.

# Distance calculation optimisation

Code of distance calculation is very short:

function [dist, klas] = associate(~, node, data)

%associate identify the nearest node for each data point and

%return the squared distance between selected node and data

%point and number of nearest node.

%

%Inputs:

% node is n-by-k matrix of mapped coordinates for tested state

% of map, where n is number of nodes and m is dimension of

% data space.

% data is m-by-k data points to test, where m is number of

% points and k is dimension of data space.

%

%Outputs:

% dist is m-by-1 matrix of squared distances from data point to

% nearest node

% klass is m-by-1 vector which contains number of nearest node

% for each data point.

dist = bsxfun(@plus, sum(data .^ 2, 2),...

sum(node .^2, 2)') - 2 \* (data \* node');

[dist, klas] = min(dist, [], 2);

end

I created two more functions

function [dist, klas] = associate1(~, node, data)

%associate identify the nearest node for each data point and

%return the squared distance between selected node and data

%point and number of nearest node.

%

%Inputs:

% node is n-by-k matrix of mapped coordinates for tested state

% of map, where n is number of nodes and m is dimension of

% data space.

% data is m-by-k data points to test, where m is number of

% points and k is dimension of data space.

%

%Outputs:

% dist is m-by-1 matrix of squared distances from data point to

% nearest node

% klass is m-by-1 vector which contains number of nearest node

% for each data point.

dist = bsxfun(@plus, sum(data .^ 2, 2),...

sum(node .^2, 2)') - 2 \* (data \* node');

[dist, klas] = min(dist, [], 2);

end

and

function [dist, klas] = associate2(~, node, data)

%associate identify the nearest node for each data point and

%return the squared distance between selected node and data

%point and number of nearest node.

%

%Inputs:

% node is n-by-k matrix of mapped coordinates for tested state

% of map, where n is number of nodes and m is dimension of

% data space.

% data is m-by-k data points to test, where m is number of

% points and k is dimension of data space.

%

%Outputs:

% dist is m-by-1 matrix of squared distances from data point to

% nearest node

% klass is m-by-1 vector which contains number of nearest node

% for each data point.

dist = bsxfun(@plus, sum(data .^ 2, 2)',...

sum(node .^2, 2)) - 2 \* (node \* data');

[dist, klas] = min(dist);

dist = dist';

klas = klas';

end

Function associate is method of class MapGeometry.

Function associate1 is exactly the same as associate but is not method of class.

Function associate2 produces the same calculations but in transposed form.

I produces the following test method to compare performance of these functions

nRep = 100;

result = zeros(nRep, 3);

for k = 1:nRep

tic;

[dist, ass] = associate(map, nodes, data);

result(k, 1) = toc;

tic;

[dist, ass] = associate1(map, nodes, data);

result(k, 2) = toc;

tic;

[dist, ass] = associate2(map, nodes, data);

result(k, 3) = toc;

end

The result of work is saved in variable “result”.

Results of statistical test are presented in table below.

|  |  |  |  |
| --- | --- | --- | --- |
| Indicator | associate | associate1 | associate2 |
| Mean time | 4.7070 | 4.7038 | 4.5832 |
| t-test: SSD from | | | |
| associate |  | 0.9563 | 0.0431 |
| associate1 | 0.9563 |  | 0.0459 |
| associate2 | 0.0431 | 0.0459 |  |
| Mann Whitney U test: SSD from | | | |
| associate |  | 0.8921 | 0.0039 |
| associate1 | 0.8921 |  | 0.0019 |
| associate2 | 0.0039 | 0.0019 |  |

The first conclusion: the transpose of calculation can improve performance.

Difference in performance between method and simple function is negligible.

# Protection from memory error

The next step is usage of two more functions with memory overflow protection: associate3 and associate4. These functions are analogue of associate1 but with memory protection. Association4 also transpose matrix node ones.

Results of testing is presented in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| Indicator | associate1 | associate3 | associate4 |
| Mean time | 4.6118 | 4.8009 | 4.7520 |
| t-test: SSD from | | | |
| associate1 |  | <0.0001 | 0.0018 |
| associate3 | <0.0001 |  | 0.2104 |
| associate4 | 0.0018 | 0.2104 |  |
| Wilcoxon signed rank test: SSD from | | | |
| associate1 |  | <0.0001 | 0.0001 |
| associate3 | <0.0001 |  | 0.0201 |
| associate4 | 0.0001 | 0.0201 |  |

This means that protected version has time with SSD from unprotected one. Protected version is slower. Optimisation with unique transposition is slightly faster but there is no SSD.

The next step is test with different maximal size of array from 8Mb to 4Gb. Test will be performed for the associate4.

# Test of differences caused by the array size

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Length of array | | | |
| Indicator | 8M | 80M | 800M | 1600M |
| Mean time | 5.6514 | 4.8042 | 4.7465 | 4.7278 |
| t-test: SSD from | | | |  |
| 8M |  | <0.0001 | <0.0001 | <0.0001 |
| 80M | <0.0001 |  | 0.2036 | 0.0259 |
| 800M | <0.0001 | 0.2036 |  | 0.6928 |
| 1600M | <0.0001 | 0.0259 | 0.6928 |  |
| Mann Whitney U test: SSD from | | | |  |
| 8M |  | <0.0001 | <0.0001 | <0.0001 |
| 80M | <0.0001 |  | 0.0226 | 0.1034 |
| 800M | <0.0001 | 0.0226 |  | 0.5292 |
| 1600M | <0.0001 | 0.1034 | 0.5292 |  |

As we can see, 8M model has time SSD from all other sizes. Difference in time between 80M and 800M is significant in level 97% (MWU test). Difference between 800M and 1600M is not SS. The best choice looks like 800M.

# The transposed version time comparison

| Indicator | Length of array | | | |
| --- | --- | --- | --- | --- |
| 8M | 80M | 800M | 1600M |
| Mean time | 5.4016 | 4.9880 | 4.5946 | 4.5549 |
| t-test: SSD from | | | |  |
| 8M |  | <0.0001 | <0.0001 | <0.0001 |
| 80M | <0.0001 |  | <0.0001 | <0.0001 |
| 800M | <0.0001 | <0.0001 |  | 0.4373 |
| 1600M | <0.0001 | <0.0001 | 0.4373 |  |
| Mann Whitney U test: SSD from | | | |  |
| 8M |  | <0.0001 | <0.0001 | <0.0001 |
| 80M | <0.0001 |  | <0.0001 | <0.0001 |
| 800M | <0.0001 | <0.0001 |  | 0.2657 |
| 1600M | <0.0001 | <0.0001 | 0.2657 |  |

As we can see, 8M and 80M model has time SSD from all other sizes. Difference in time between 80M and 800M is significant in level 97% (MWU test). Difference between 800M and 1600M is not SS. The best choice looks like 800M.

# Final comparison

|  |  |  |  |
| --- | --- | --- | --- |
| Indicator | associate | associate4 (800M) | associate5 (800M) |
| Mean time | 4.7070 | 4.7465 | 4.5946 |
| t-test: SSD from | | | |
| associate |  | 0.4914 | 0.0444 |
| associate4 (800M) | 0.4914 |  | 0.0054 |
| associate5 (800M) | 0.0444 | 0.0054 |  |
| Mann Whitney U test: SSD from | | | |
| associate |  | 0.1176 | 0.0779 |
| associate4 (800M) | 0.1176 |  | 0.0036 |
| associate5 (800M) | 0.0779 | 0.0036 |  |

We can see that associate5 is SS faster than two other functions. This function is used for EM.