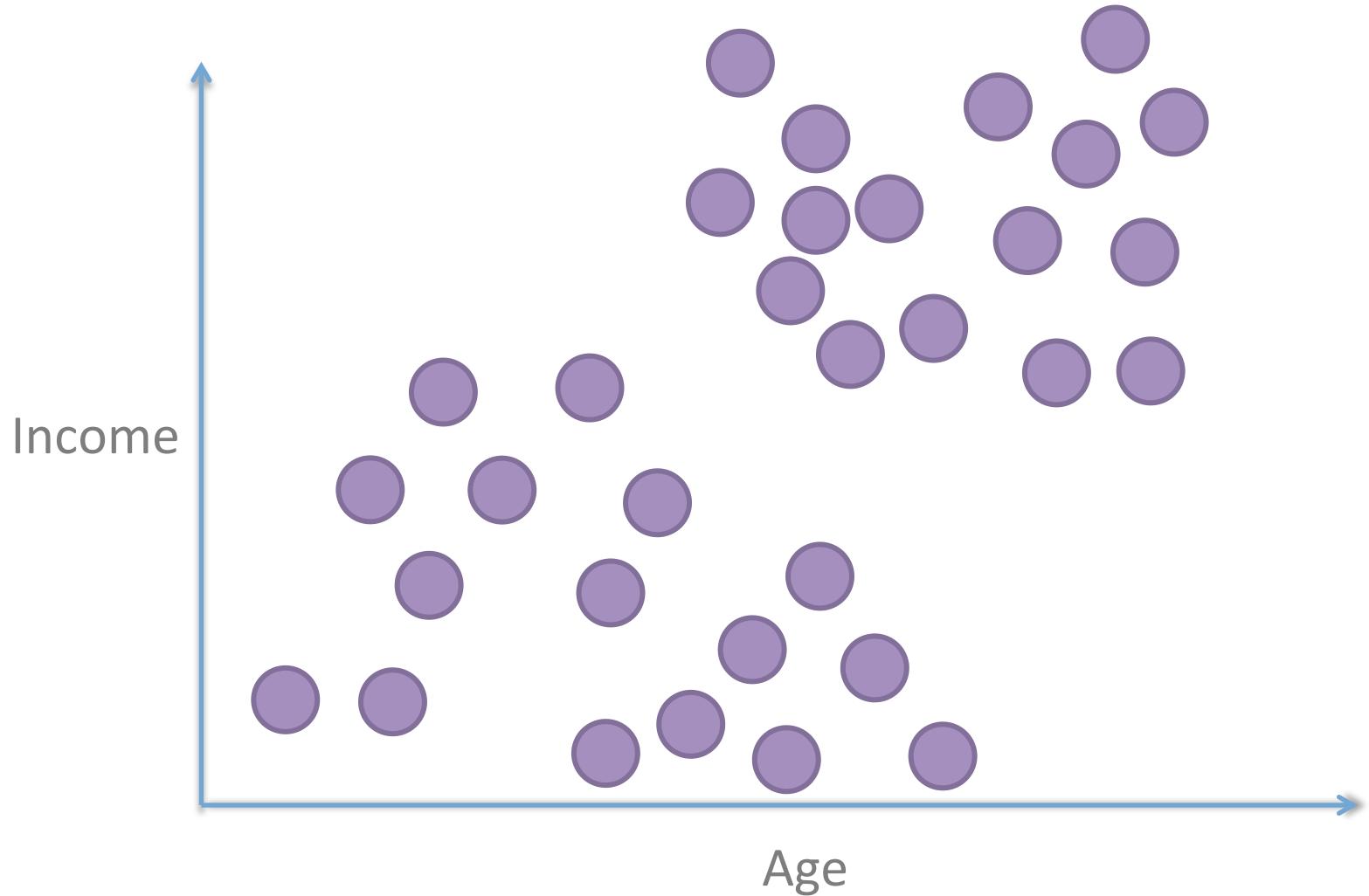


Other Clustering Algorithms

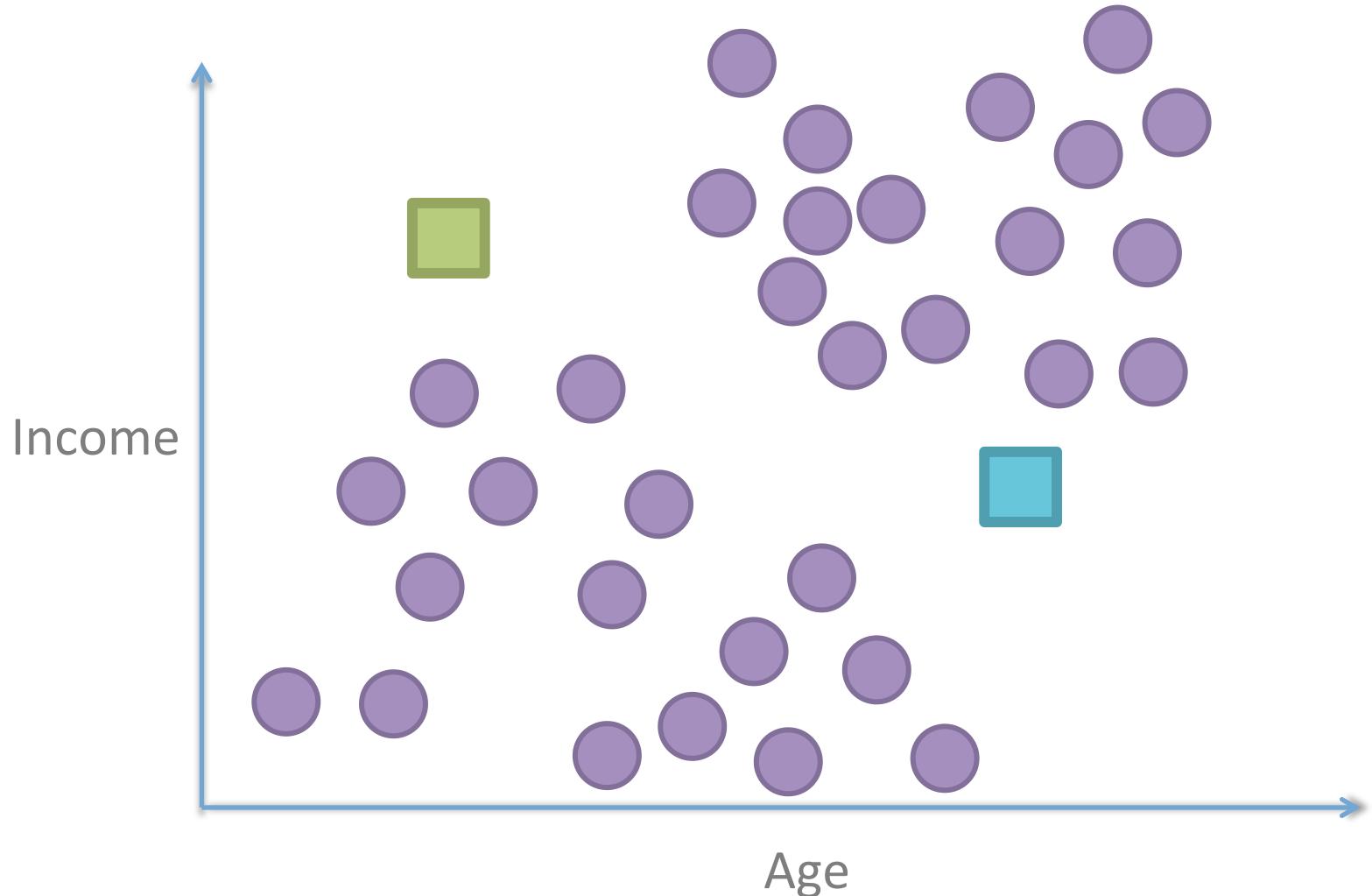


Remember K--Means?



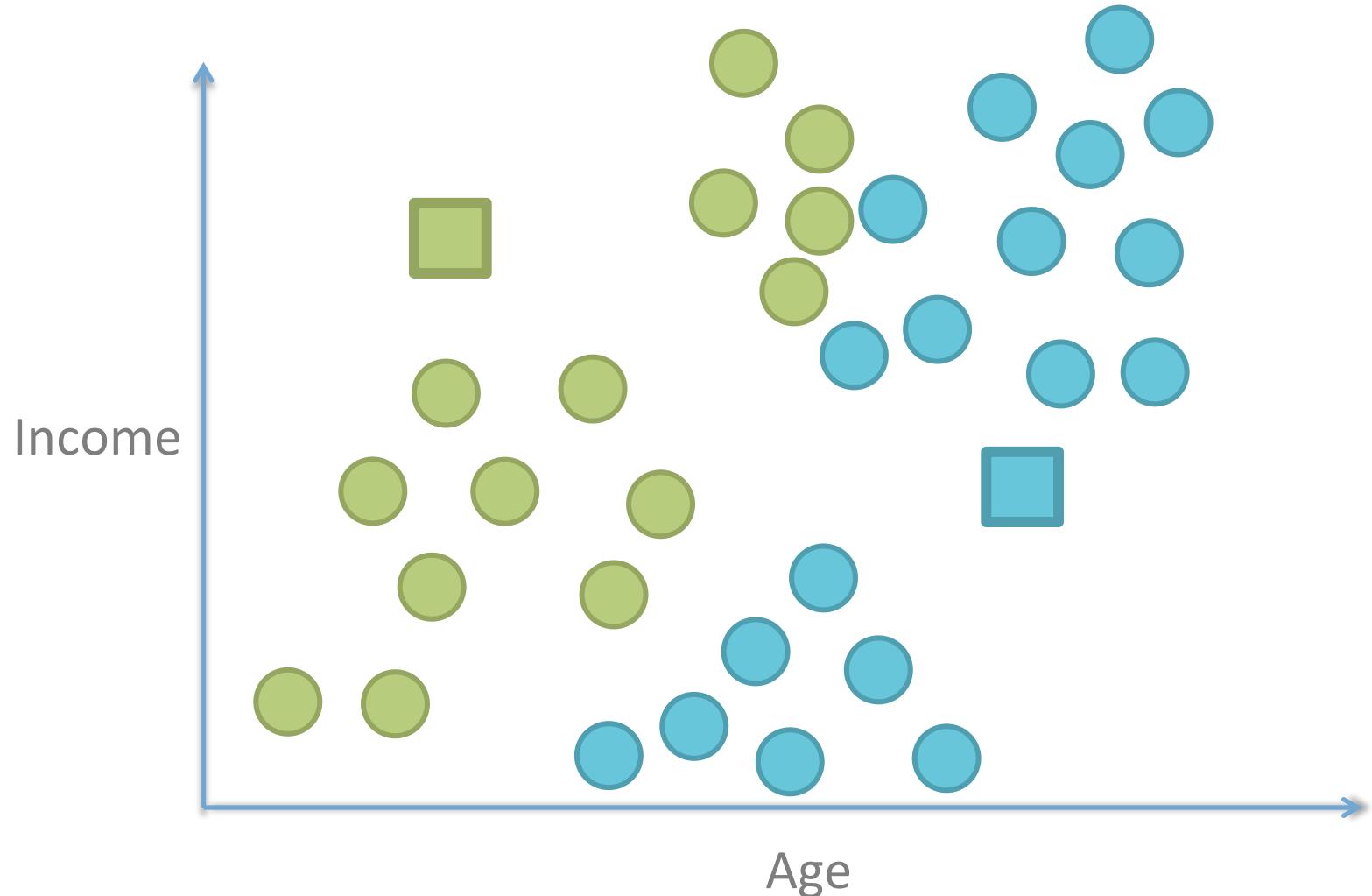
K-Means K=2

Randomly assign two cluster centers



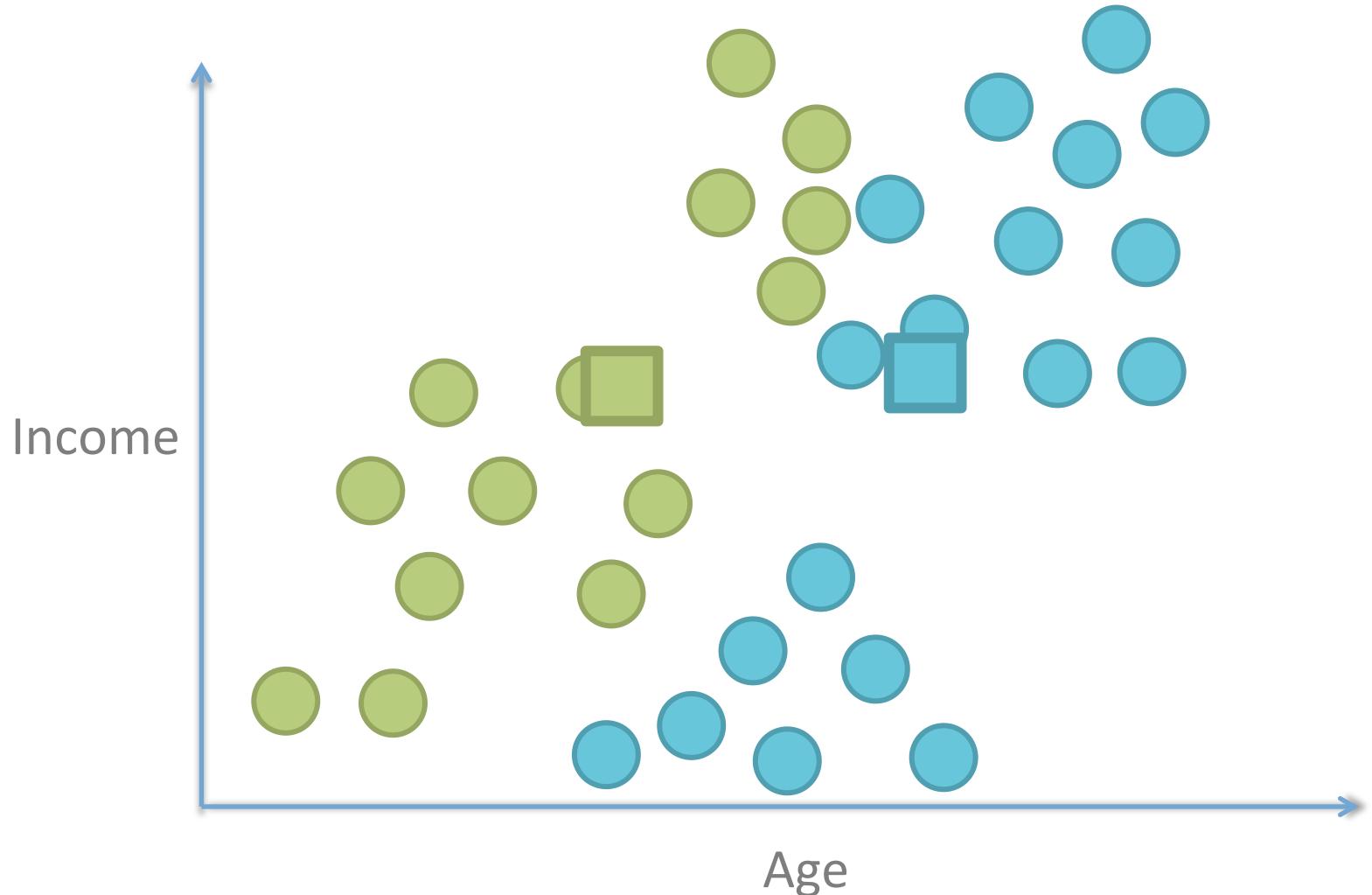
K-Means K=2

Each point belongs to closest center



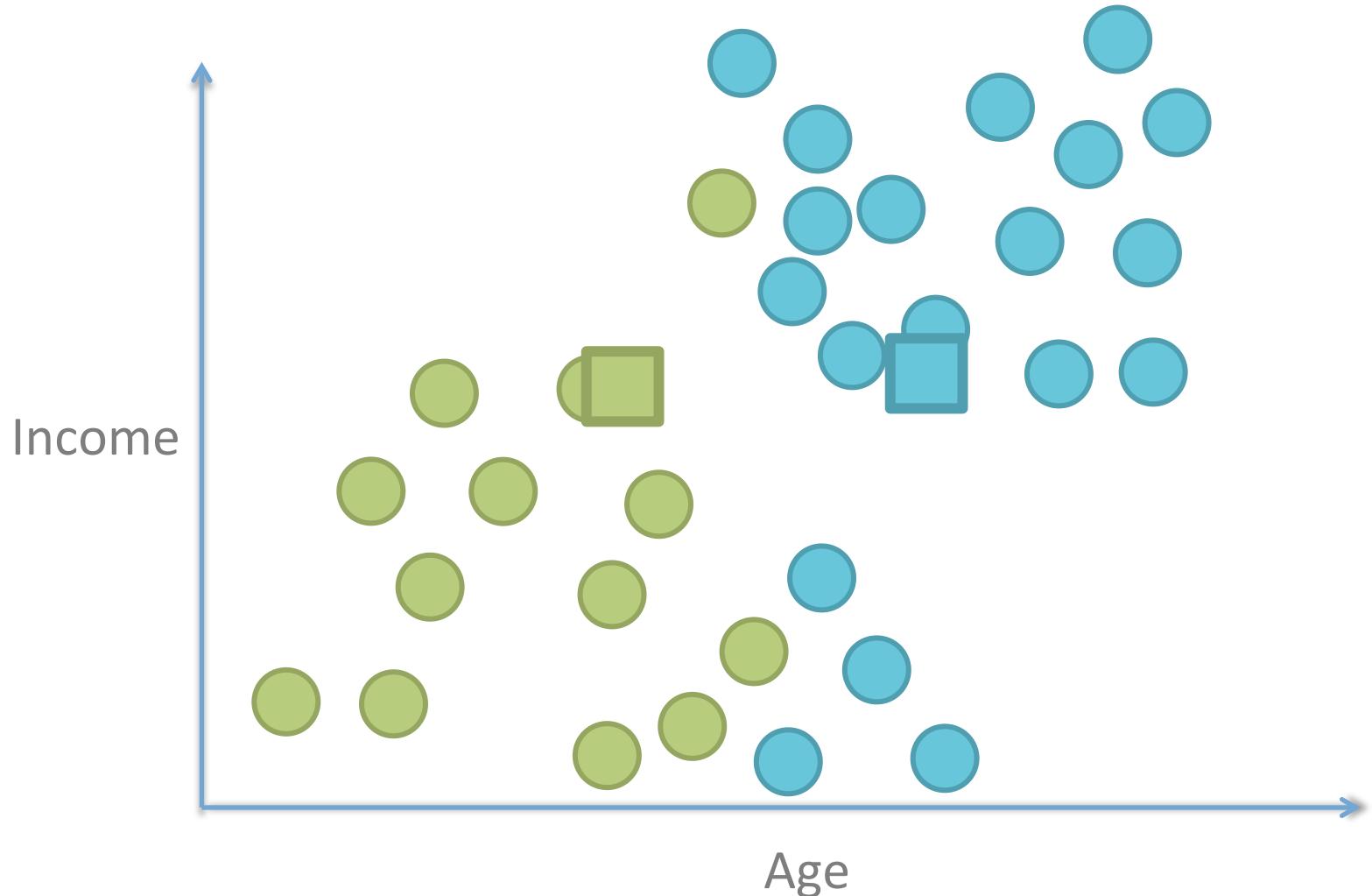
K-Means K=2

Move each center to the cluster's mean



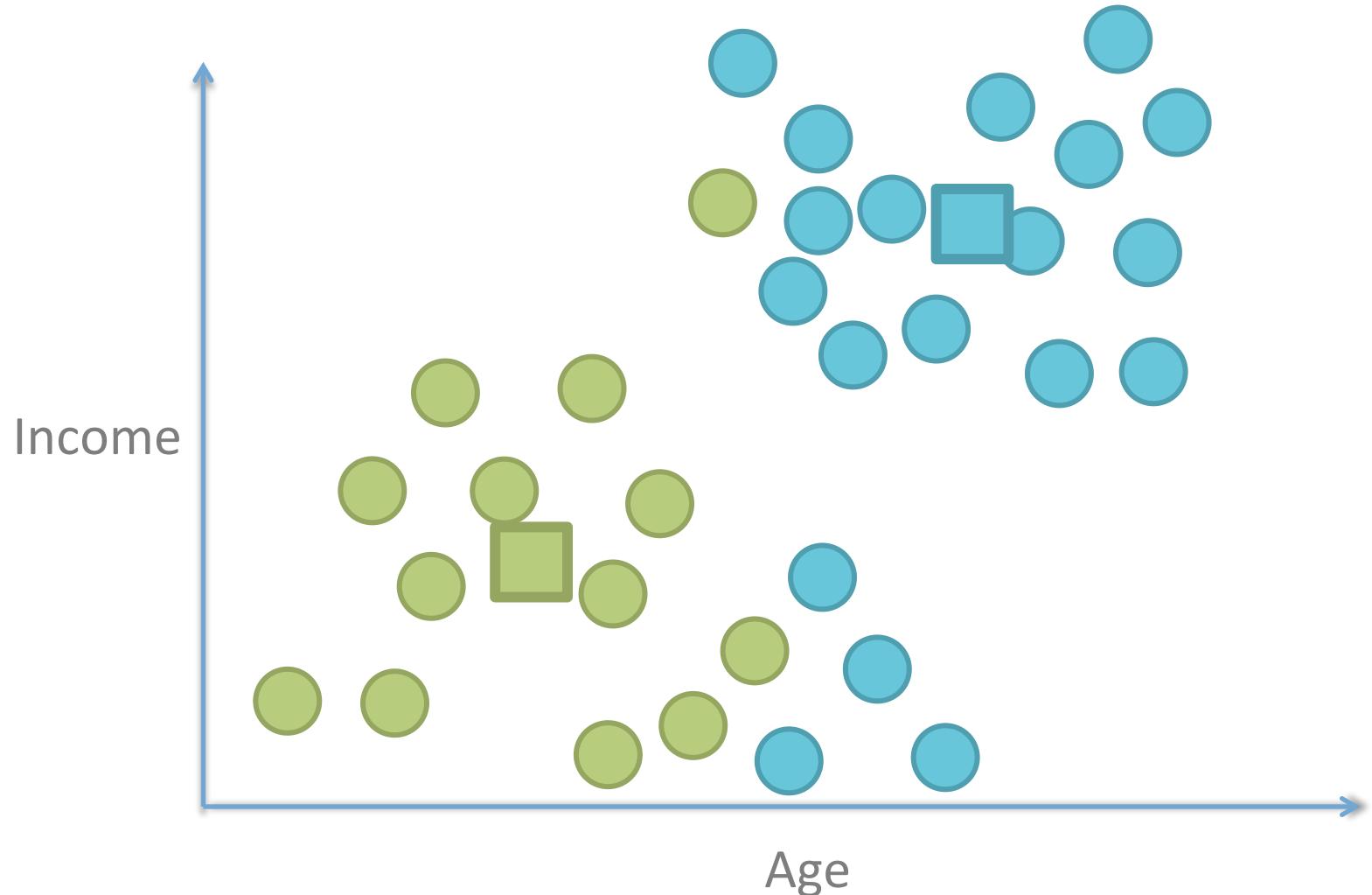
K-Means K=2

Each point belongs to the closest center



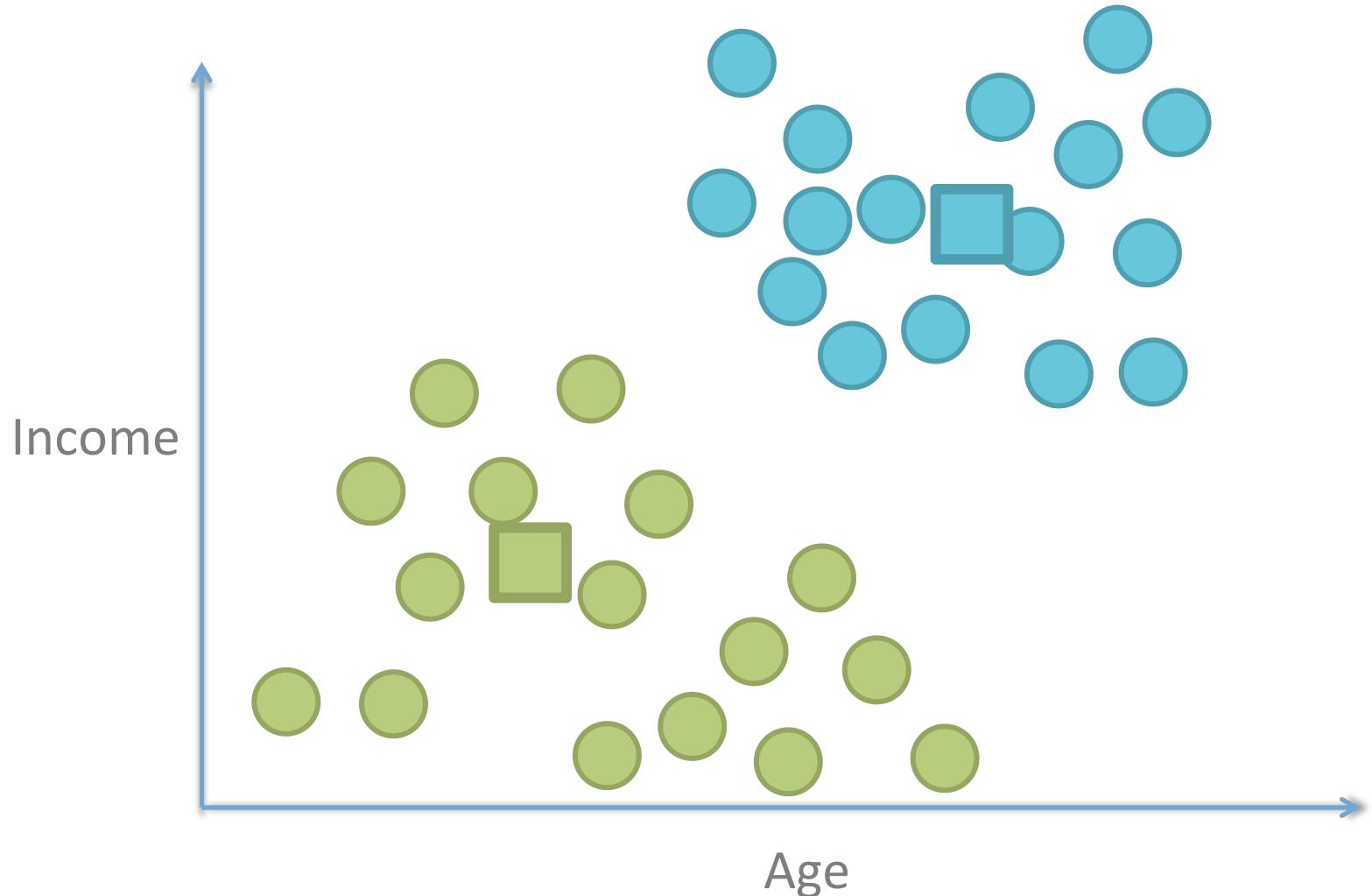
K-Means K=2

Move each center to the cluster's mean



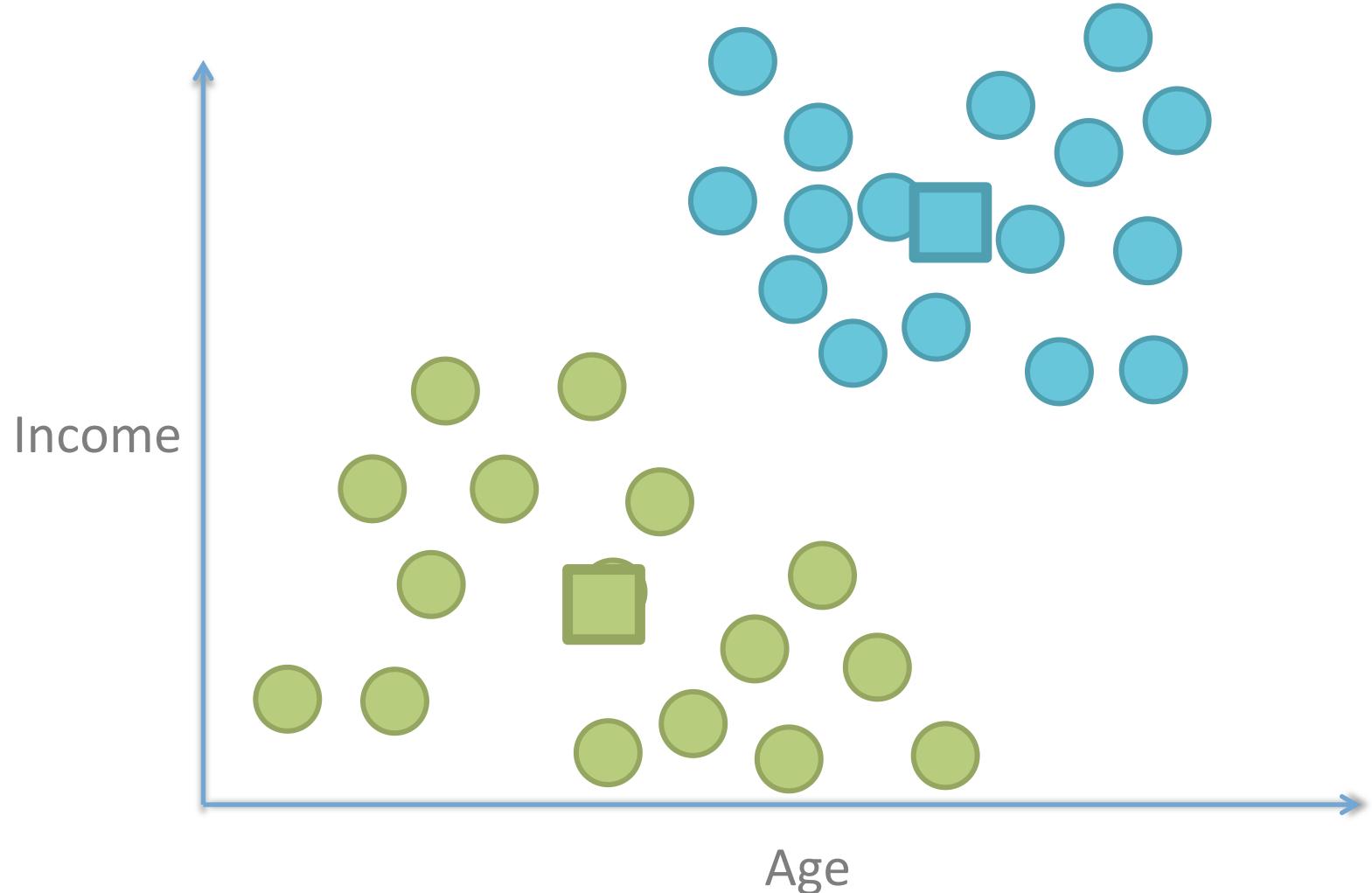
K-Means K=2

Each point belongs to closest center



K-Means K=2

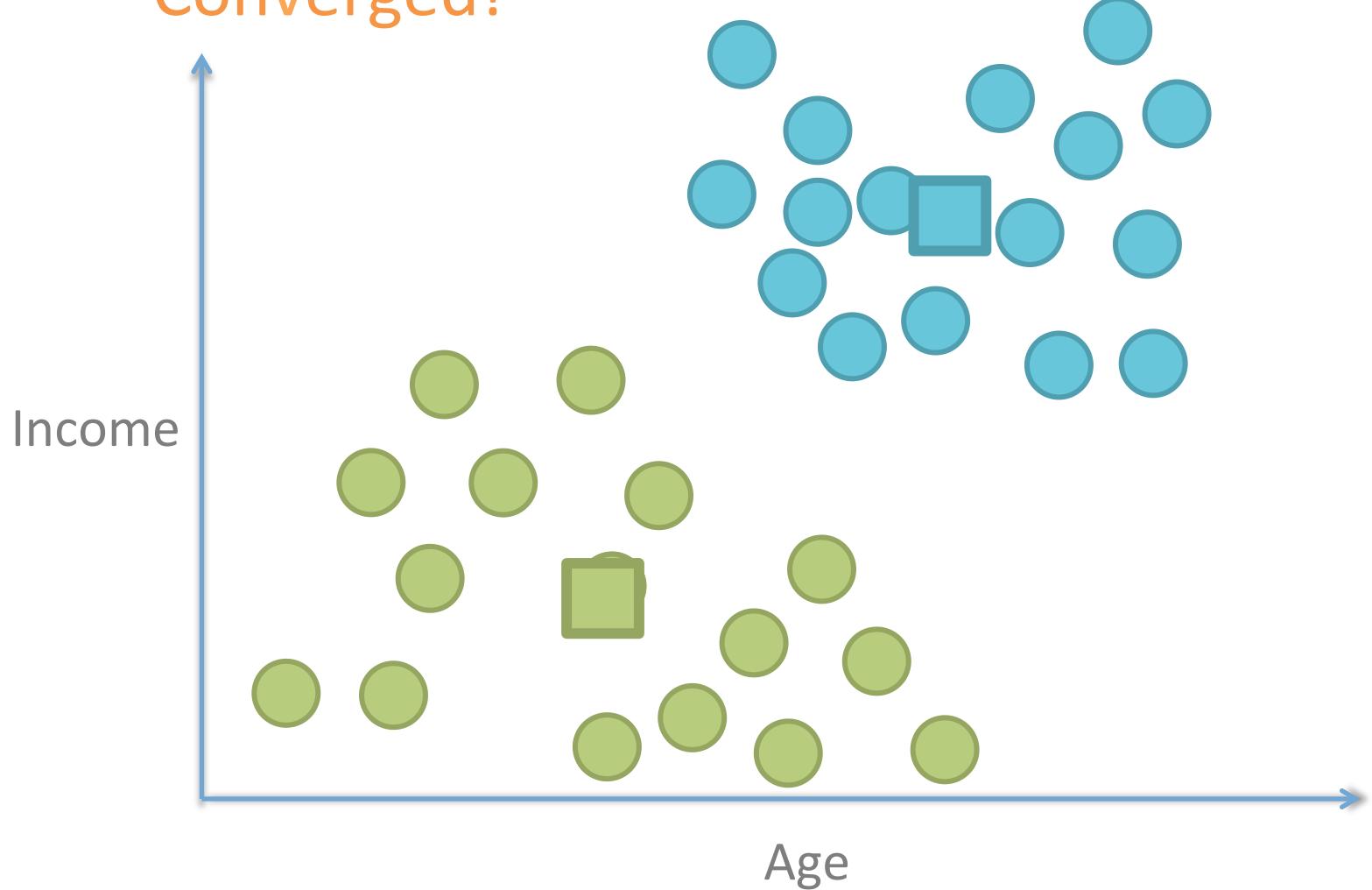
Move each center to the cluster's mean



K-Means K=2

Points don't change anymore.

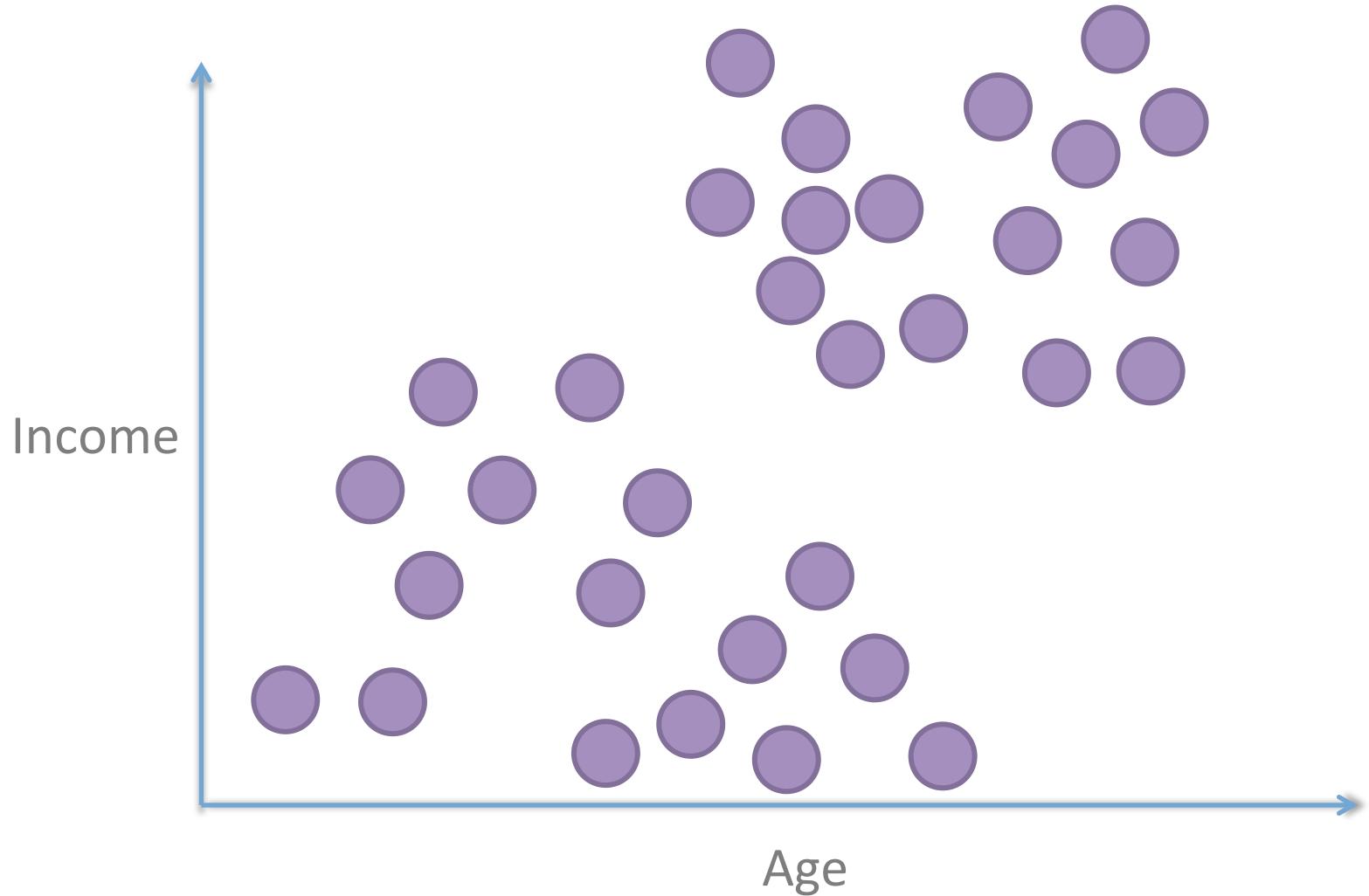
Converged!



Hierarchical Agglomerative Clustering

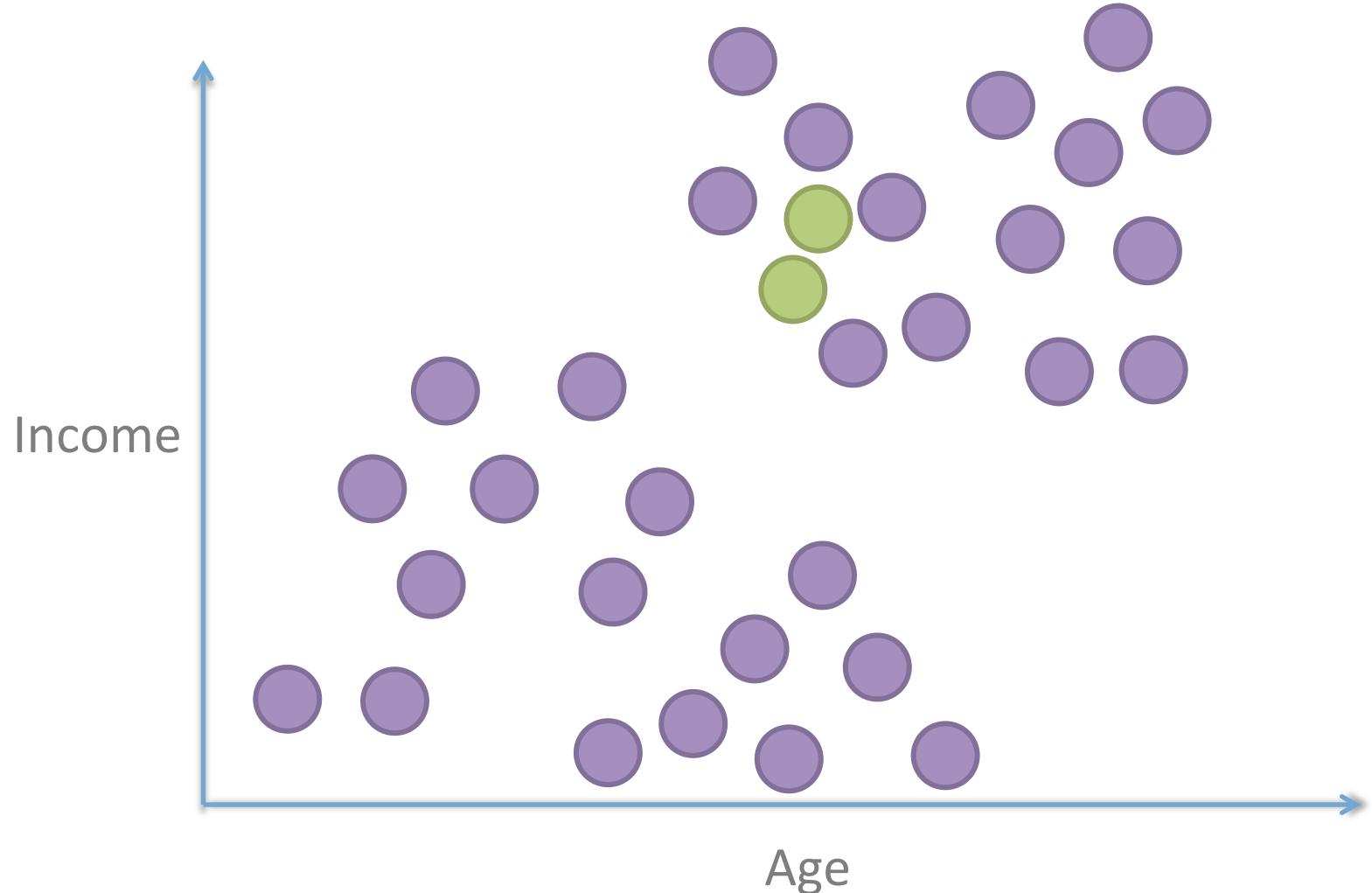


Hierarchical Agglomerative Clustering



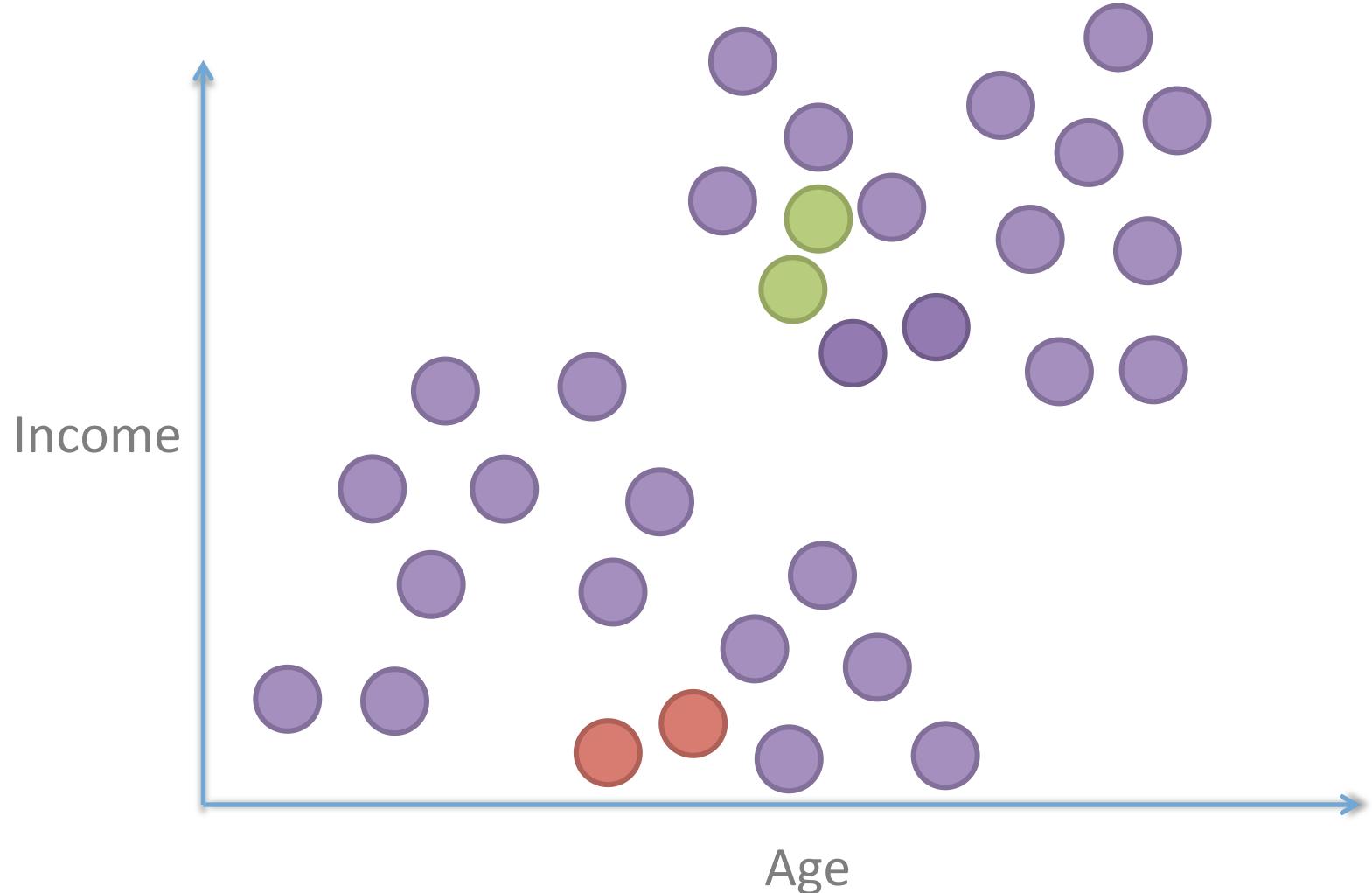
Agglomerative Hierarchical Clustering

Find closet pair, merge into a cluster



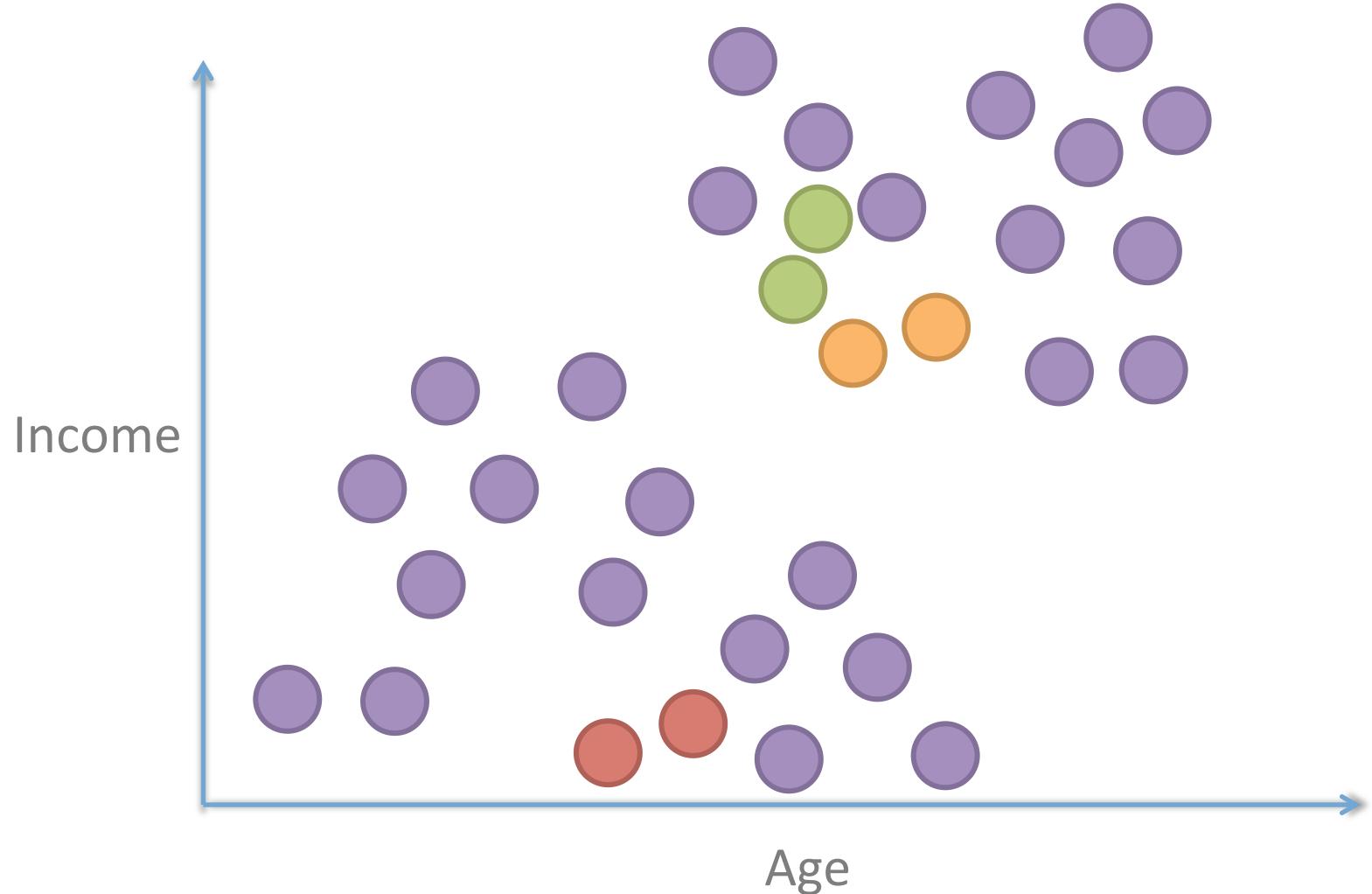
Agglomerative Hierarchical Clustering

Find next closet pair, merge



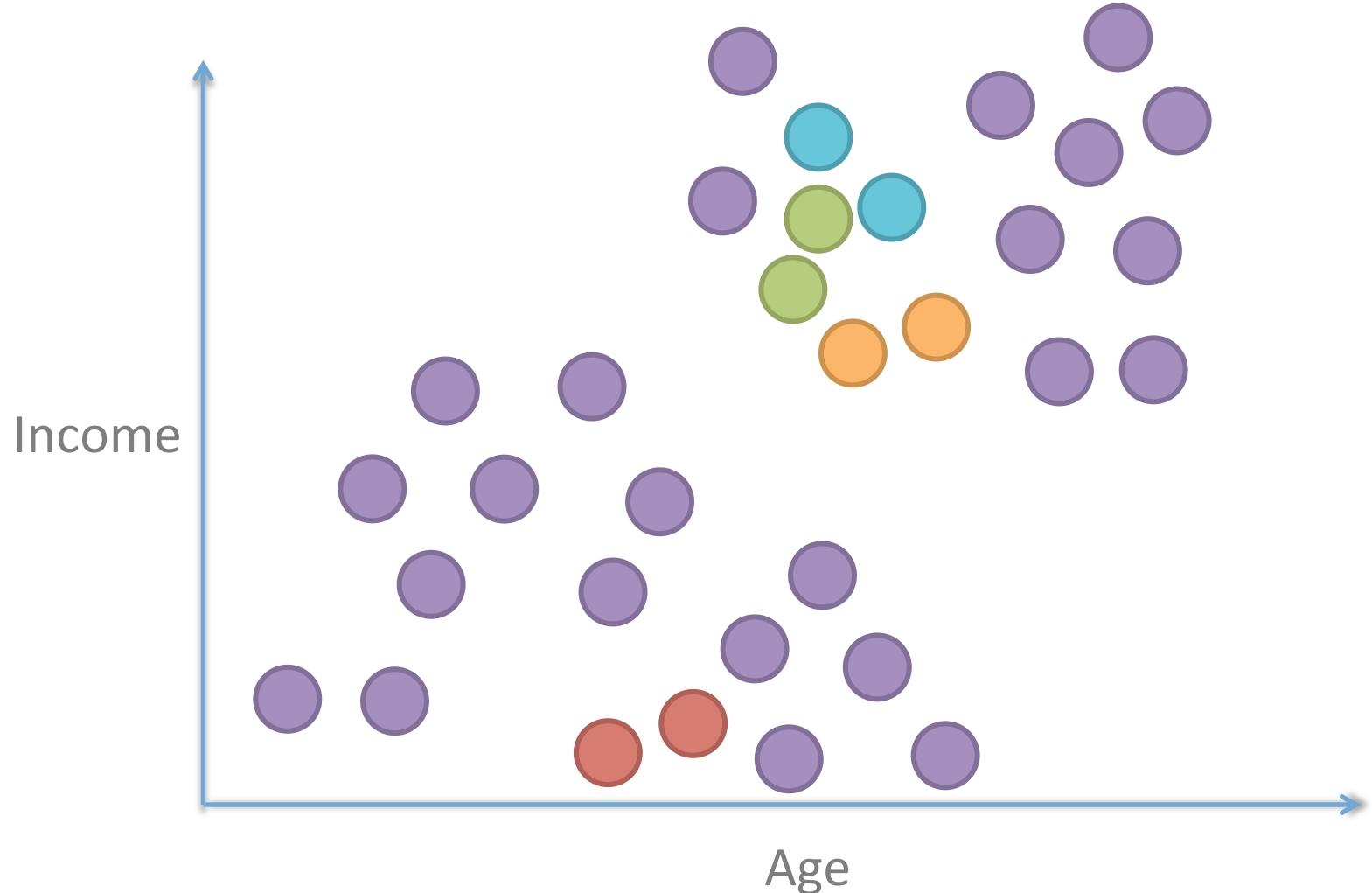
Agglomerative Hierarchical Clustering

Find next closet pair, merge



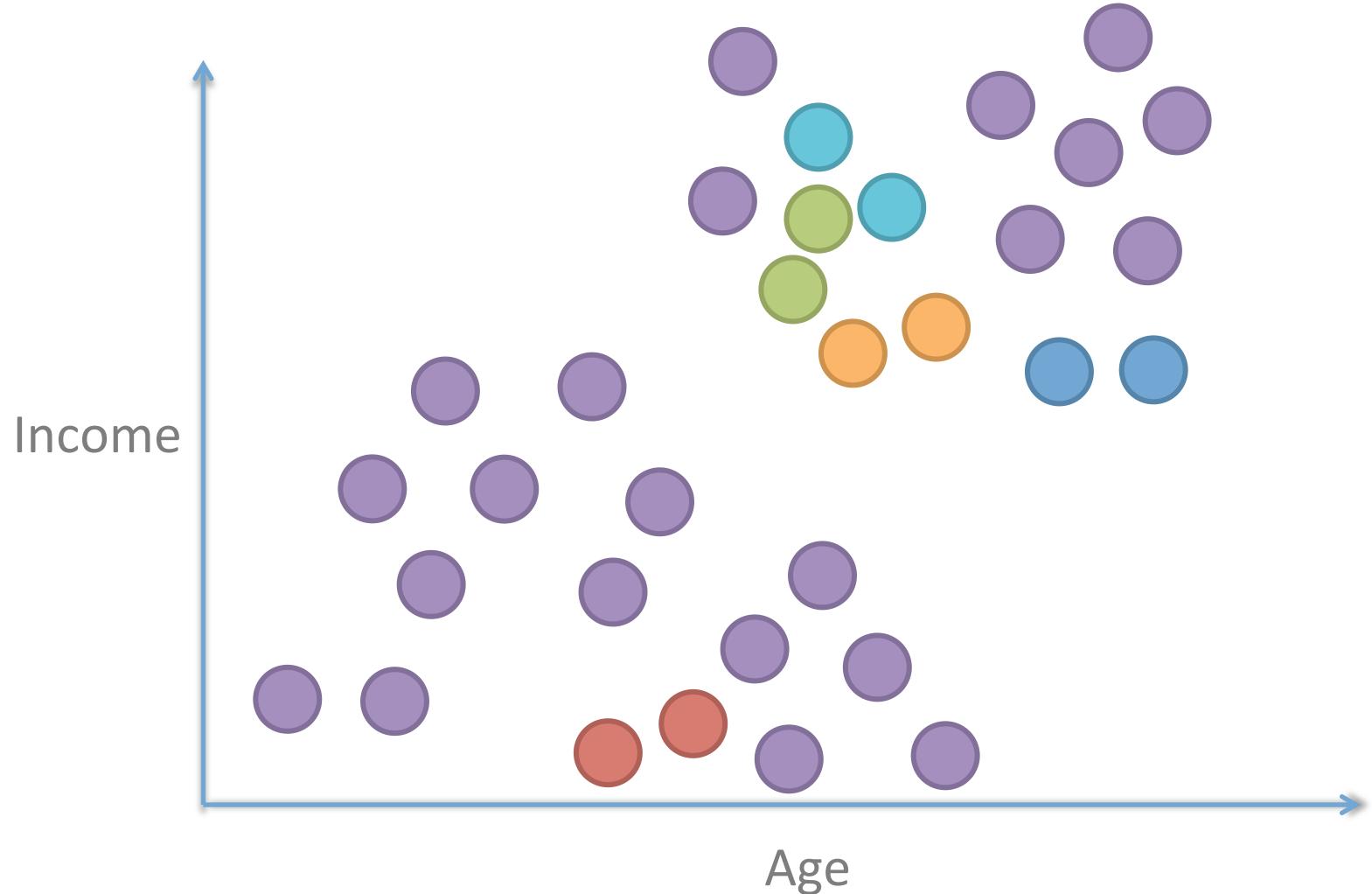
Agglomerative Hierarchical Clustering

Find next closet pair, merge



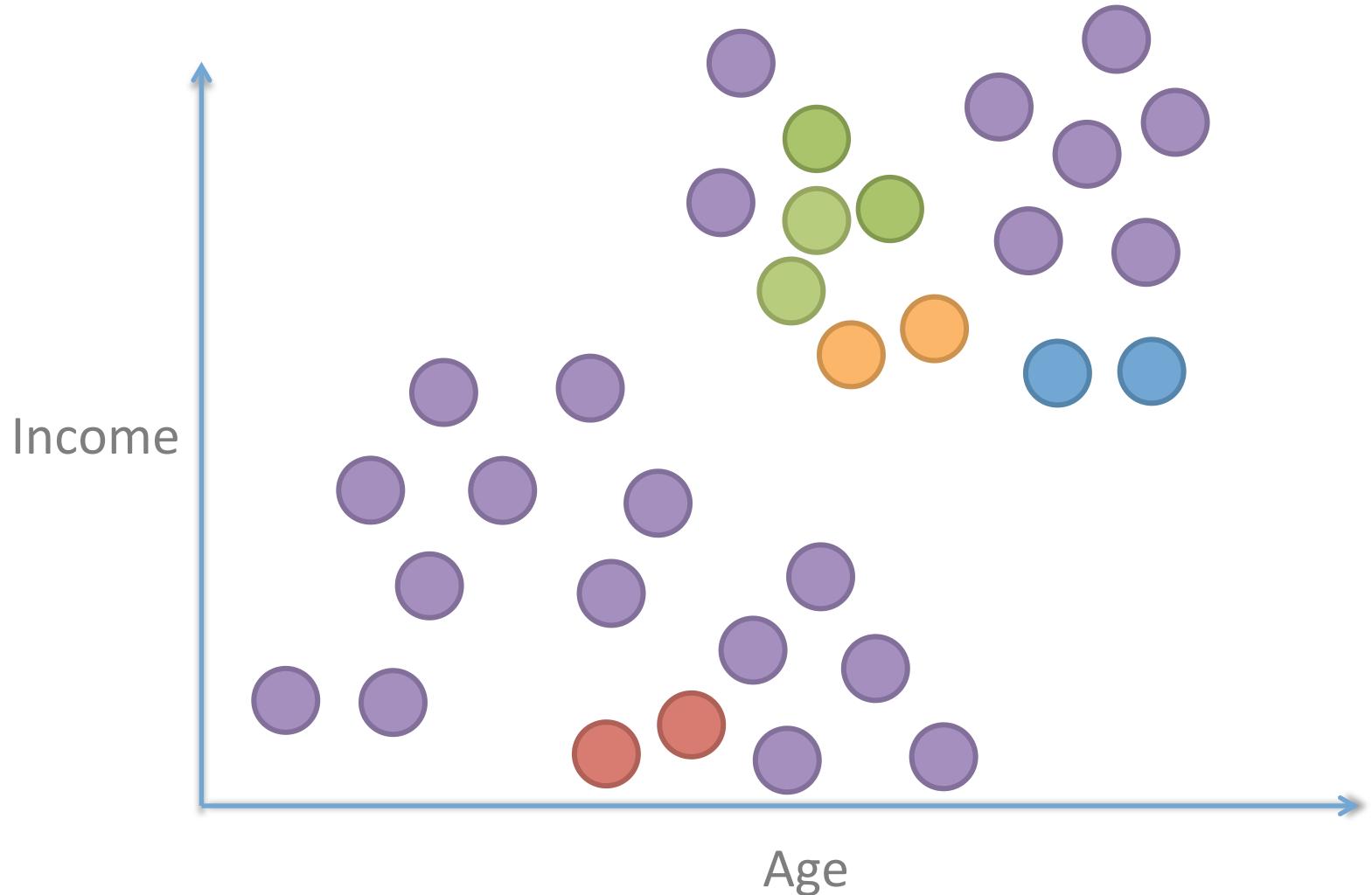
Agglomerative Hierarchical Clustering

Find next closet pair, merge



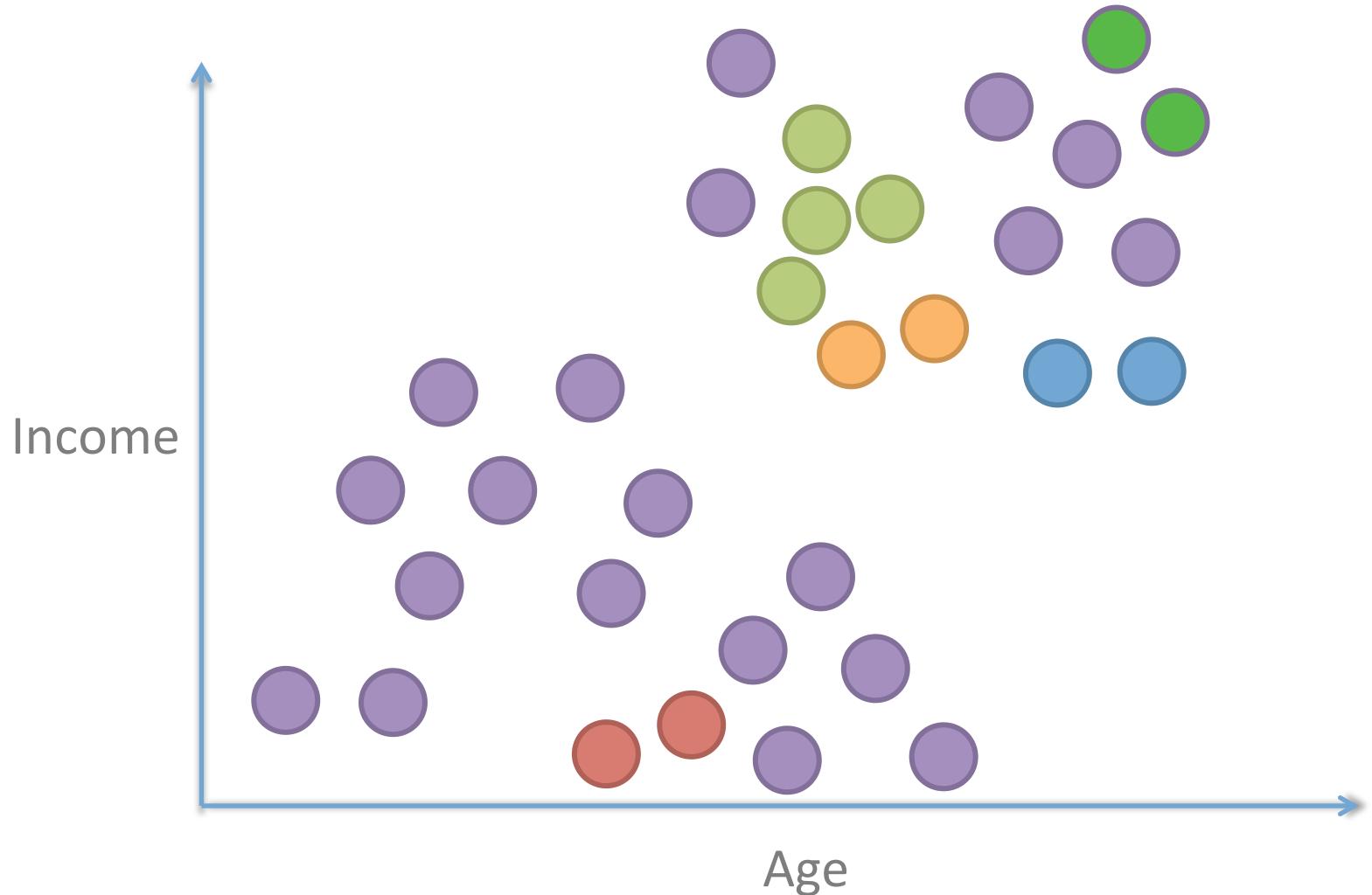
Agglomerative Hierarchical Clustering

If closest pair are clusters, same: merge



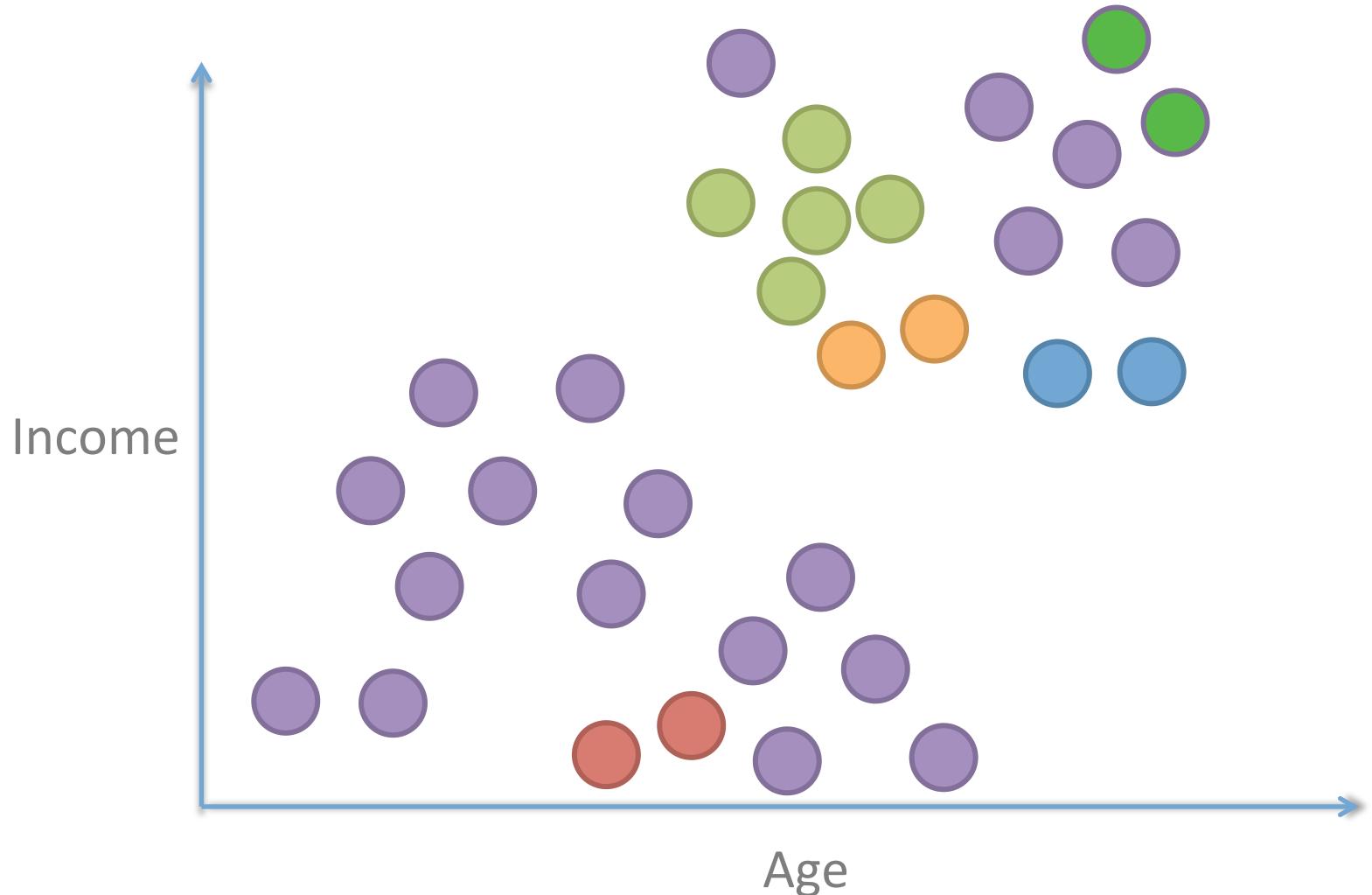
Agglomerative Hierarchical Clustering

If closest pair are clusters, same: merge



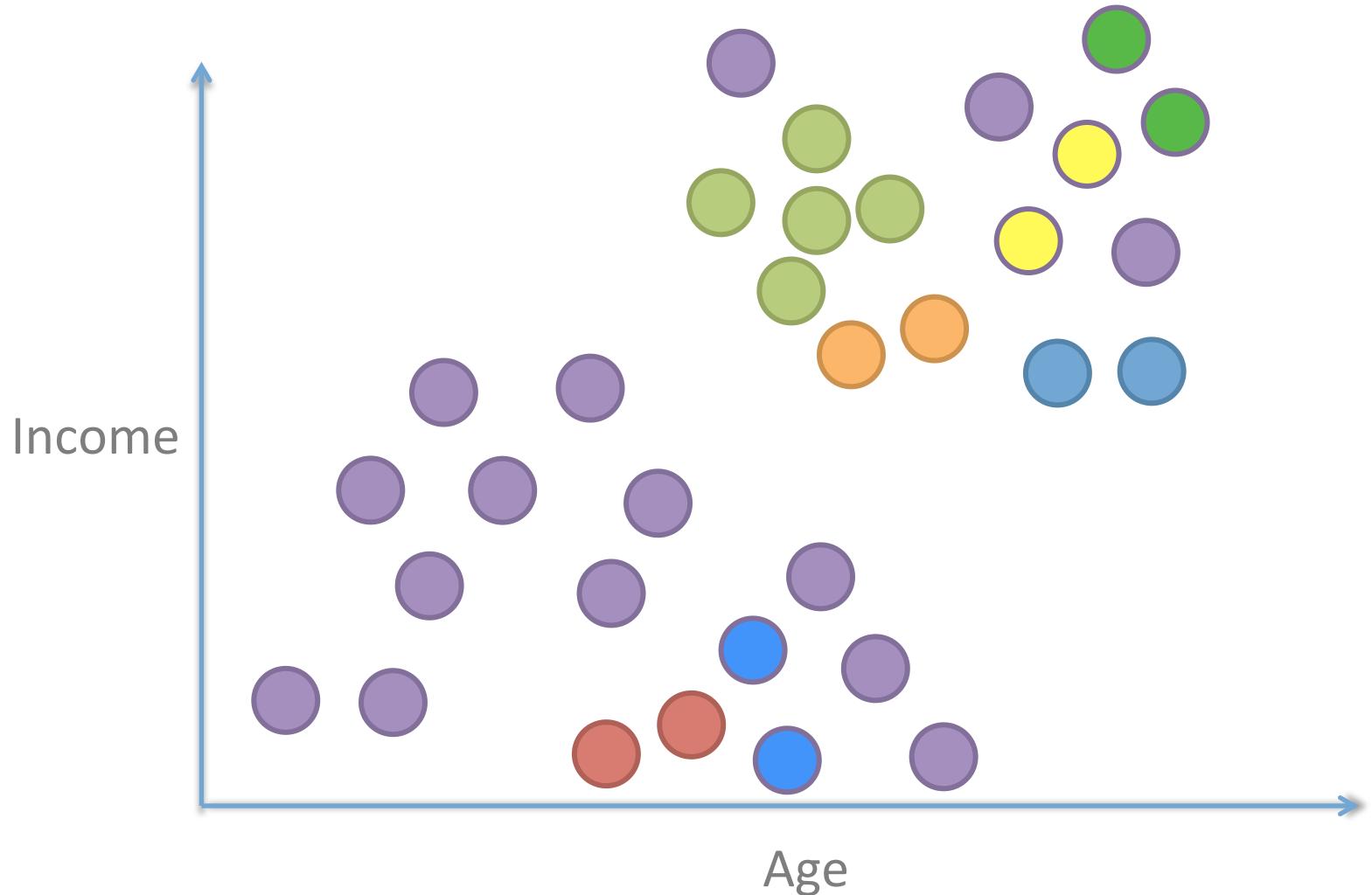
Agglomerative Hierarchical Clustering

If closest pair are clusters, same: merge



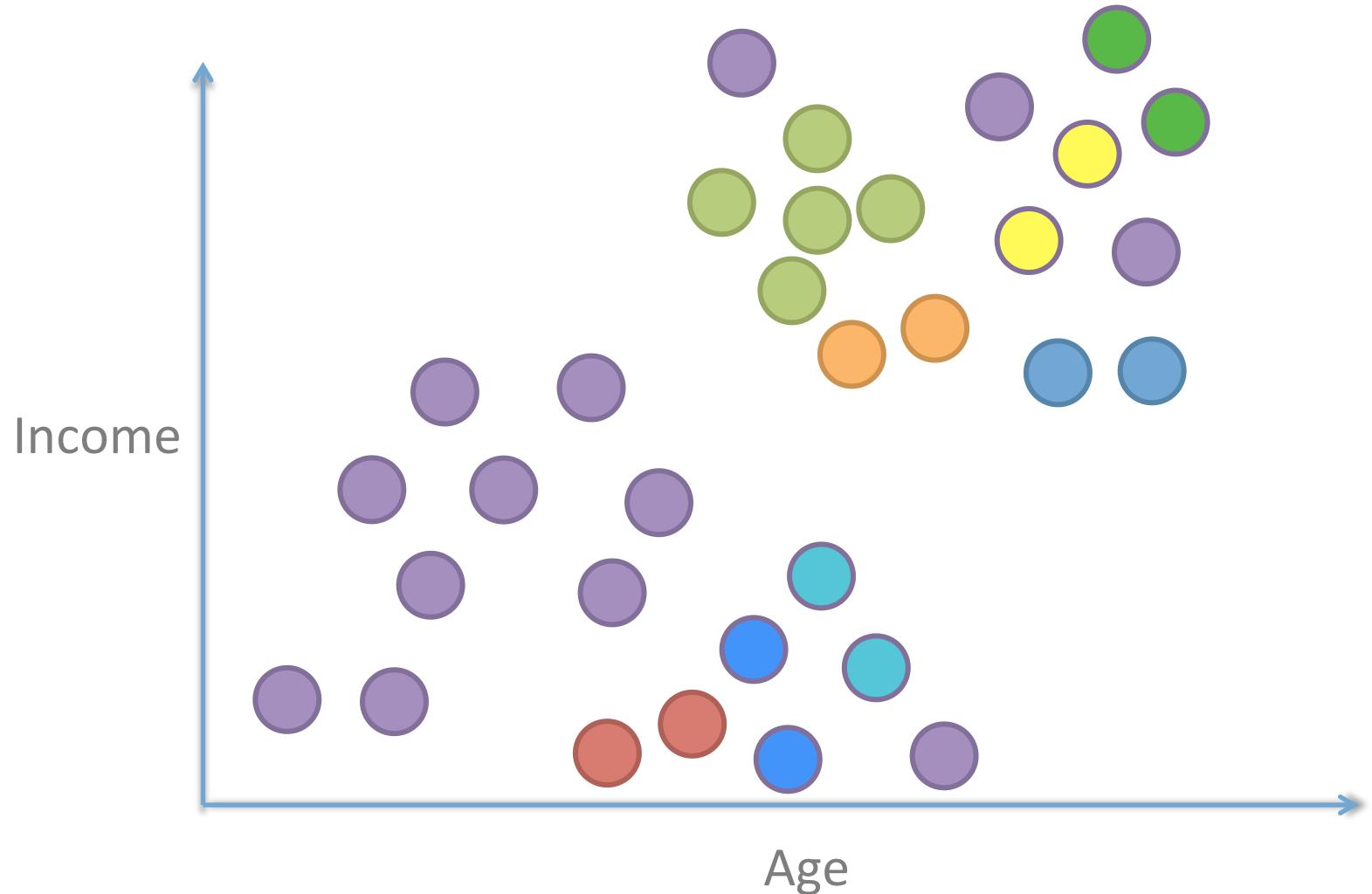
Agglomerative Hierarchical Clustering

If closest pair are clusters, same: merge



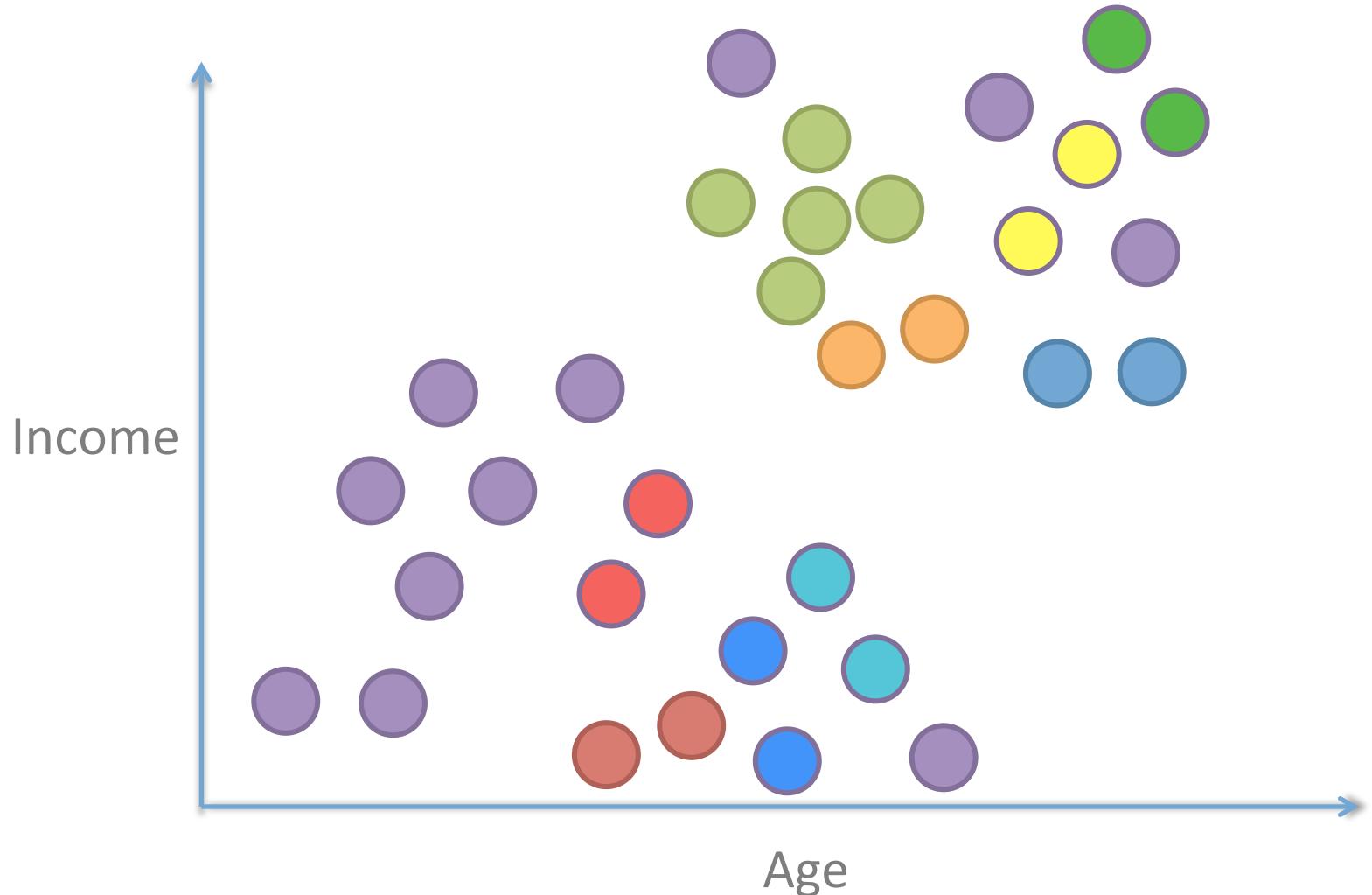
Agglomerative Hierarchical Clustering

If closest pair are clusters, same: merge



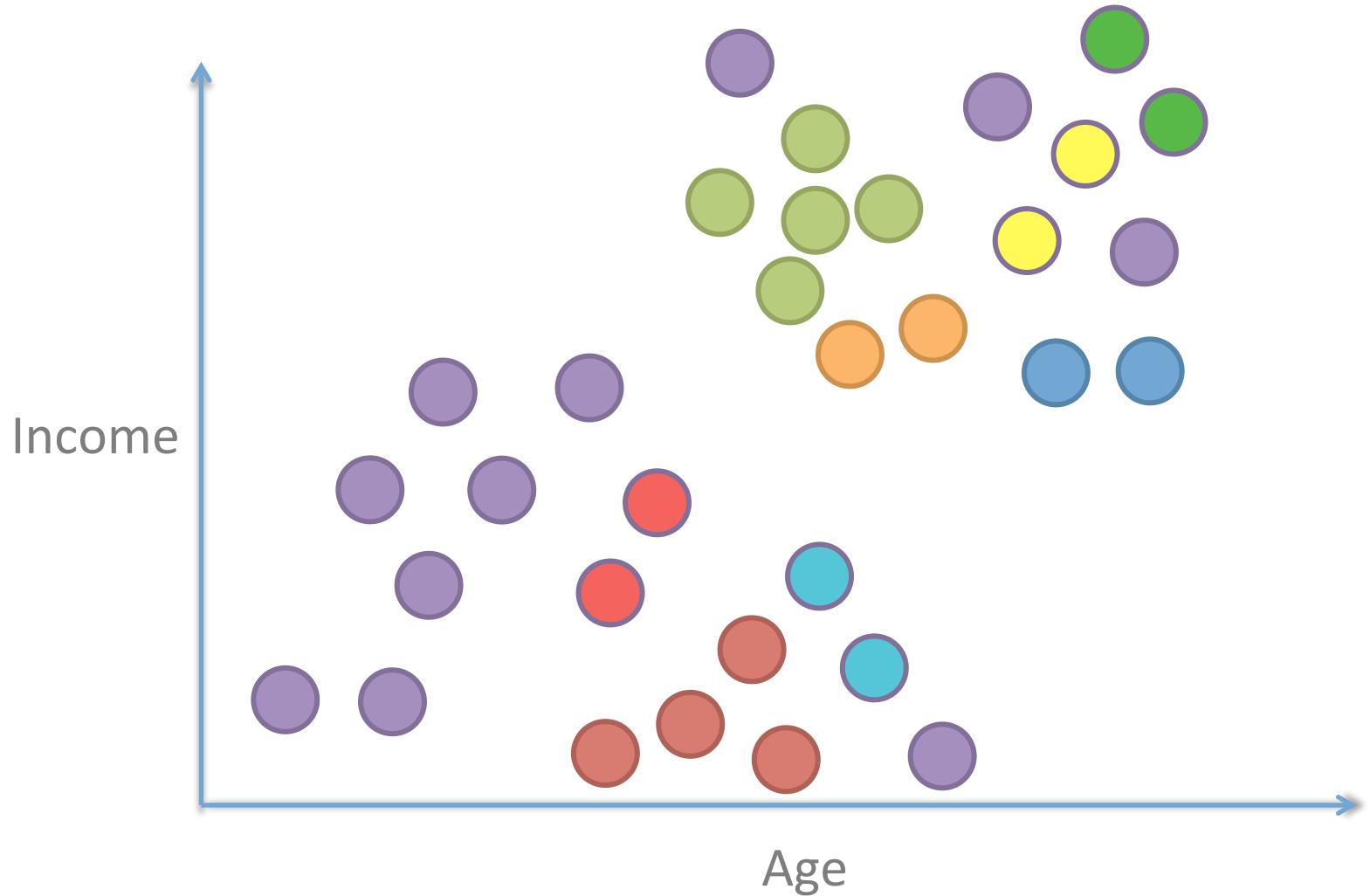
Agglomerative Hierarchical Clustering

If closest pair are clusters, same: merge



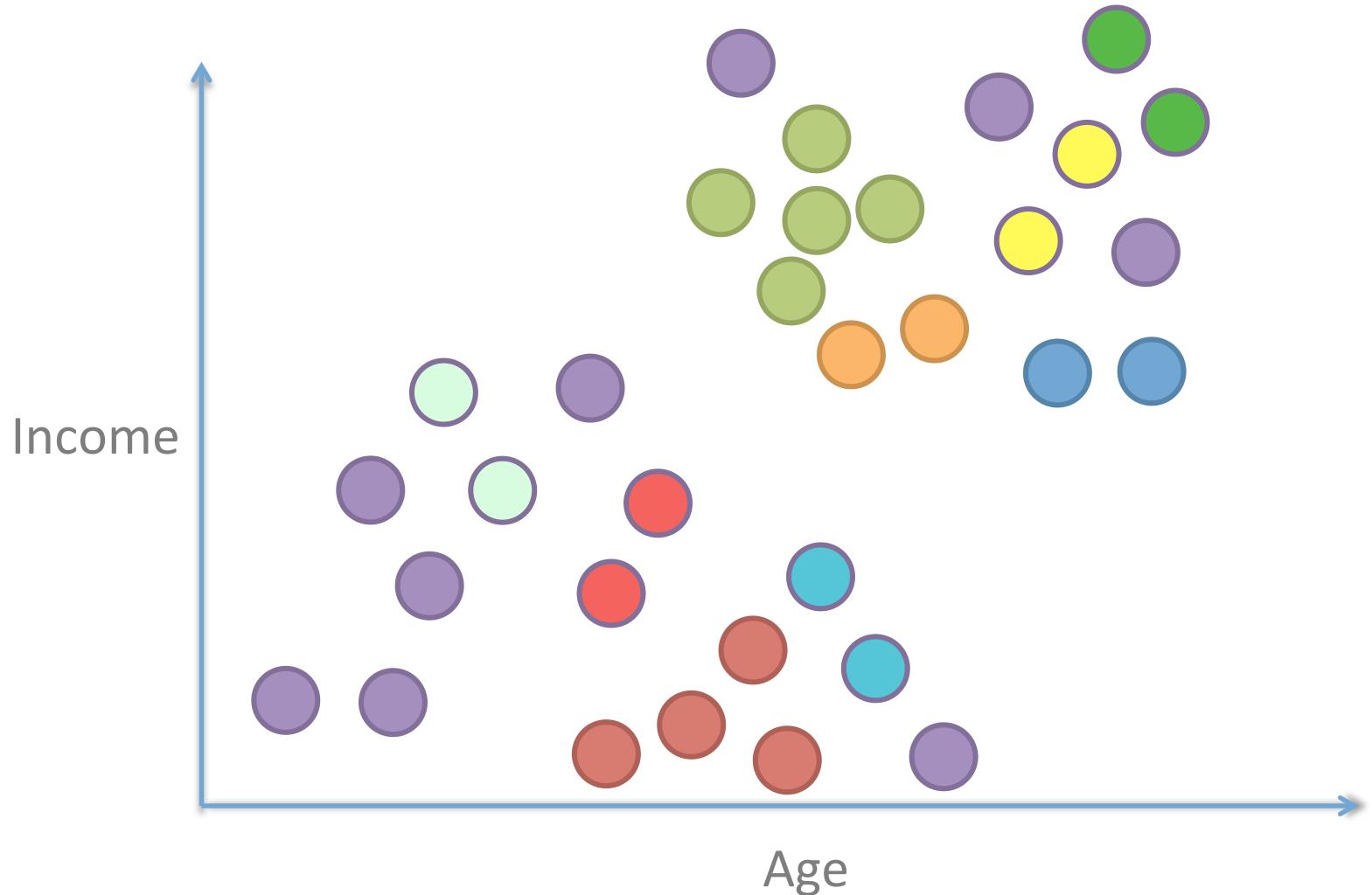
Agglomerative Hierarchical Clustering

If closest pair are clusters, same: merge



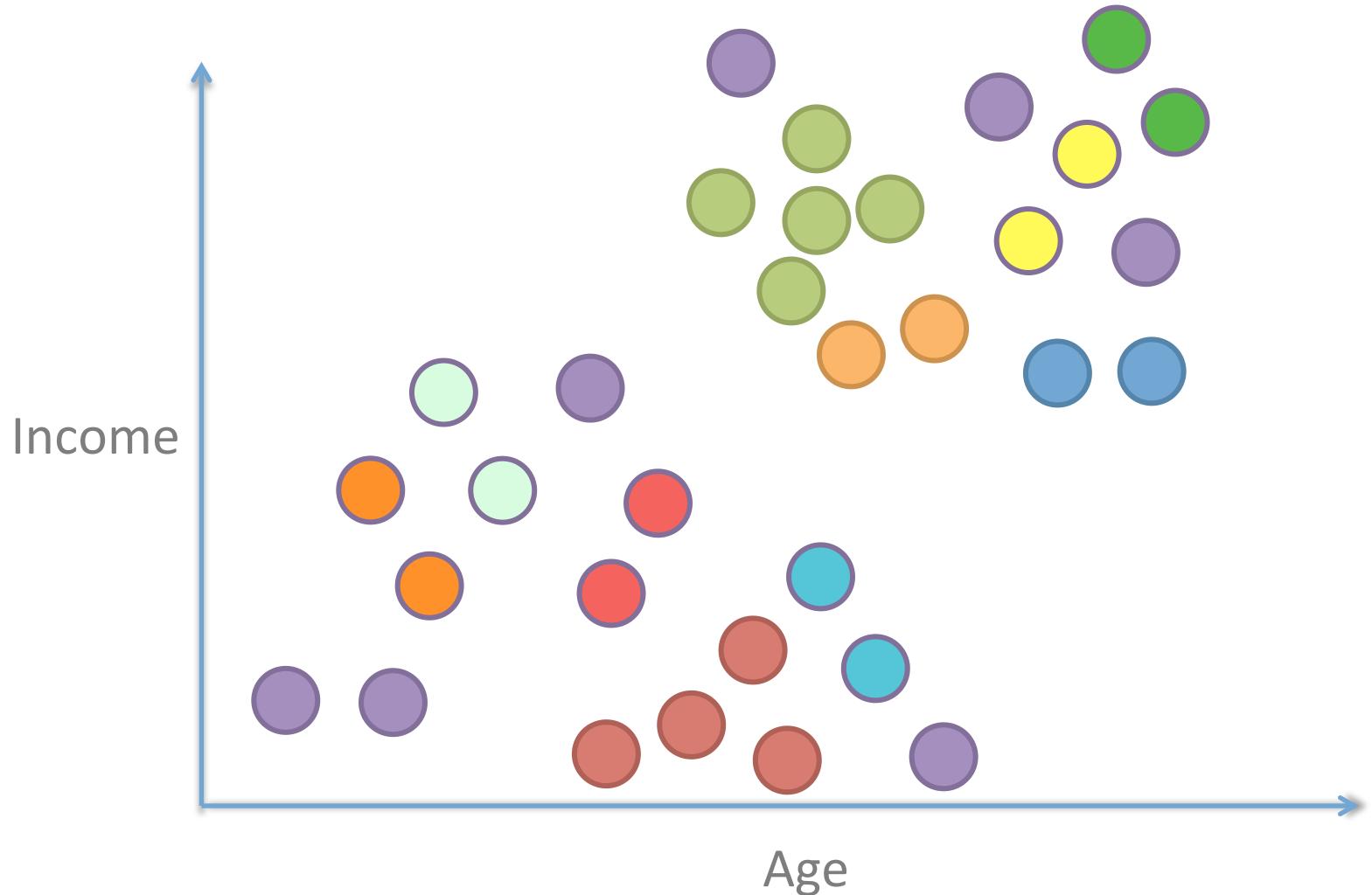
Agglomerative Hierarchical Clustering

If closest pair are clusters, same: merge



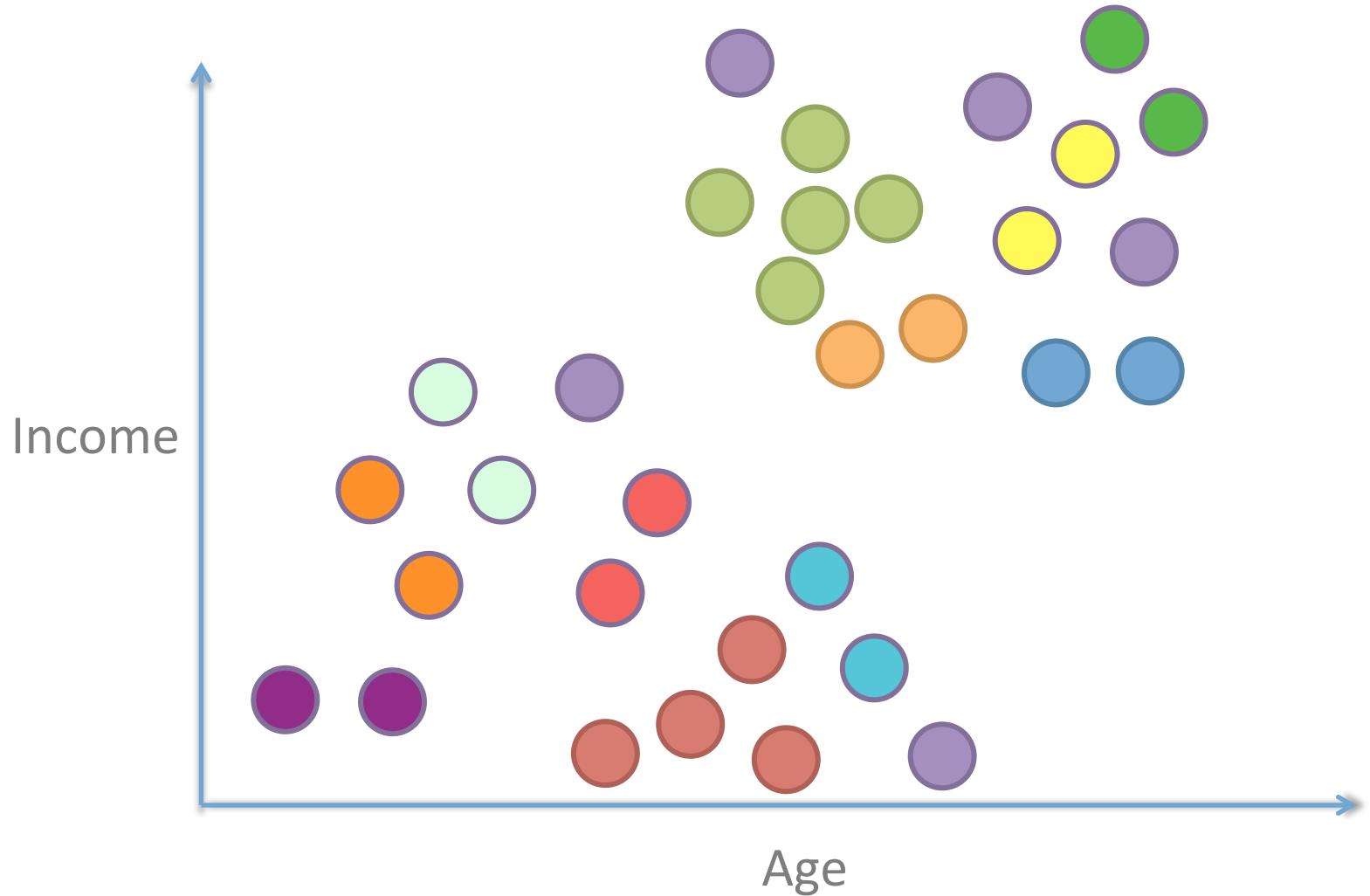
Agglomerative Hierarchical Clustering

If closest pair are clusters, same: merge



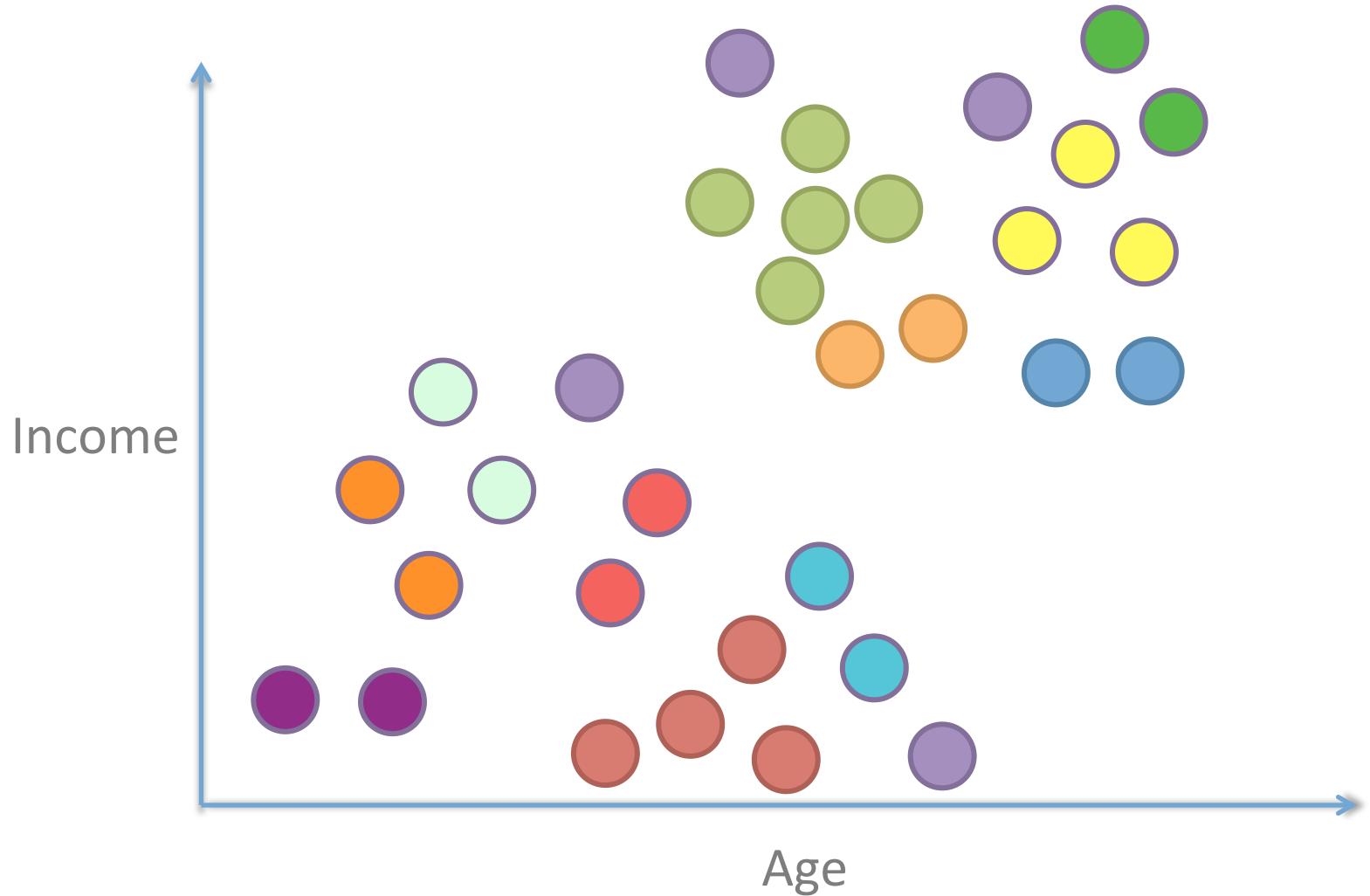
Agglomerative Hierarchical Clustering

If closest pair are clusters, same: merge



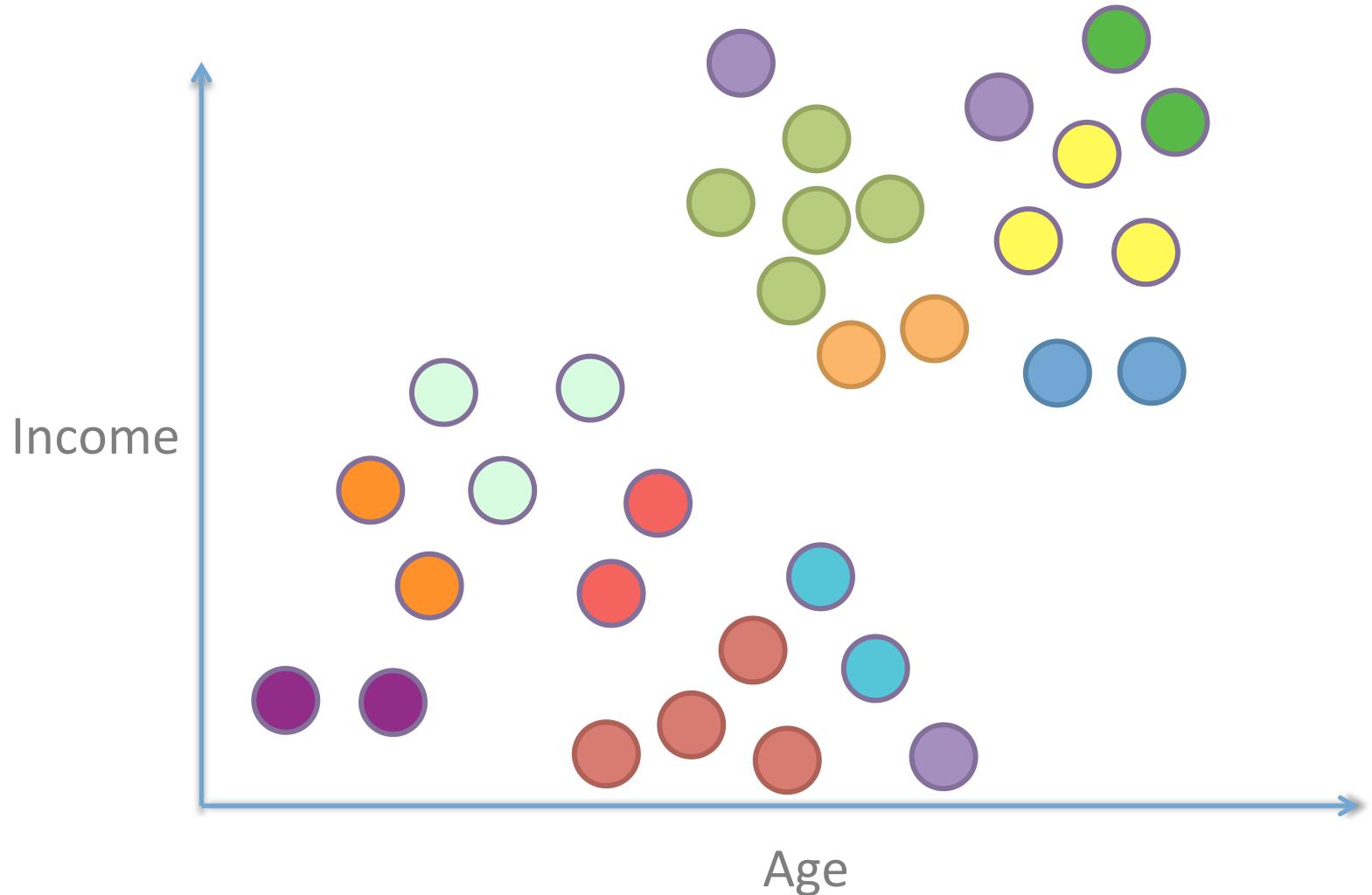
Agglomerative Hierarchical Clustering

If closest pair are clusters, same: merge



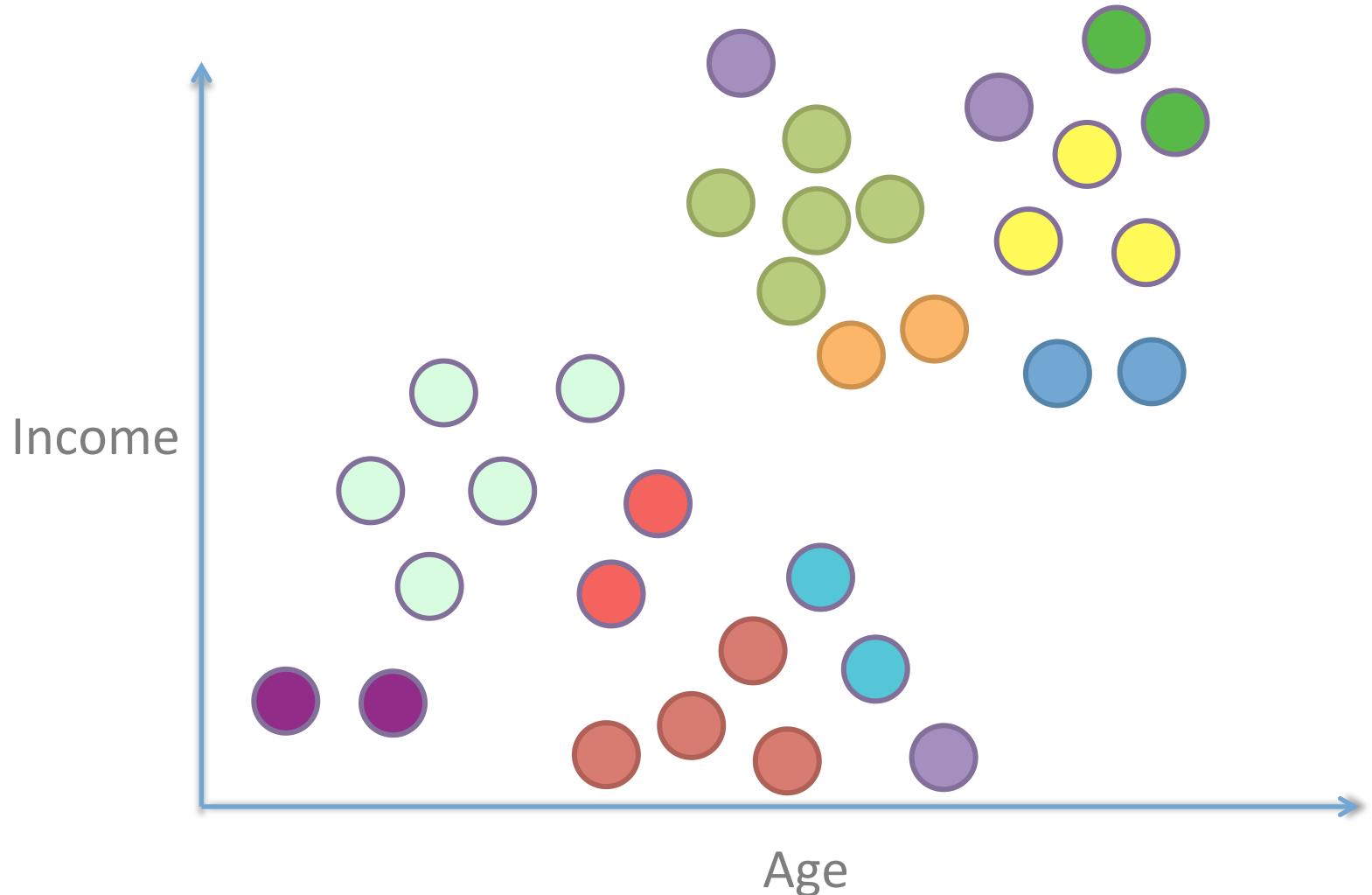
Agglomerative Hierarchical Clustering

If closest pair are clusters, same: merge



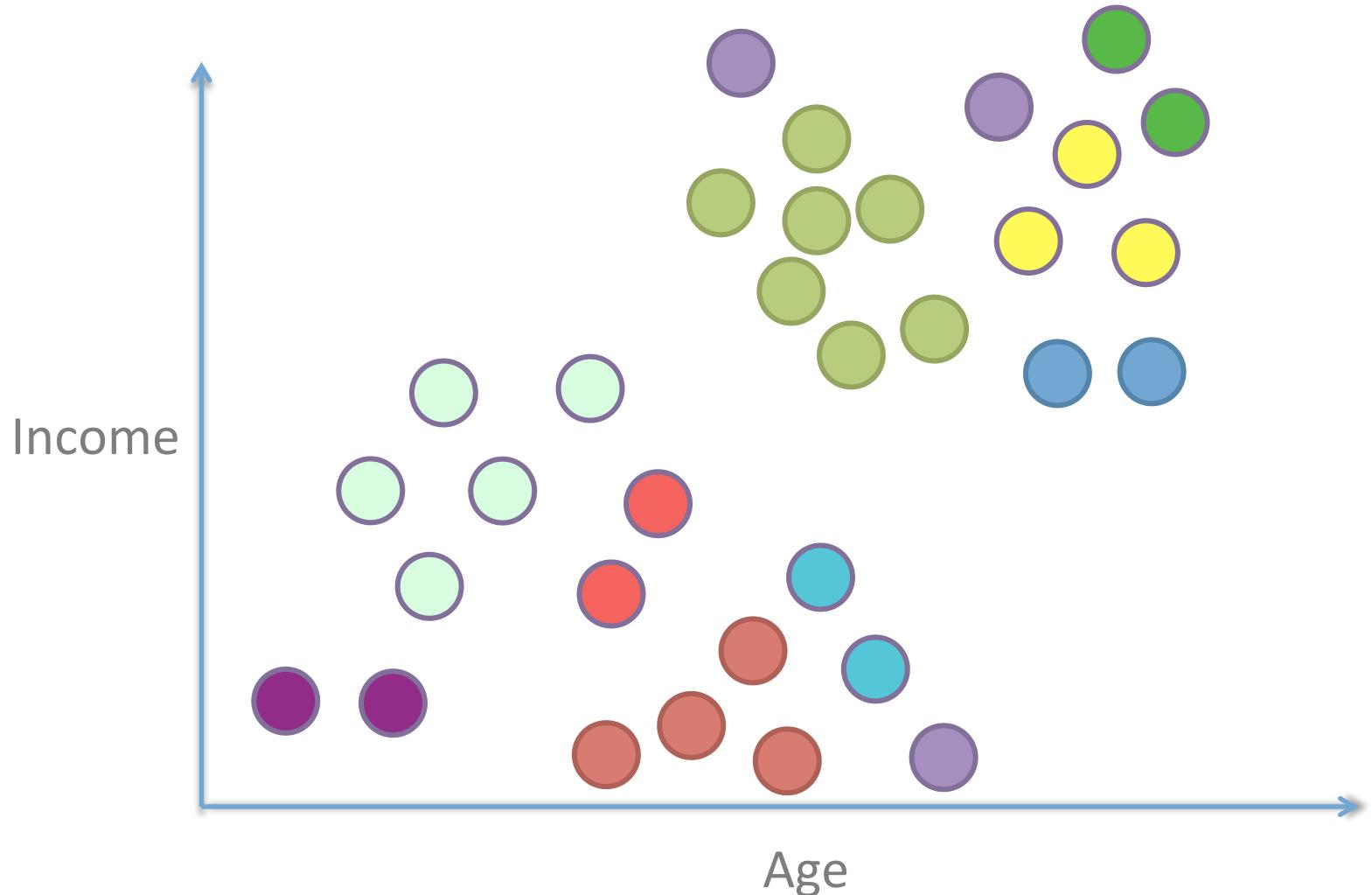
Agglomerative Hierarchical Clustering

If closest pair are clusters, same: merge



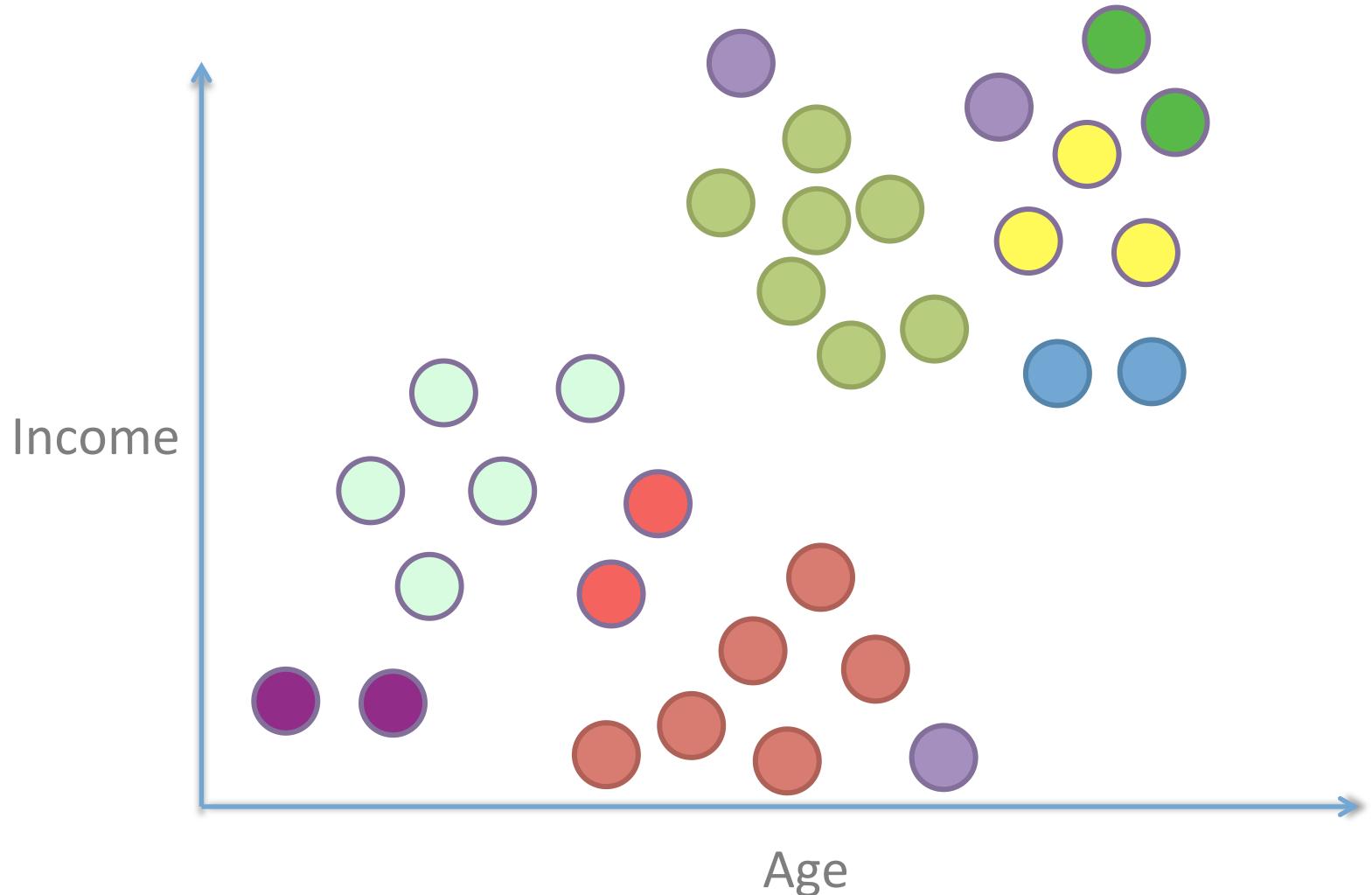
Agglomerative Hierarchical Clustering

If closest pair are clusters, same: merge



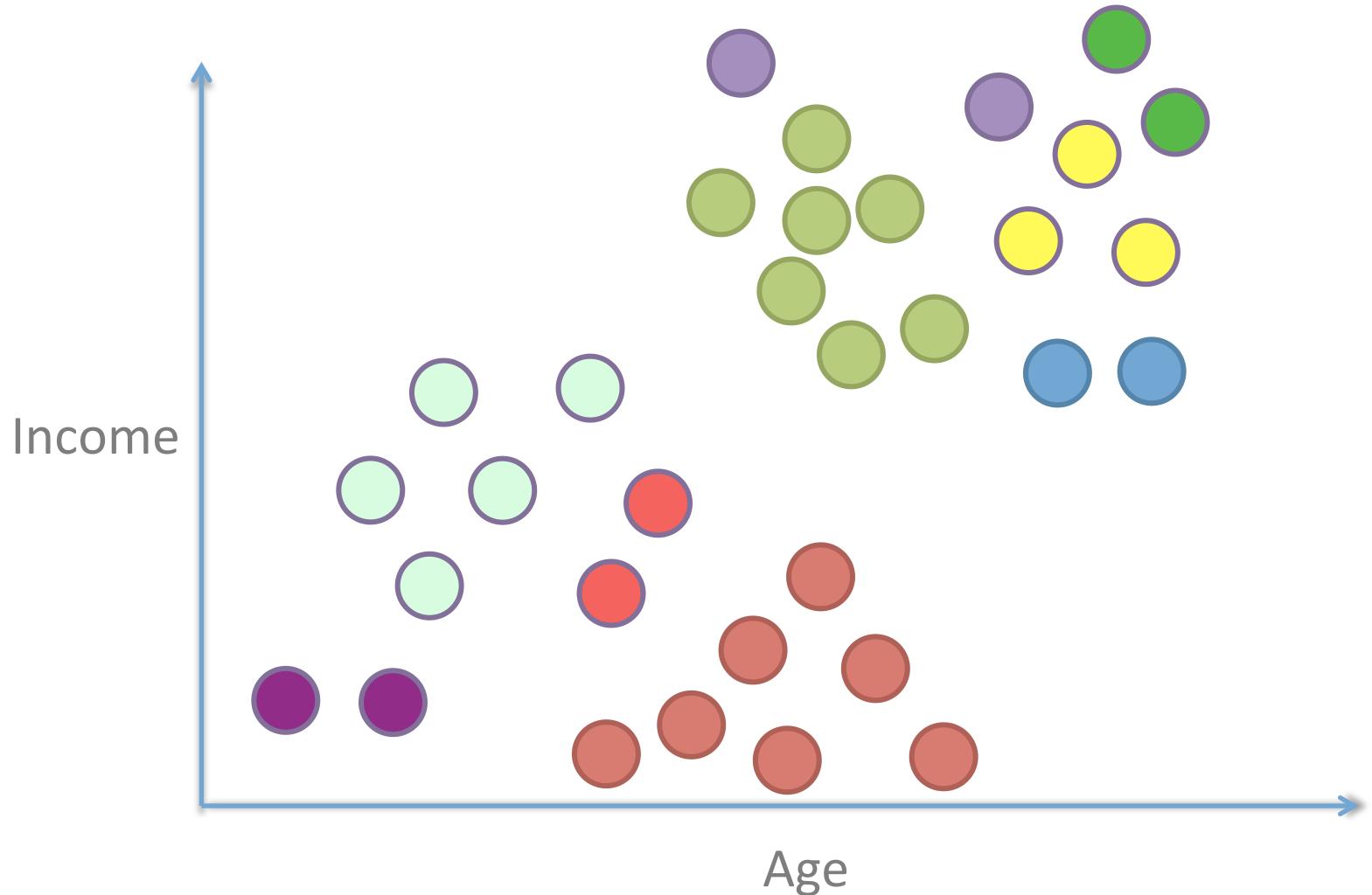
Agglomerative Hierarchical Clustering

If closest pair are clusters, same: merge



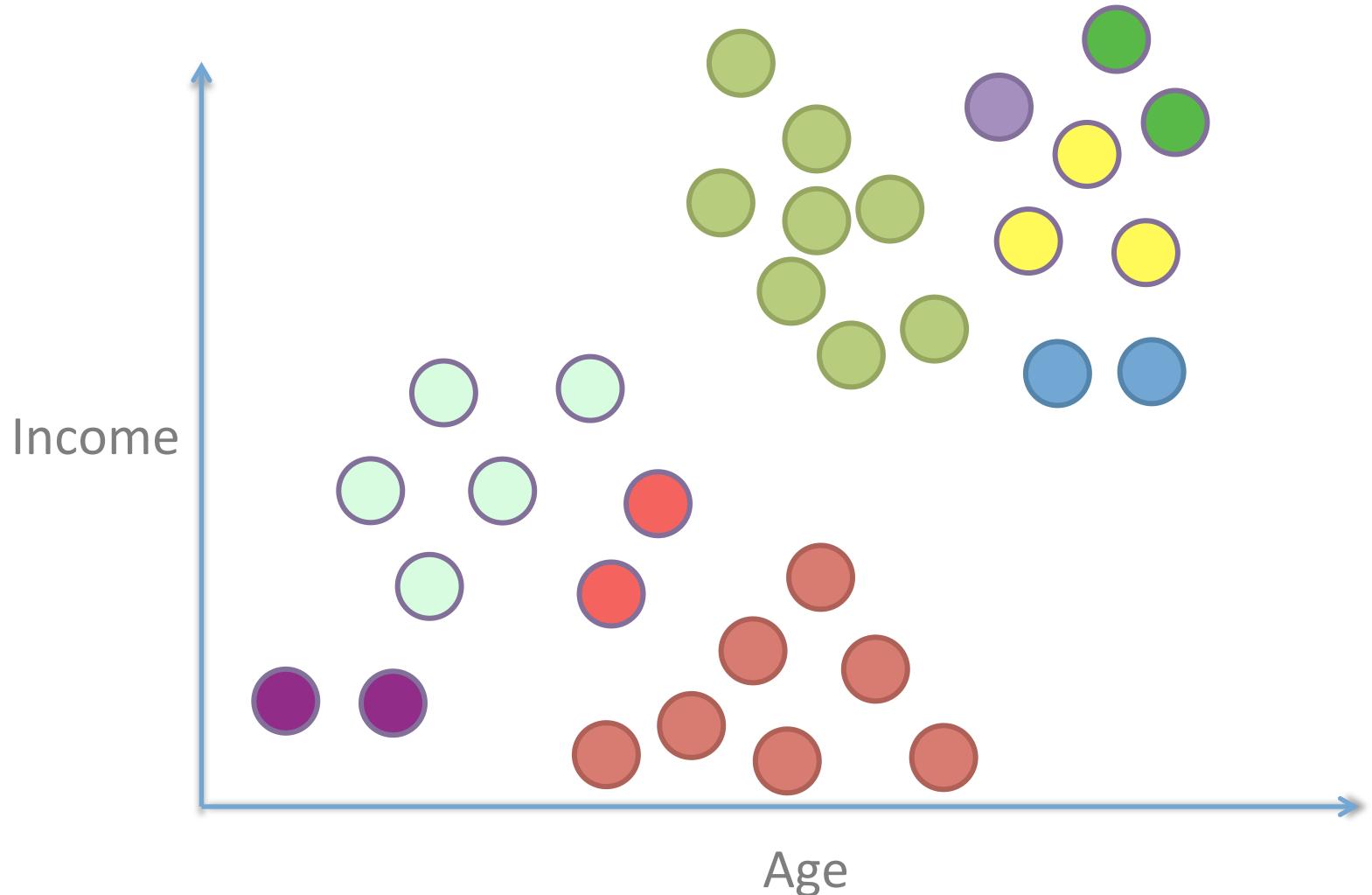
Agglomerative Hierarchical Clustering

If closest pair are clusters, same: merge



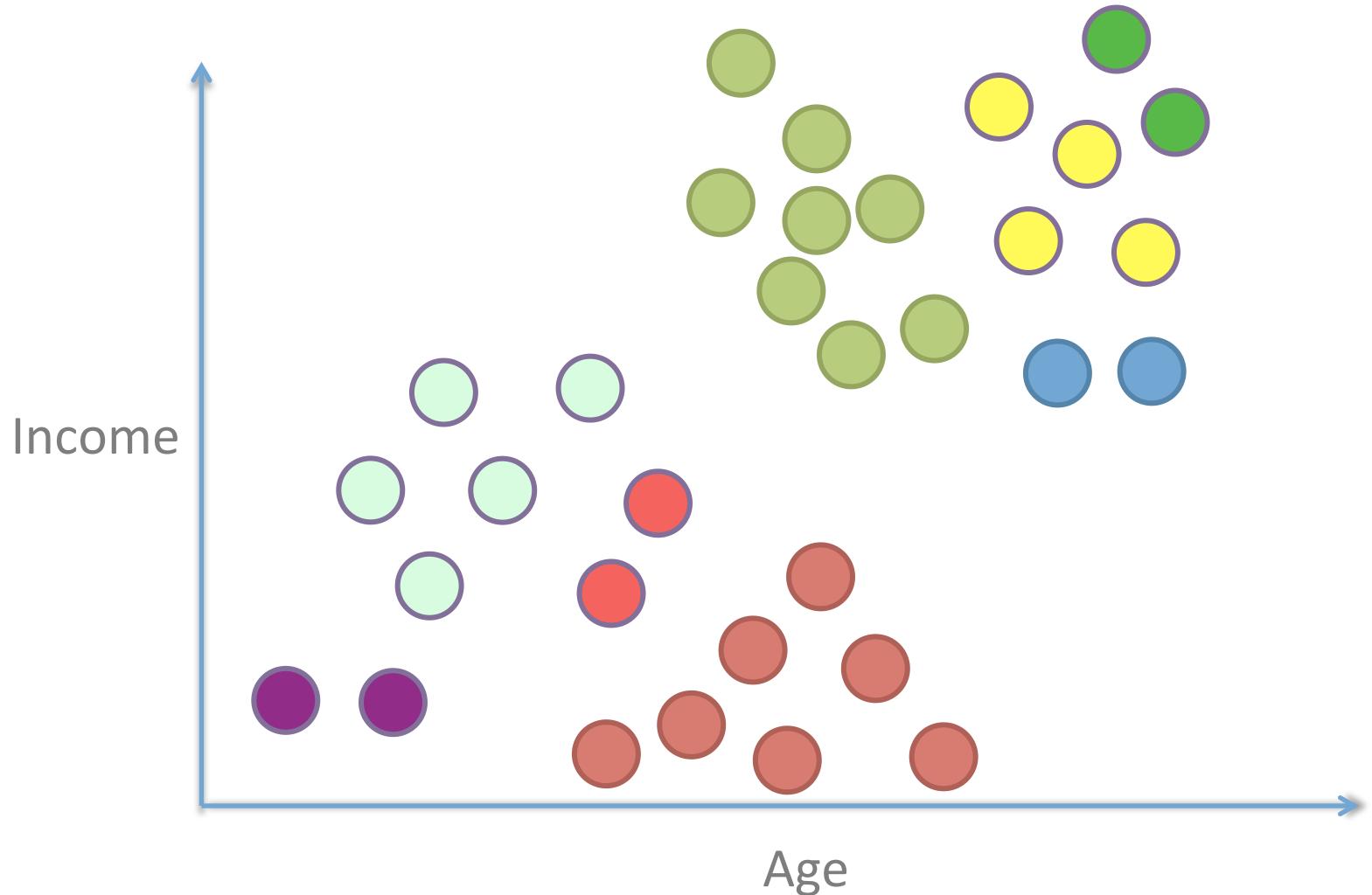
Agglomerative Hierarchical Clustering

If closest pair are clusters, same: merge



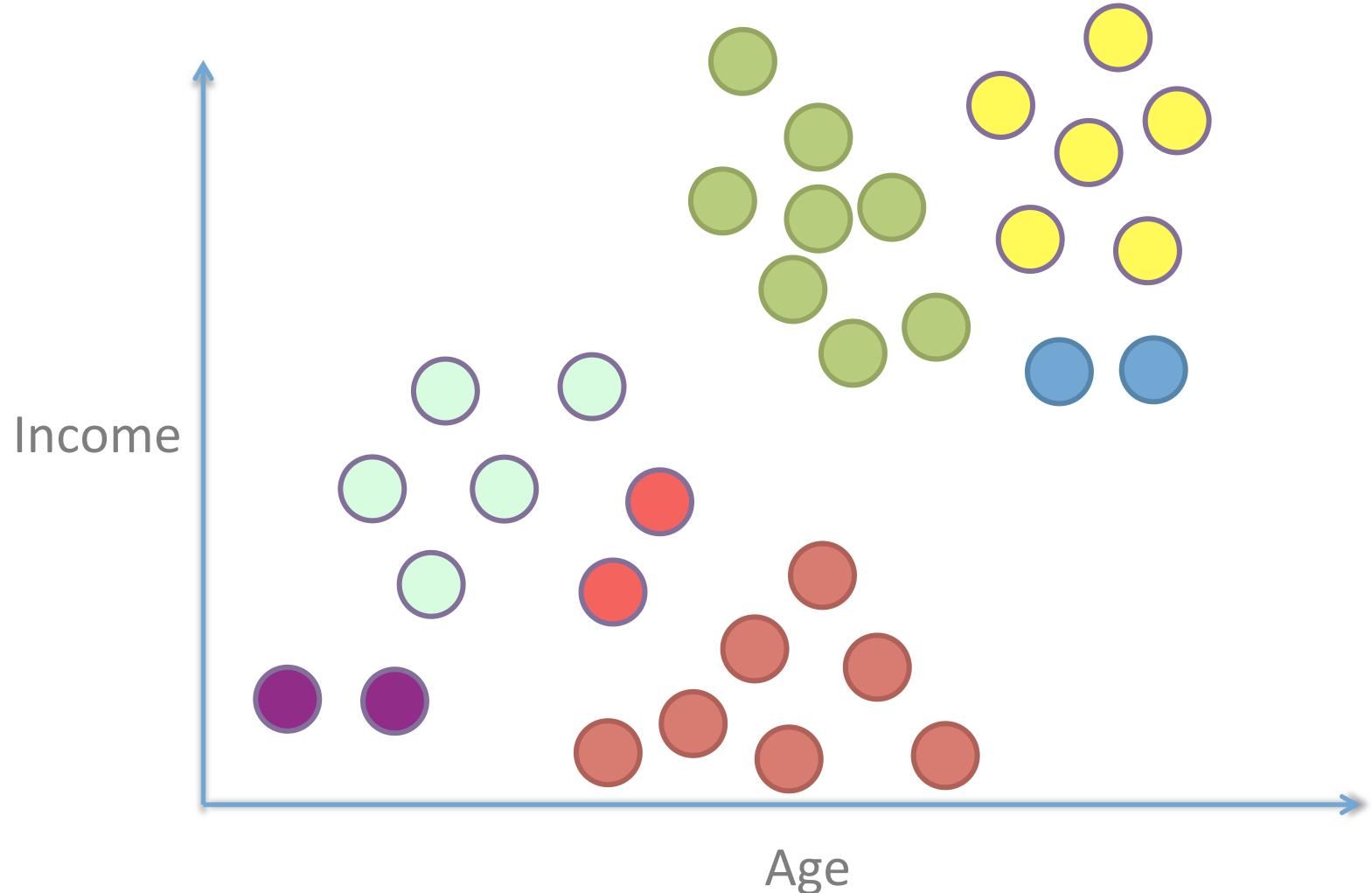
Agglomerative Hierarchical Clustering

If closest pair are clusters, same: merge



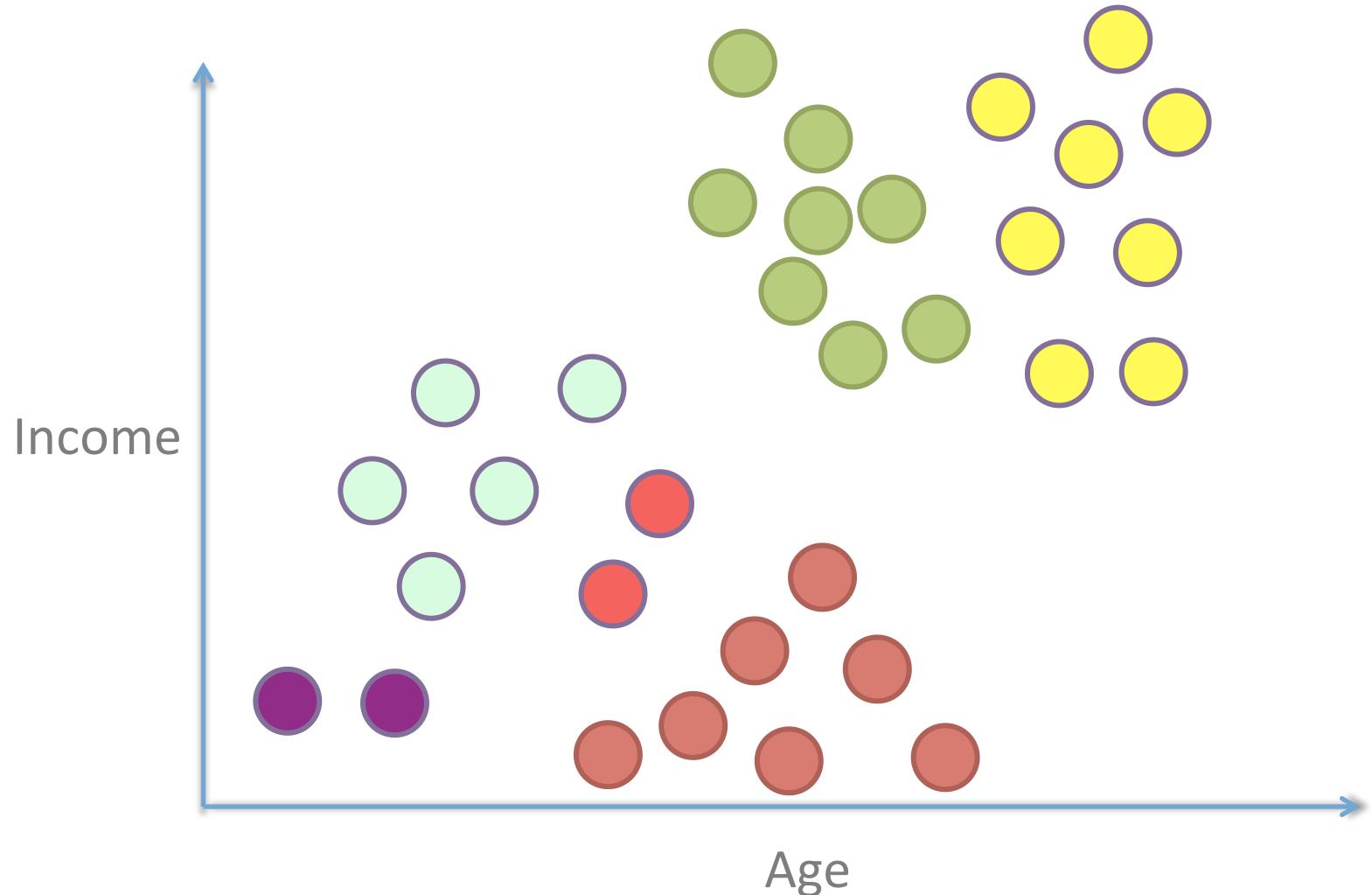
Agglomerative Hierarchical Clustering

Current `num_clusters = 7`



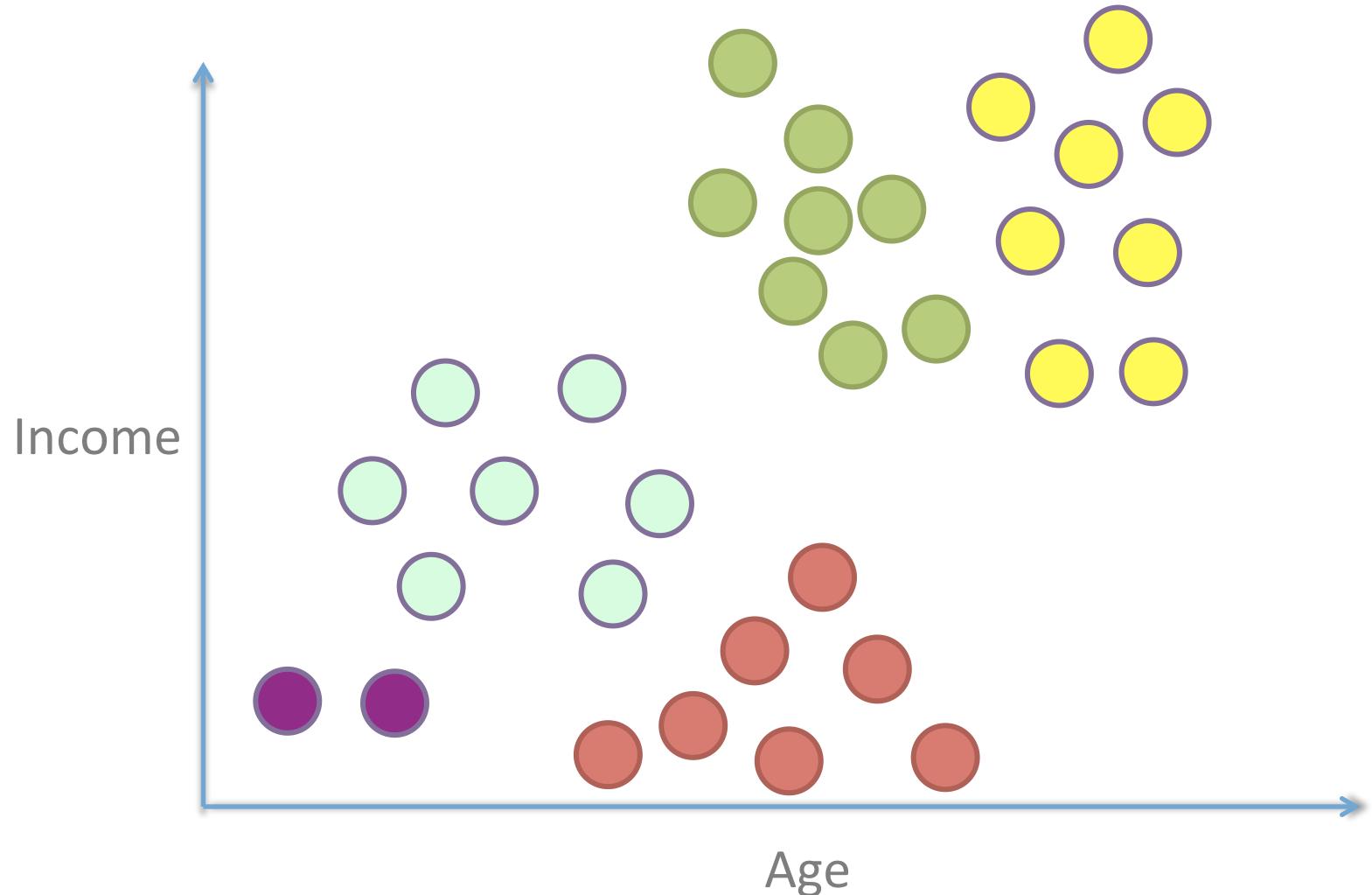
Agglomerative Hierarchical Clustering

Current `num_clusters = 6`



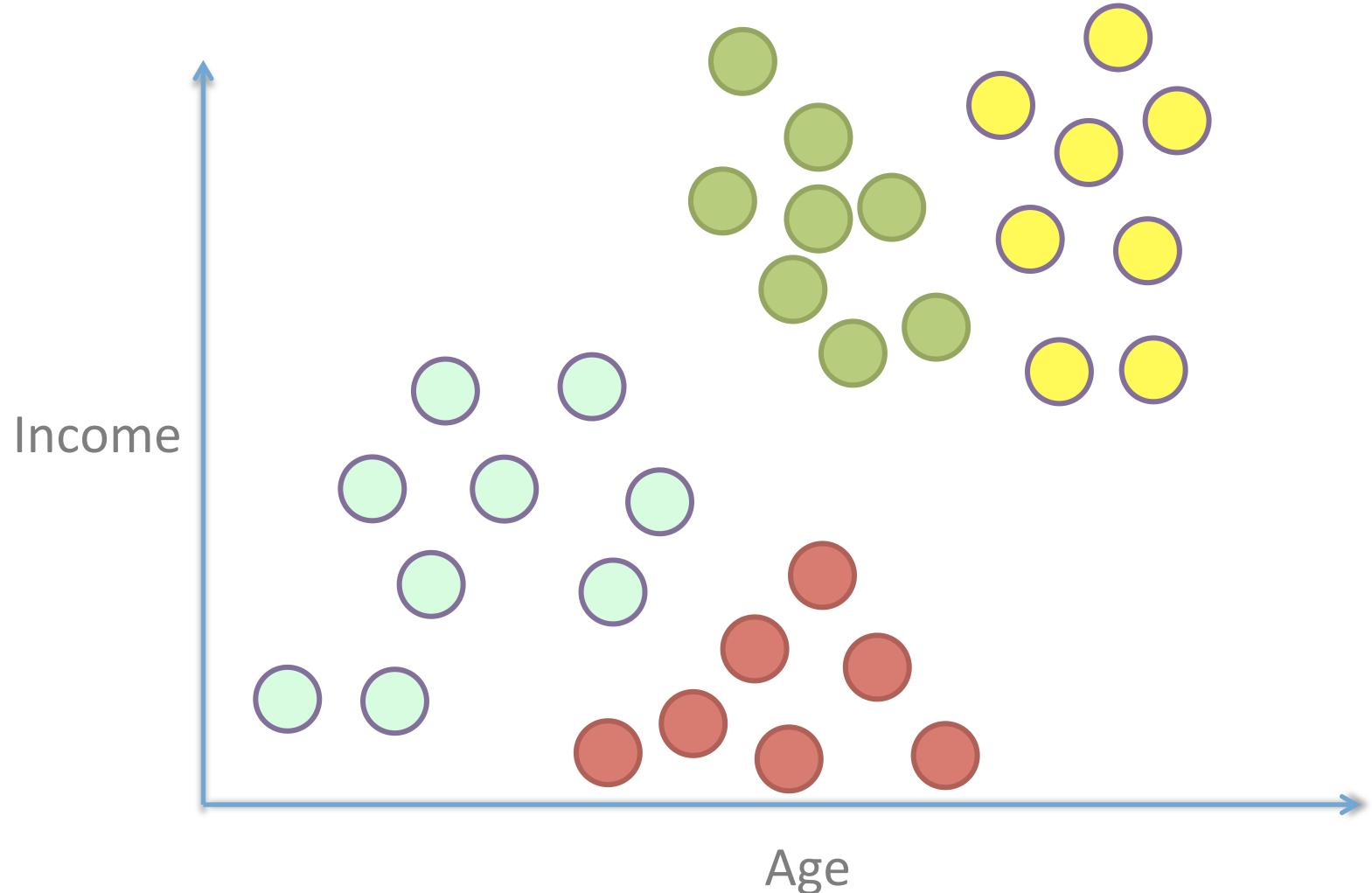
Agglomerative Hierarchical Clustering

Current `num_clusters = 5`



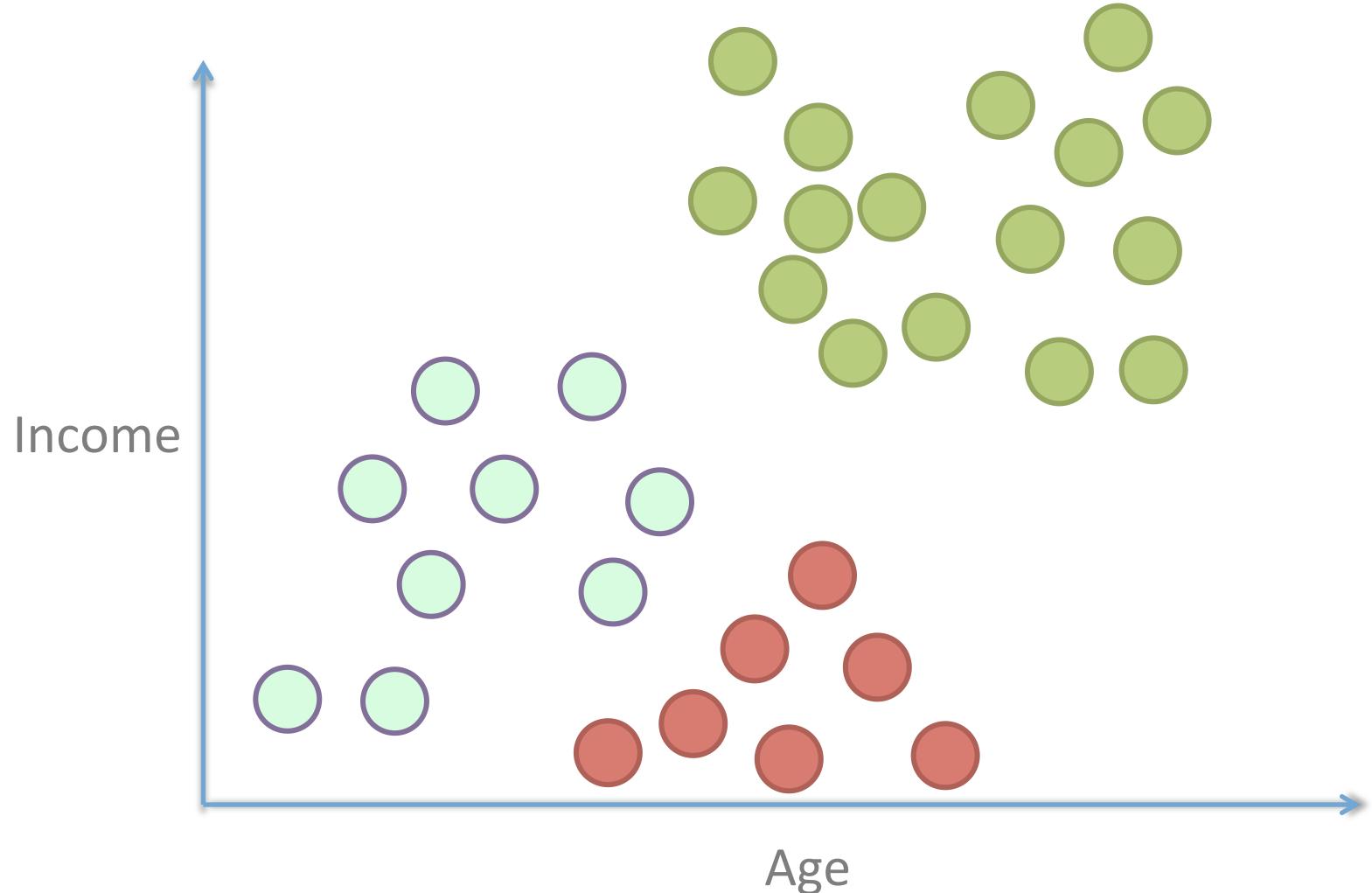
Agglomerative Hierarchical Clustering

Current `num_clusters = 4`



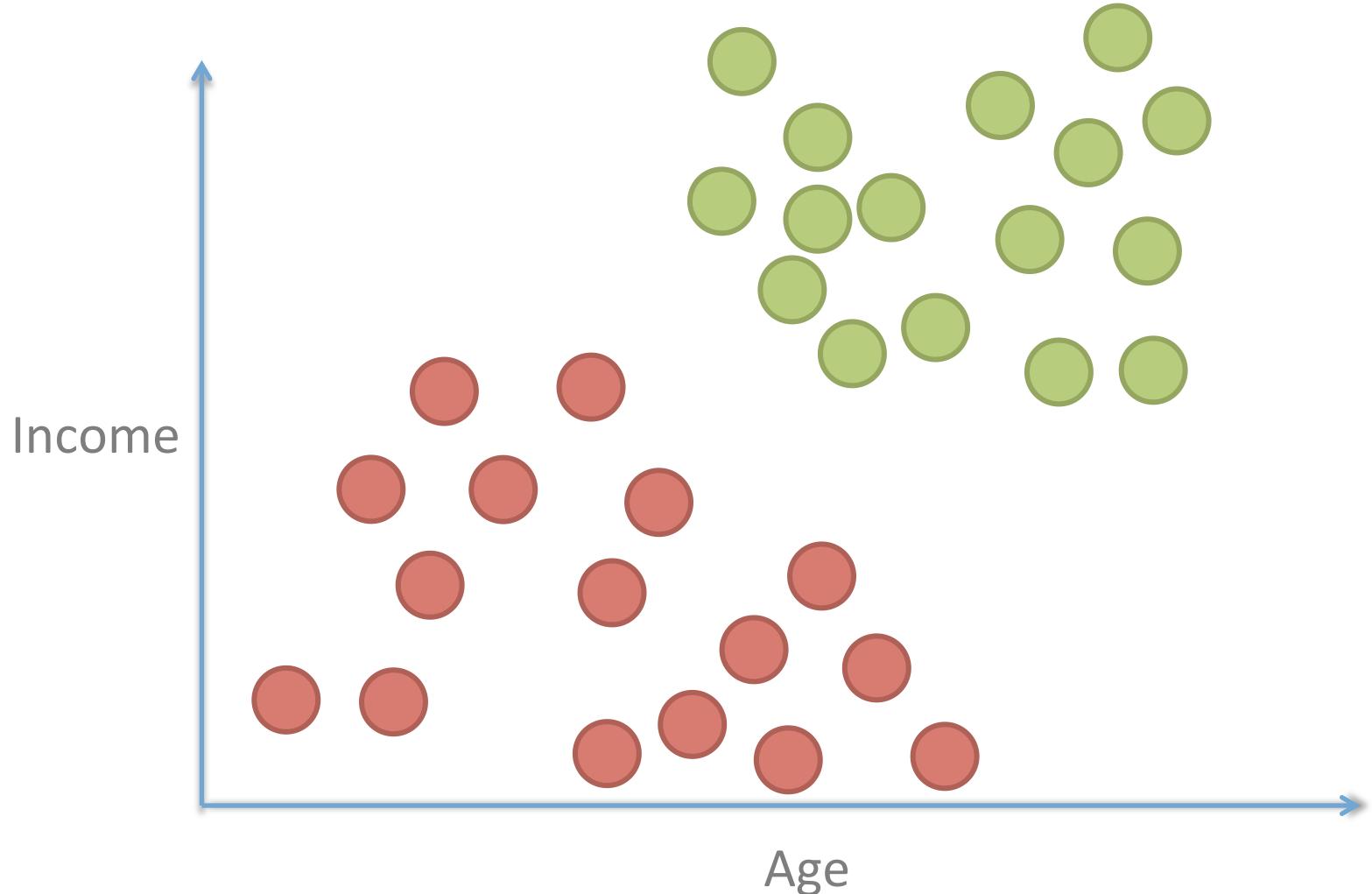
Agglomerative Hierarchical Clustering

Current `num_clusters = 3`



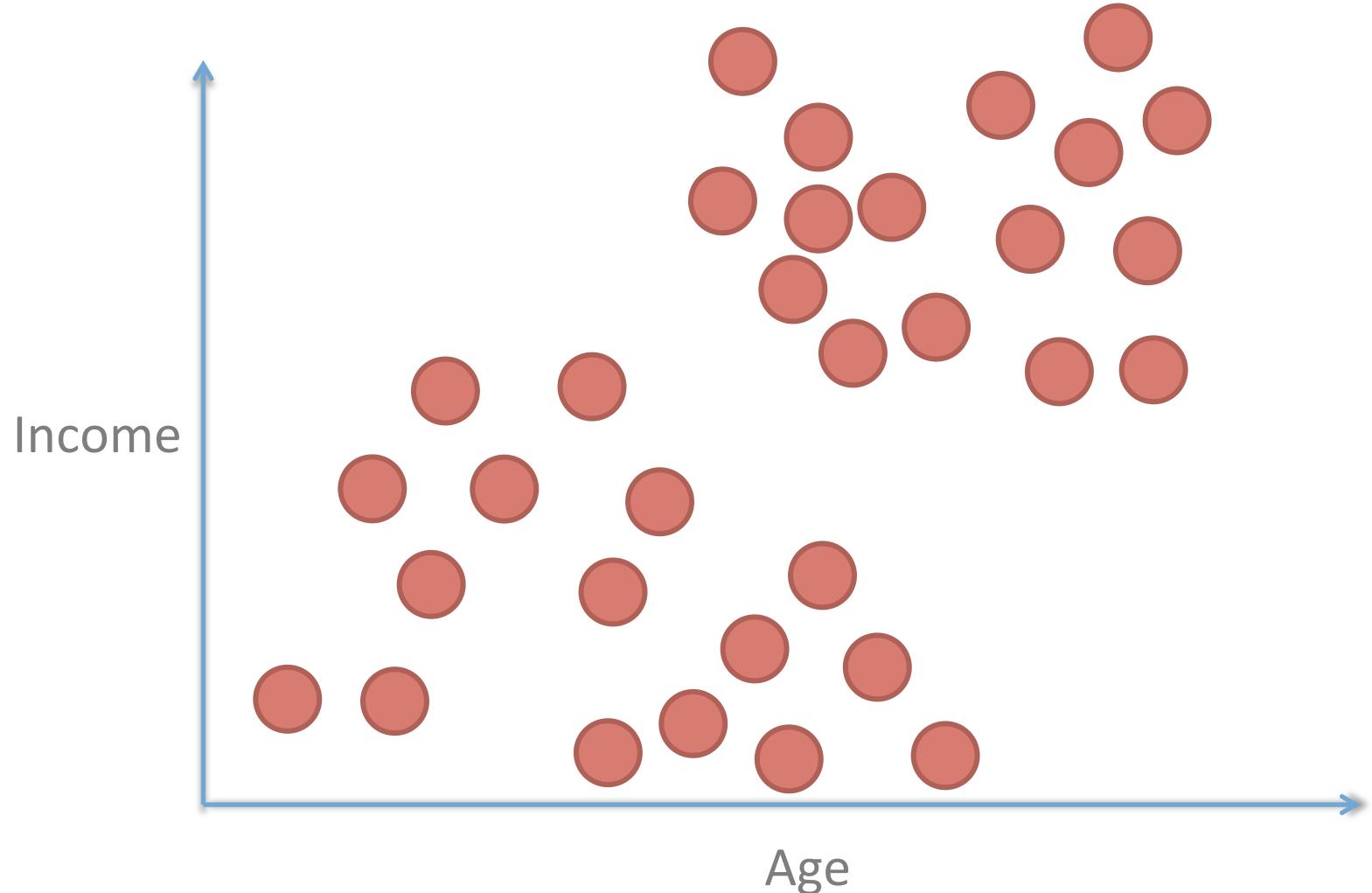
Agglomerative Hierarchical Clustering

Current `num_clusters = 2`



Agglomerative Hierarchical Clustering

Current `num_clusters = 1`



Agglomerative Hierarchical Clustering

Stop either

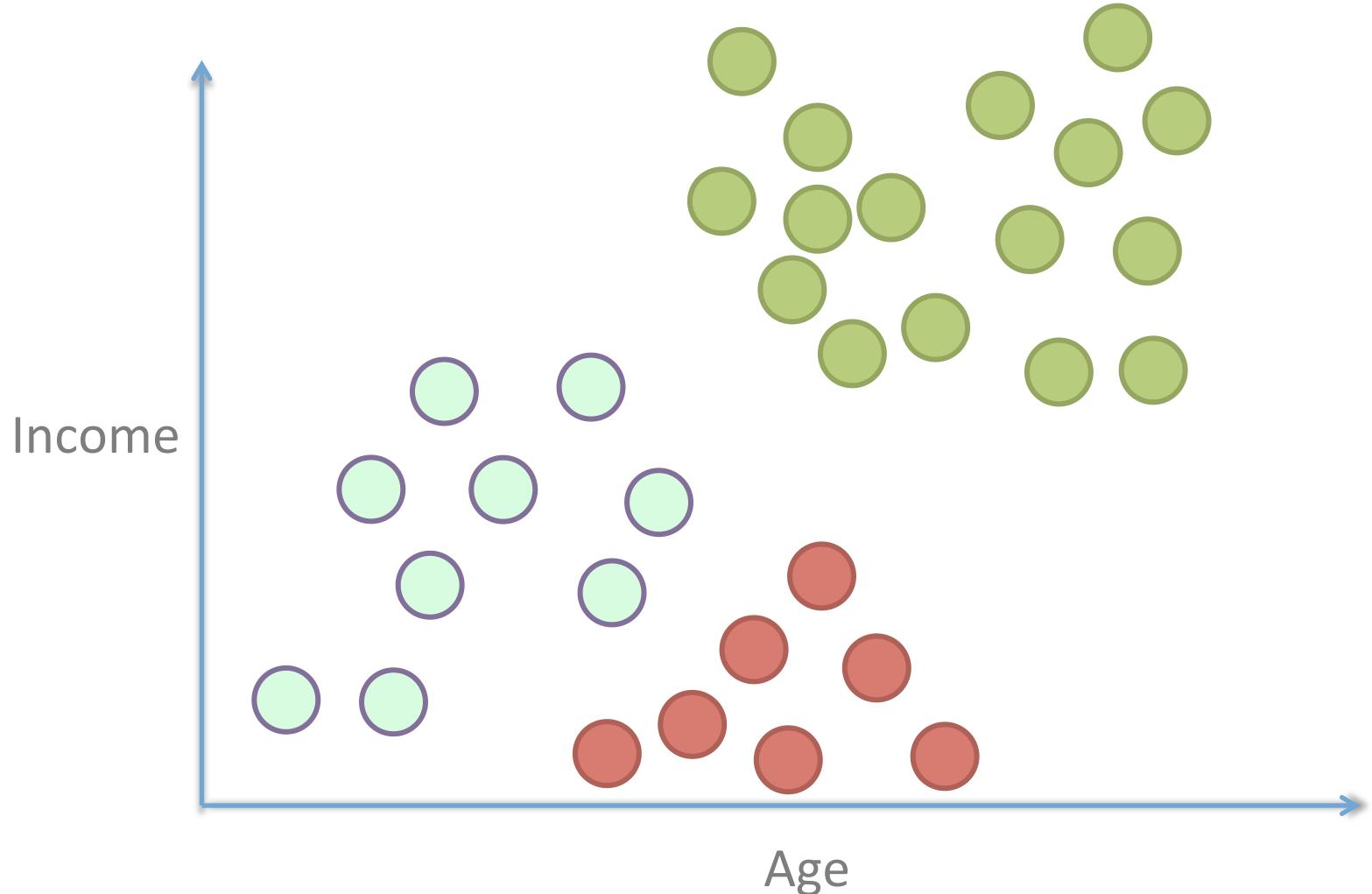
- when you reach the correct `num_clusters` (next step would mean fewer clusters than desired)

Or

- when `min cluster distance` reaches a set value (next step would mean merging along too large a distance)

Agglomerative Hierarchical Clustering

Current `num_clusters = 3`



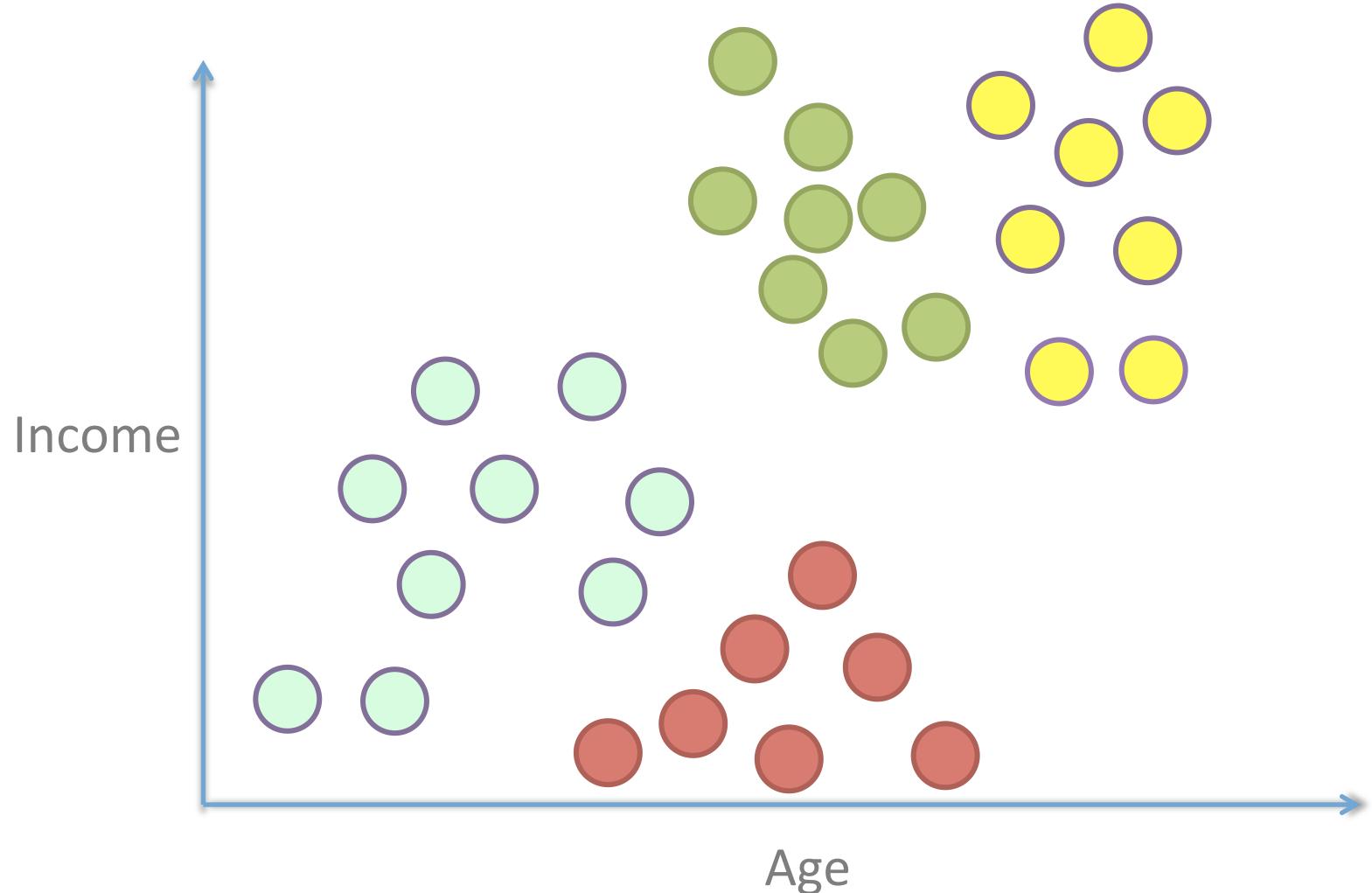
Agglomerative Hierarchical Clustering

Current `num_clusters` = 3



Agglomerative Hierarchical Clustering

Current `num_clusters = 4`



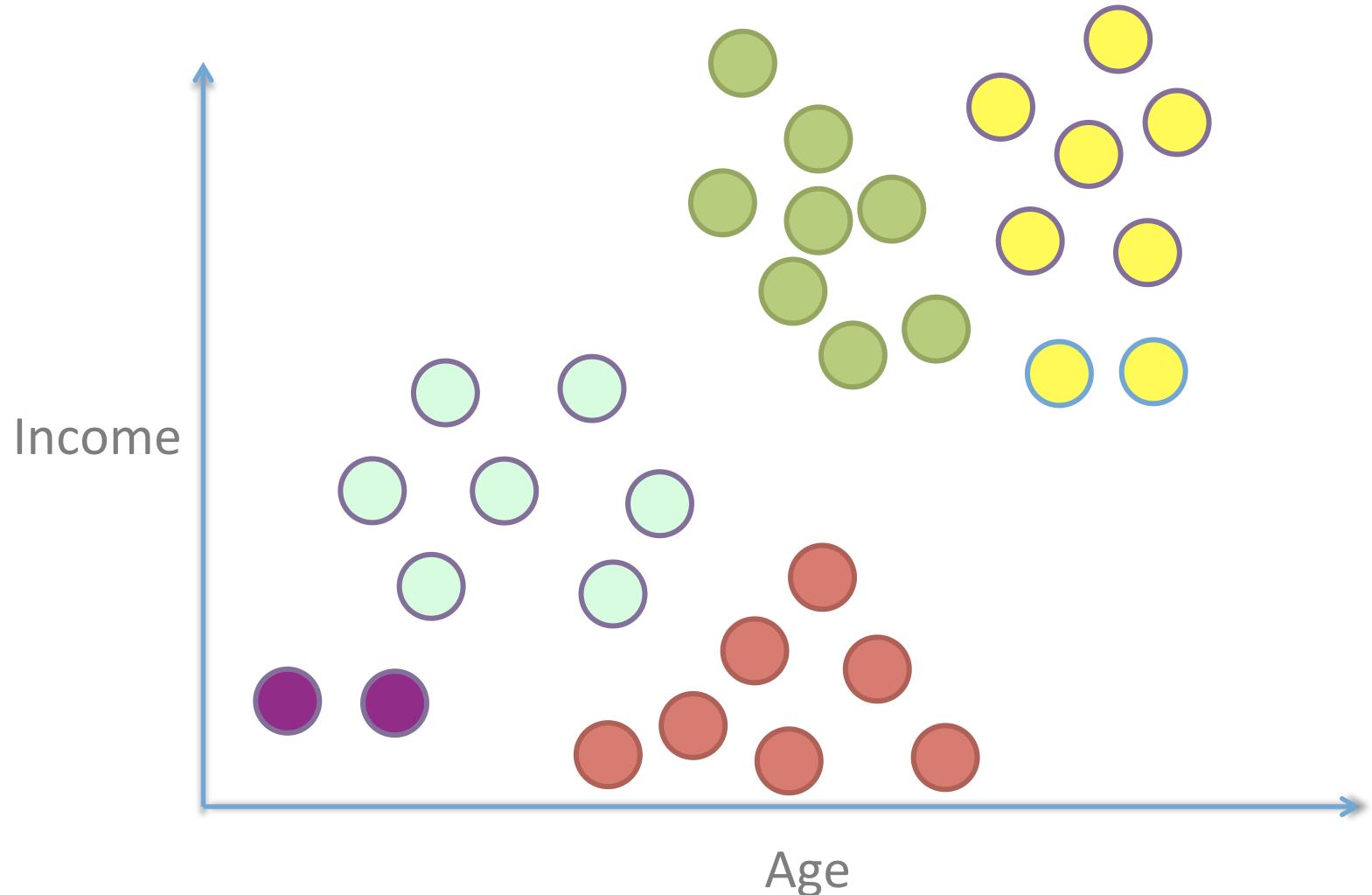
Agglomerative Hierarchical Clustering

Current `num_clusters = 4`



Agglomerative Hierarchical Clustering

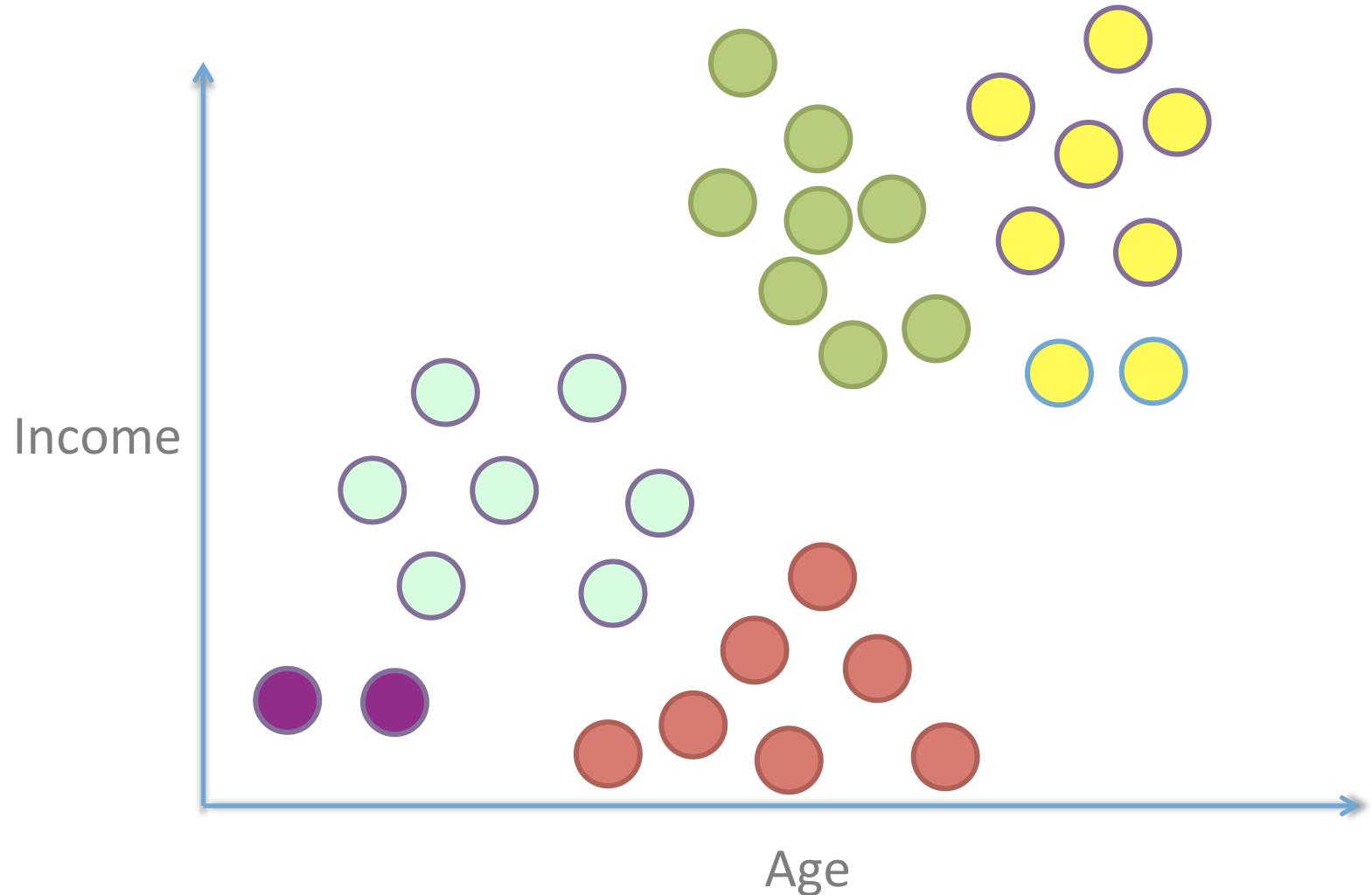
Agglomerative Hierarchical Clustering



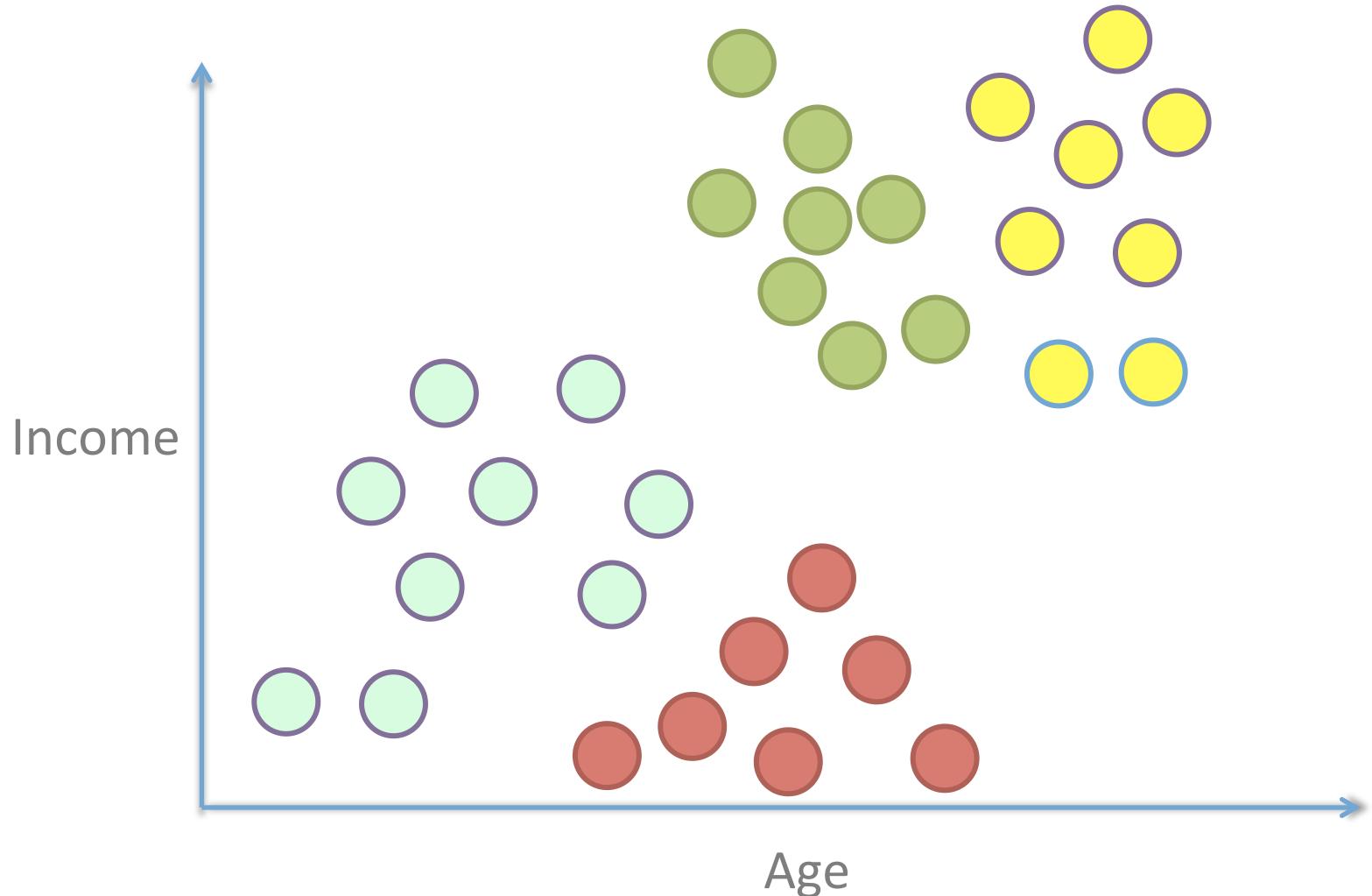
Agglomerative Hierarchical Clustering



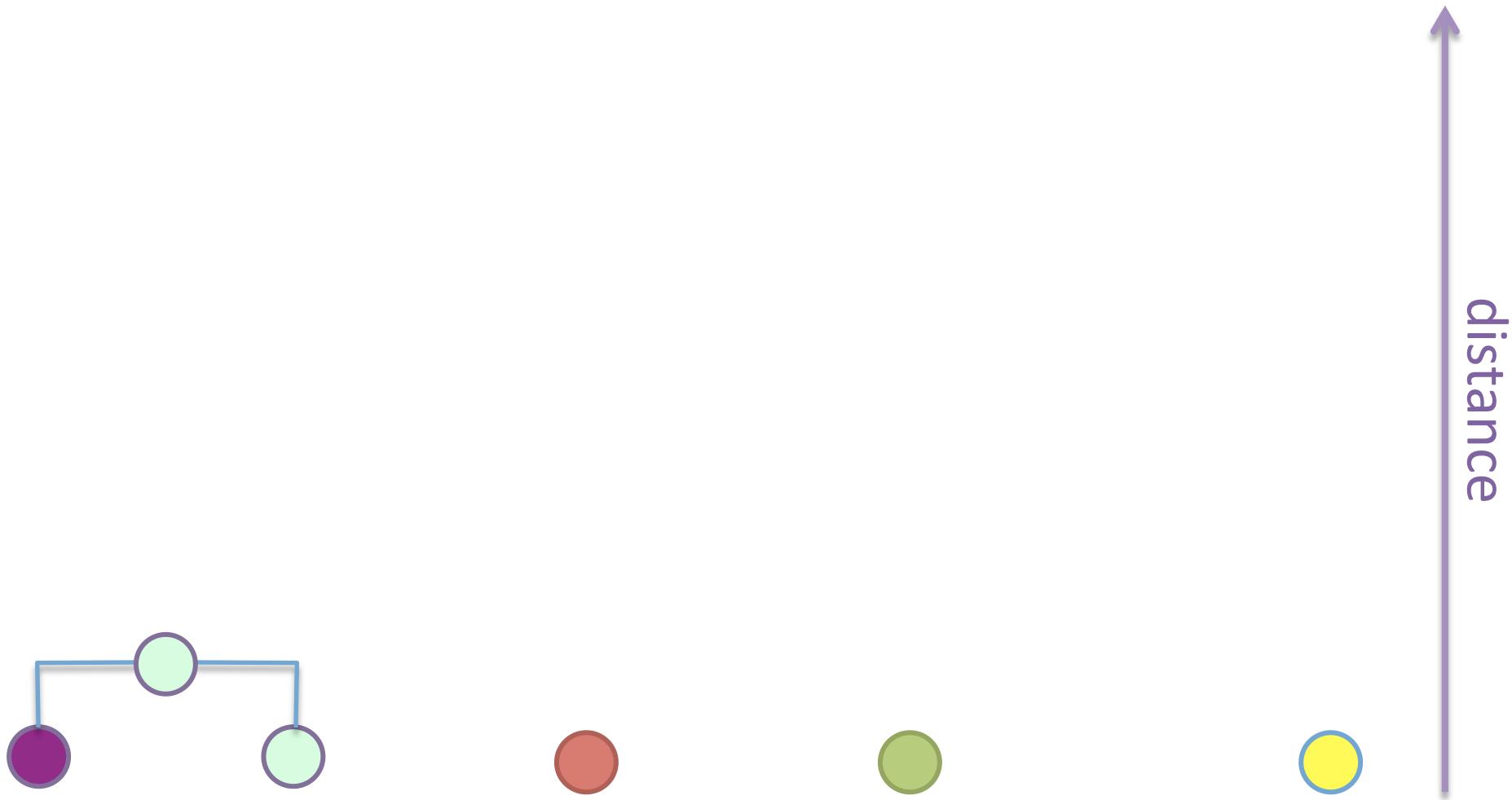
Agglomerative Hierarchical Clustering



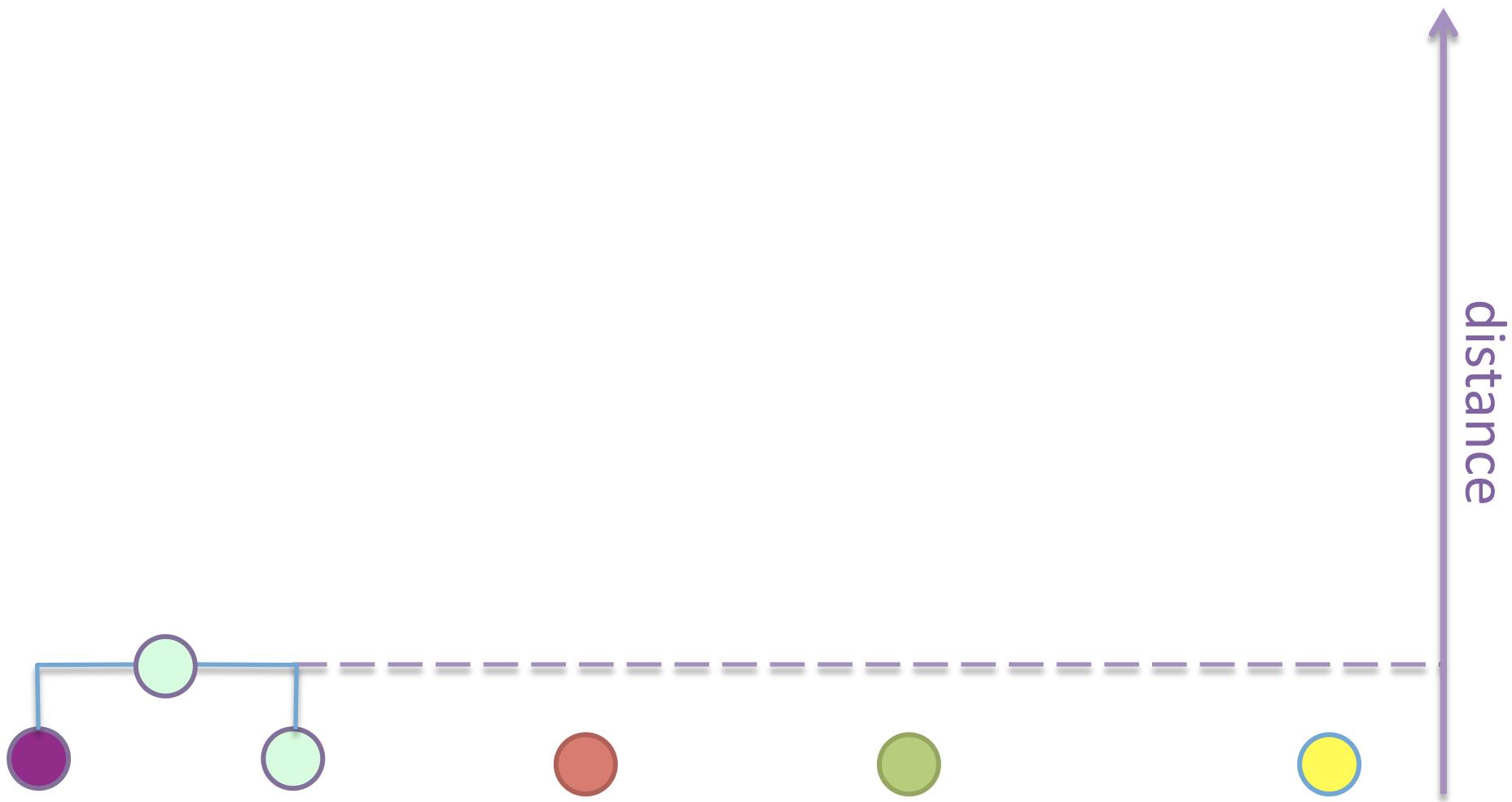
Agglomerative Hierarchical Clustering



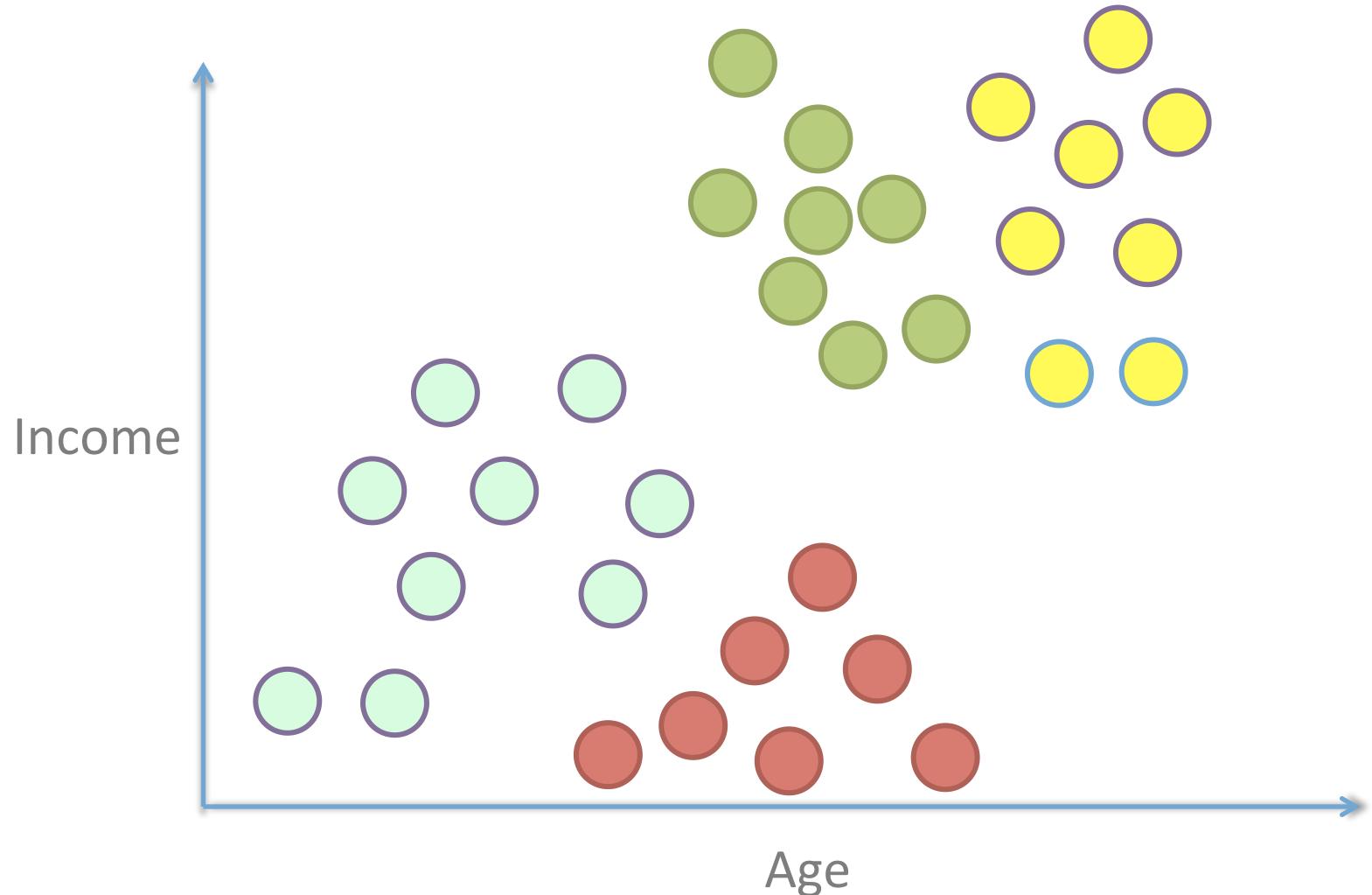
Agglomerative Hierarchical Clustering



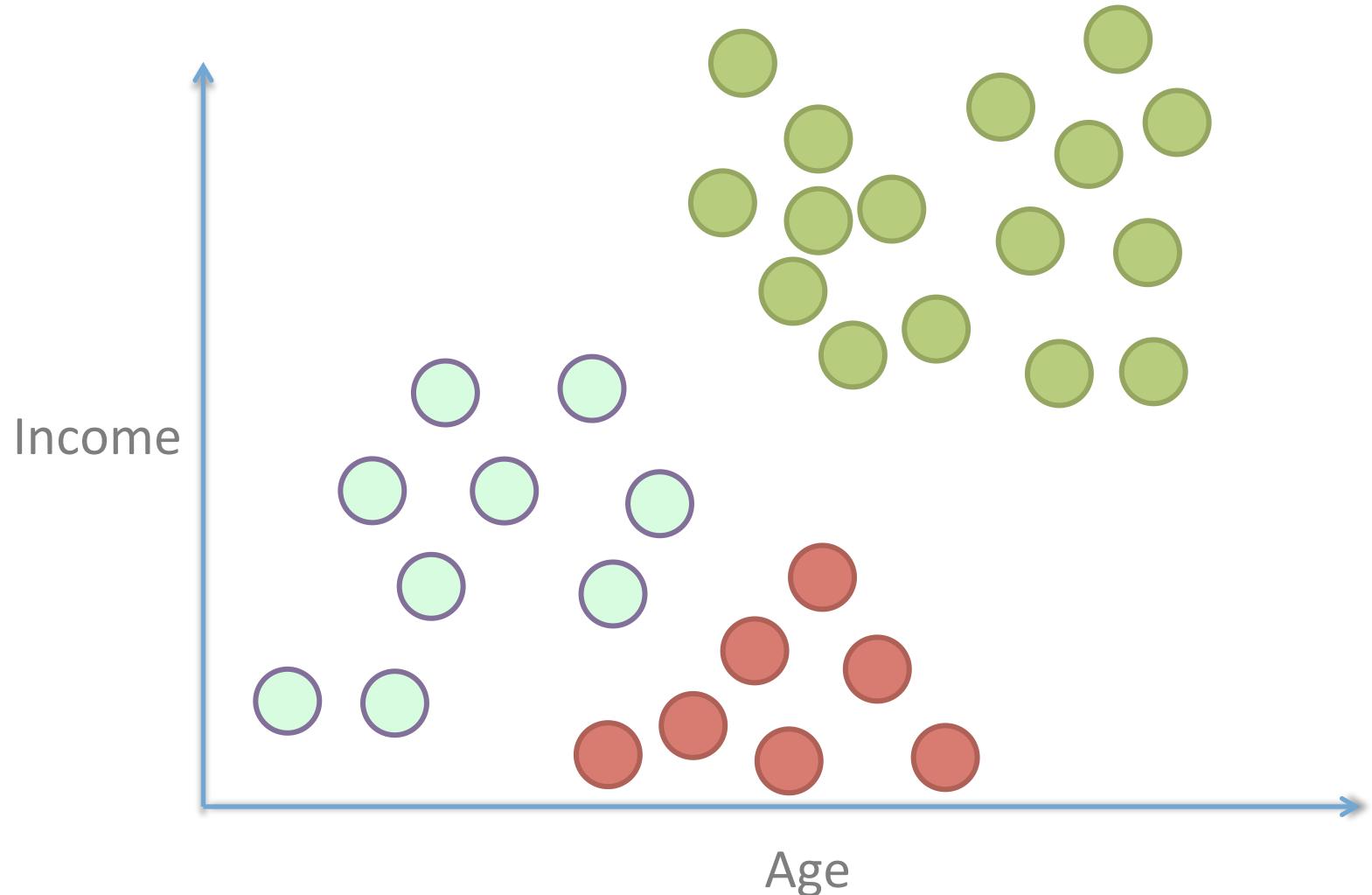
Agglomerative Hierarchical Clustering



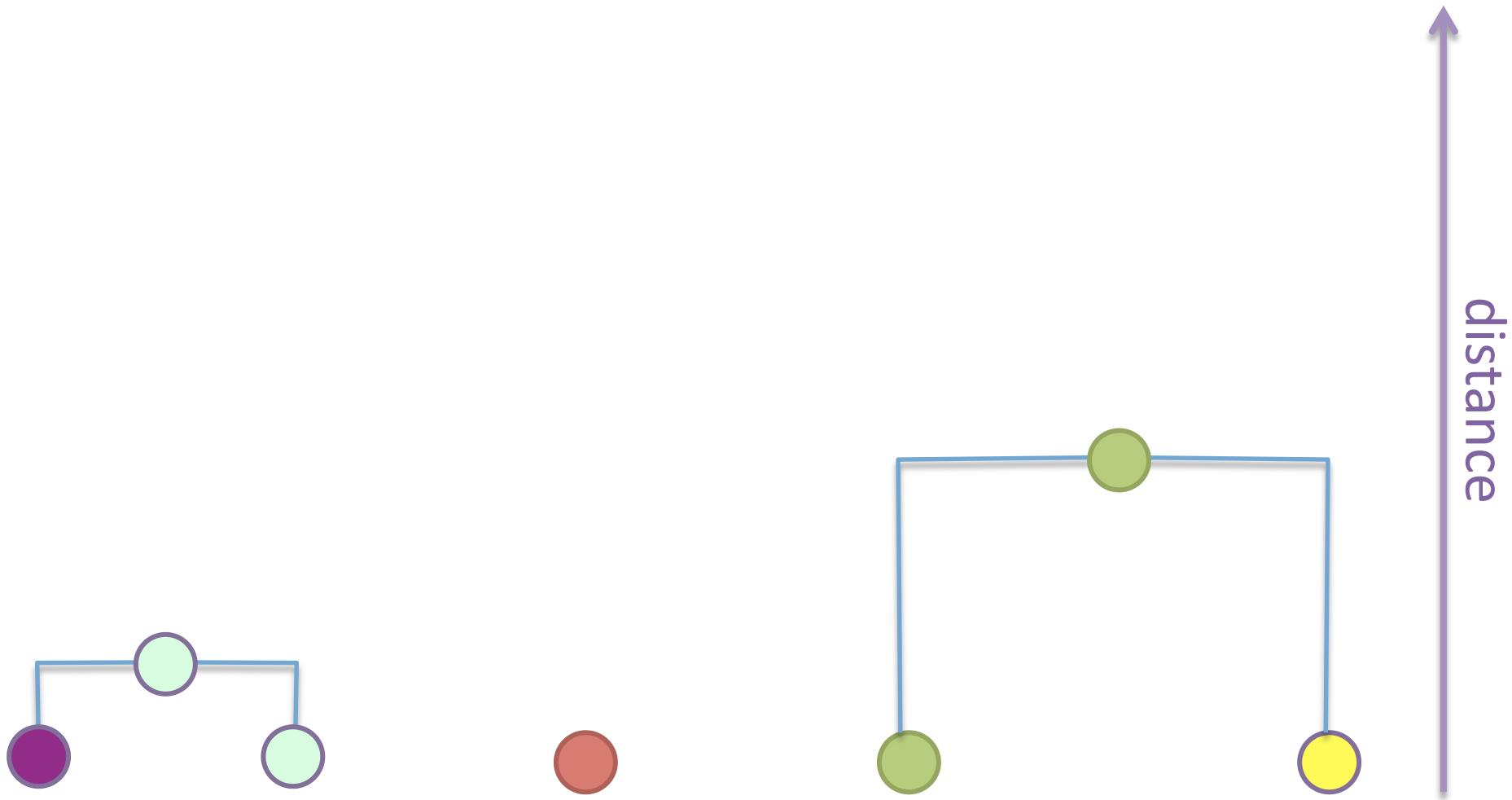
Agglomerative Hierarchical Clustering



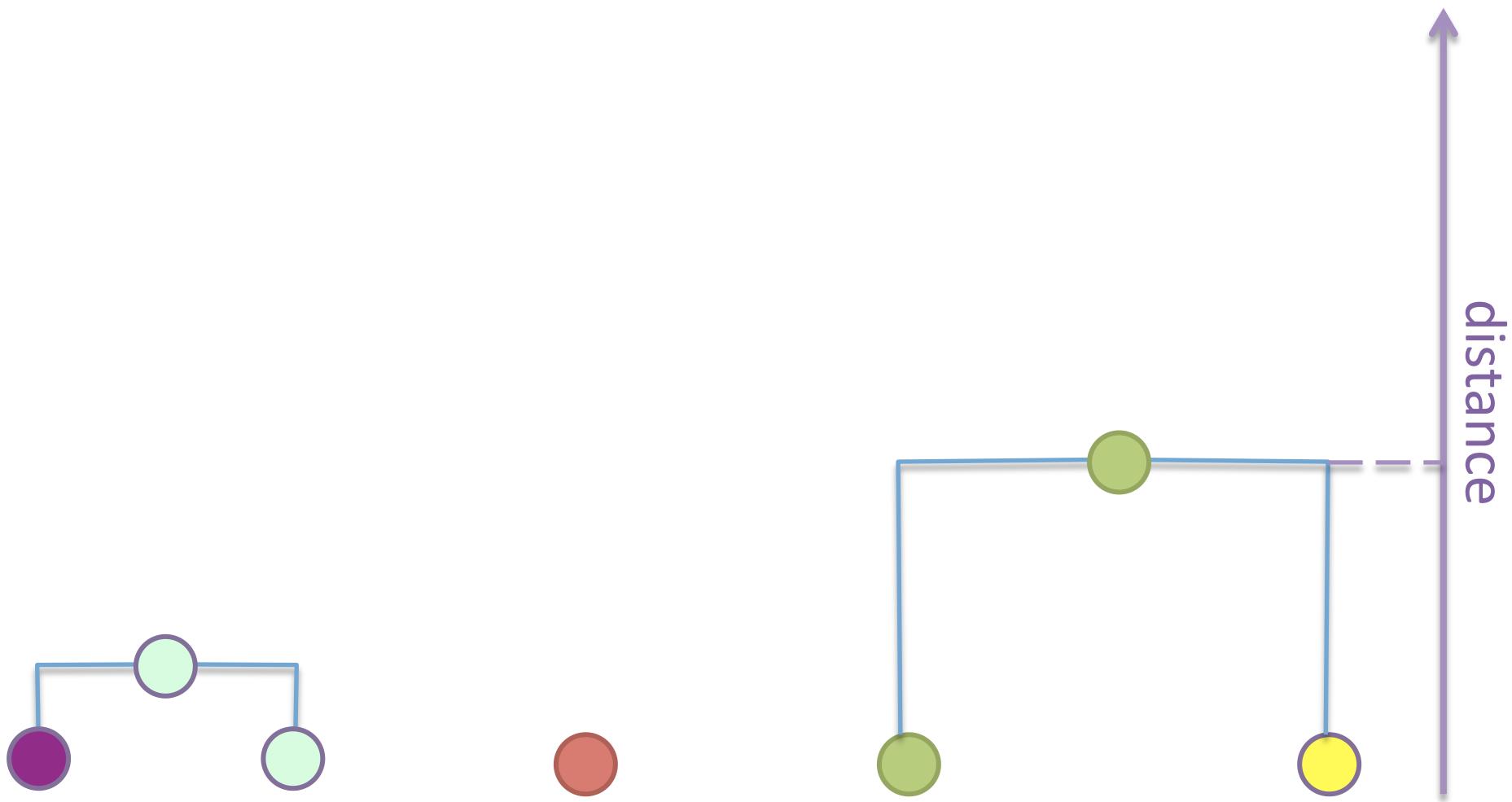
Agglomerative Hierarchical Clustering



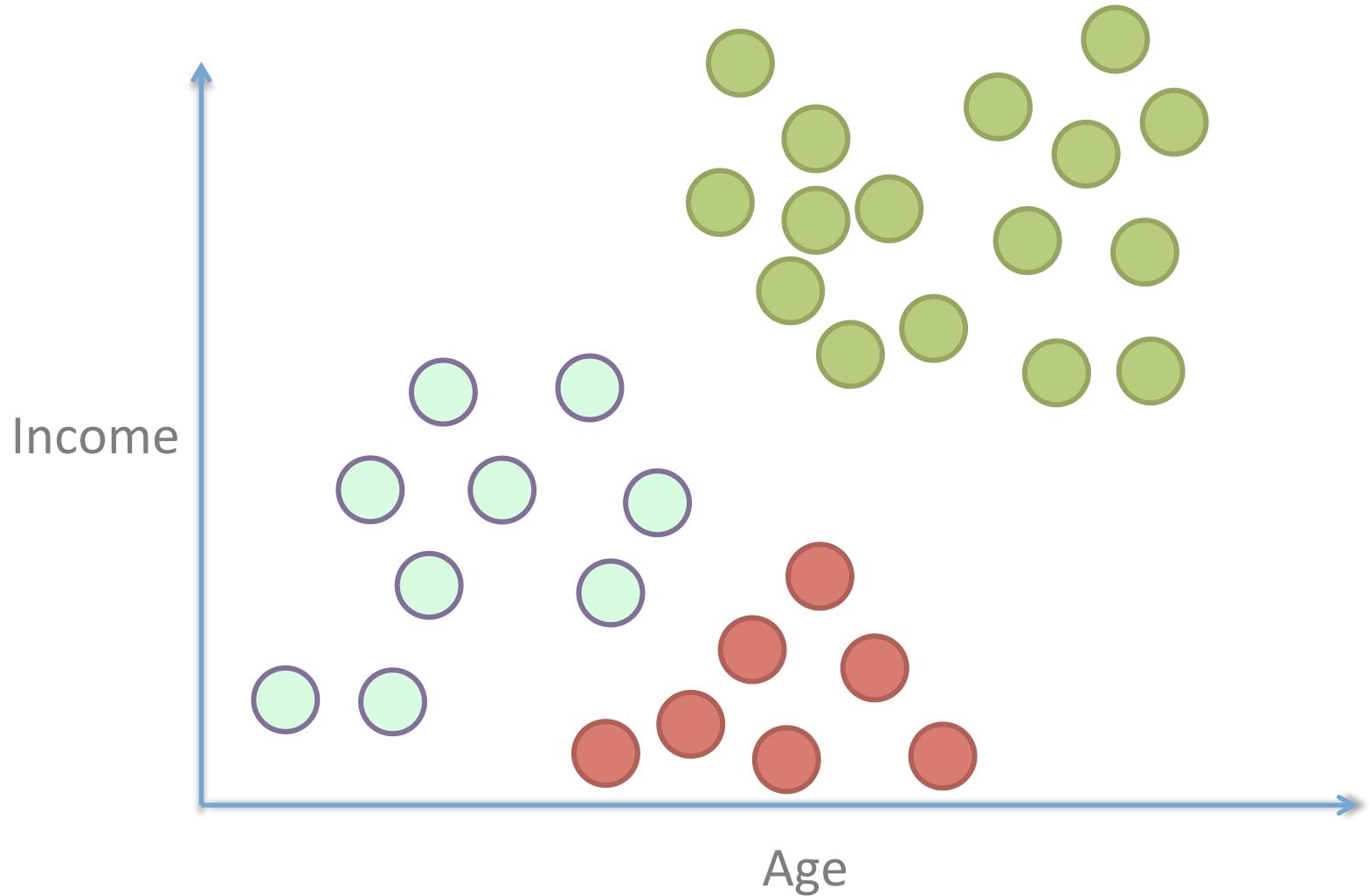
Agglomerative Hierarchical Clustering



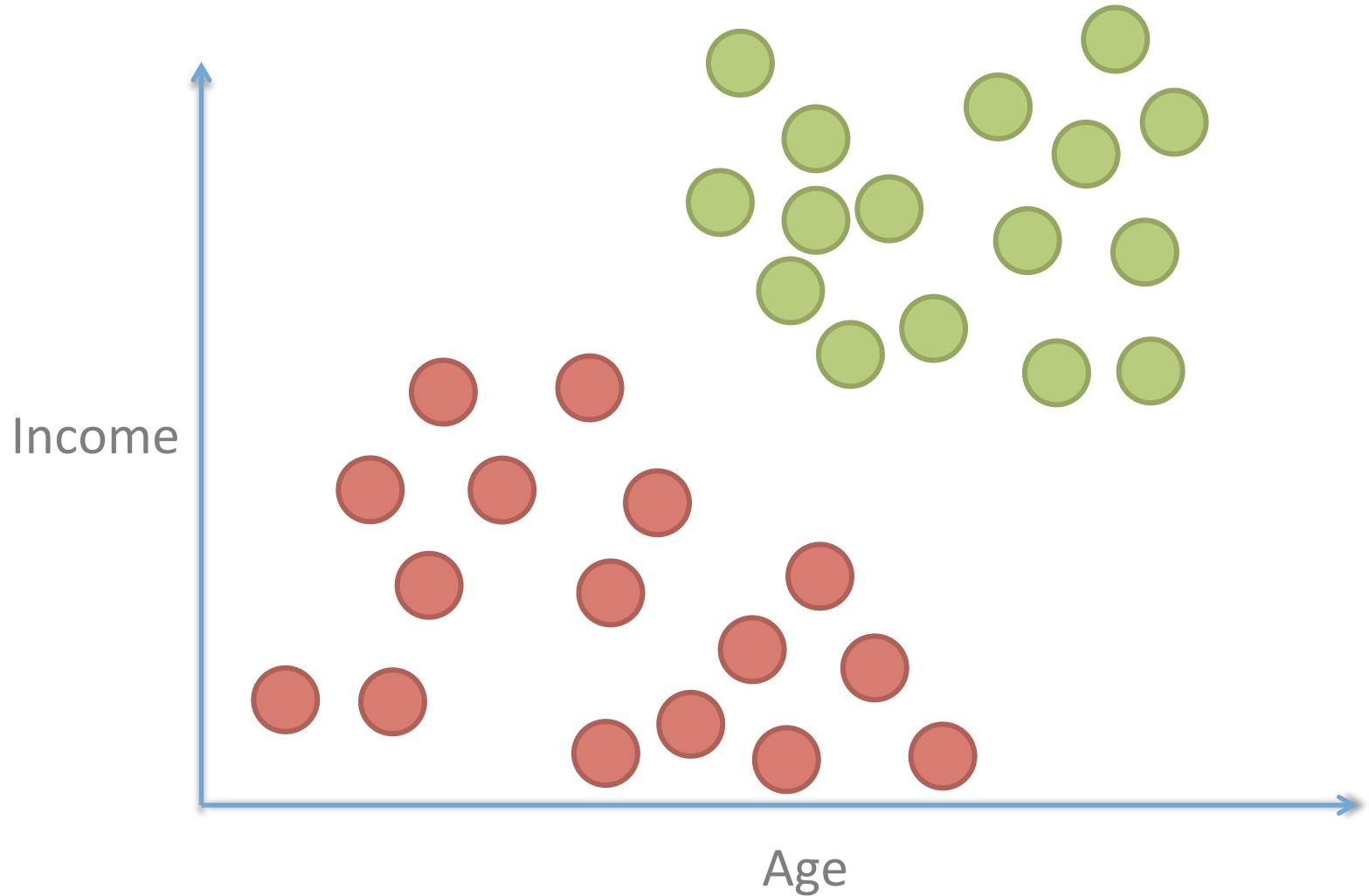
Agglomerative Hierarchical Clustering



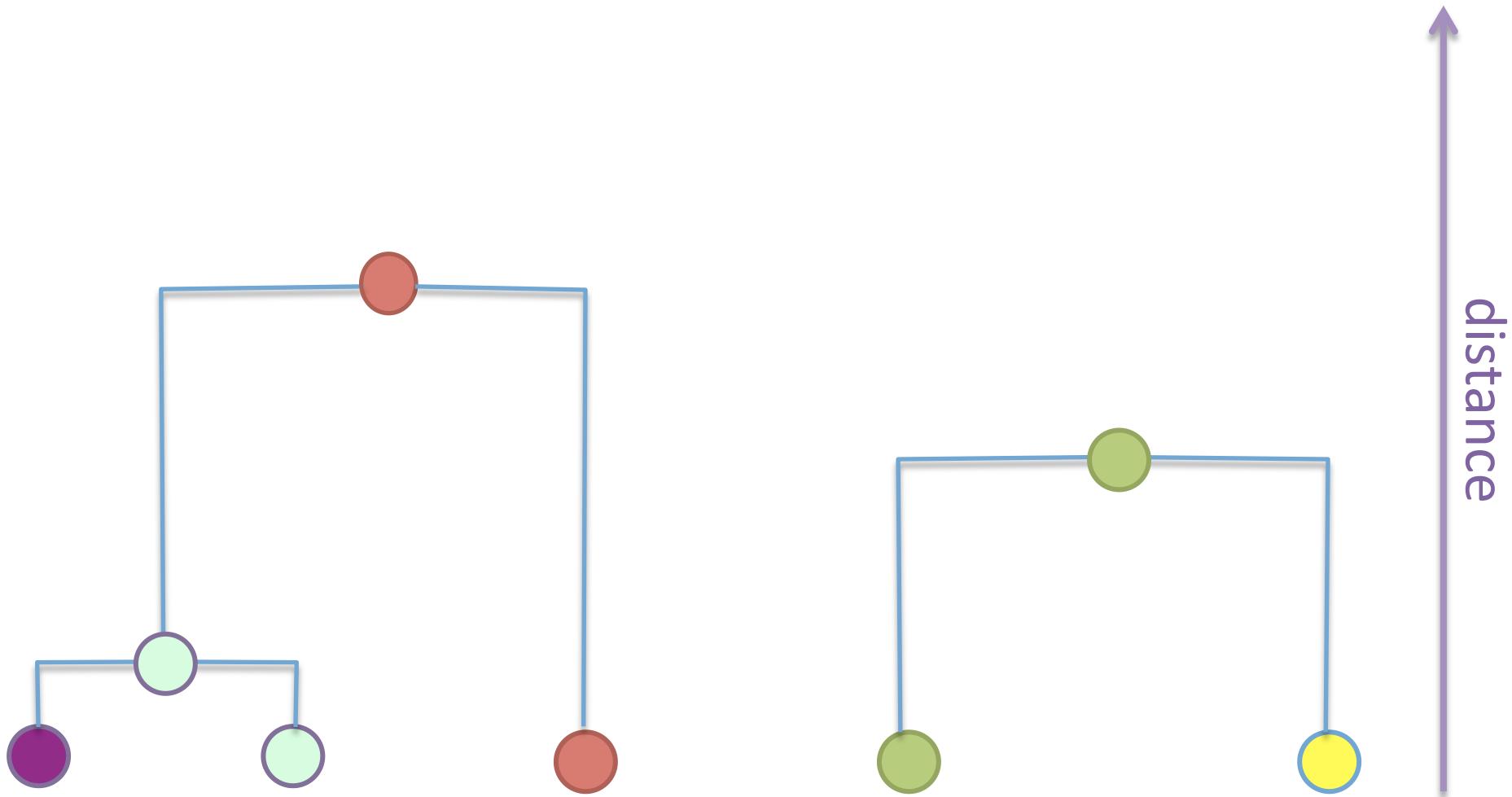
Agglomerative Hierarchical Clustering



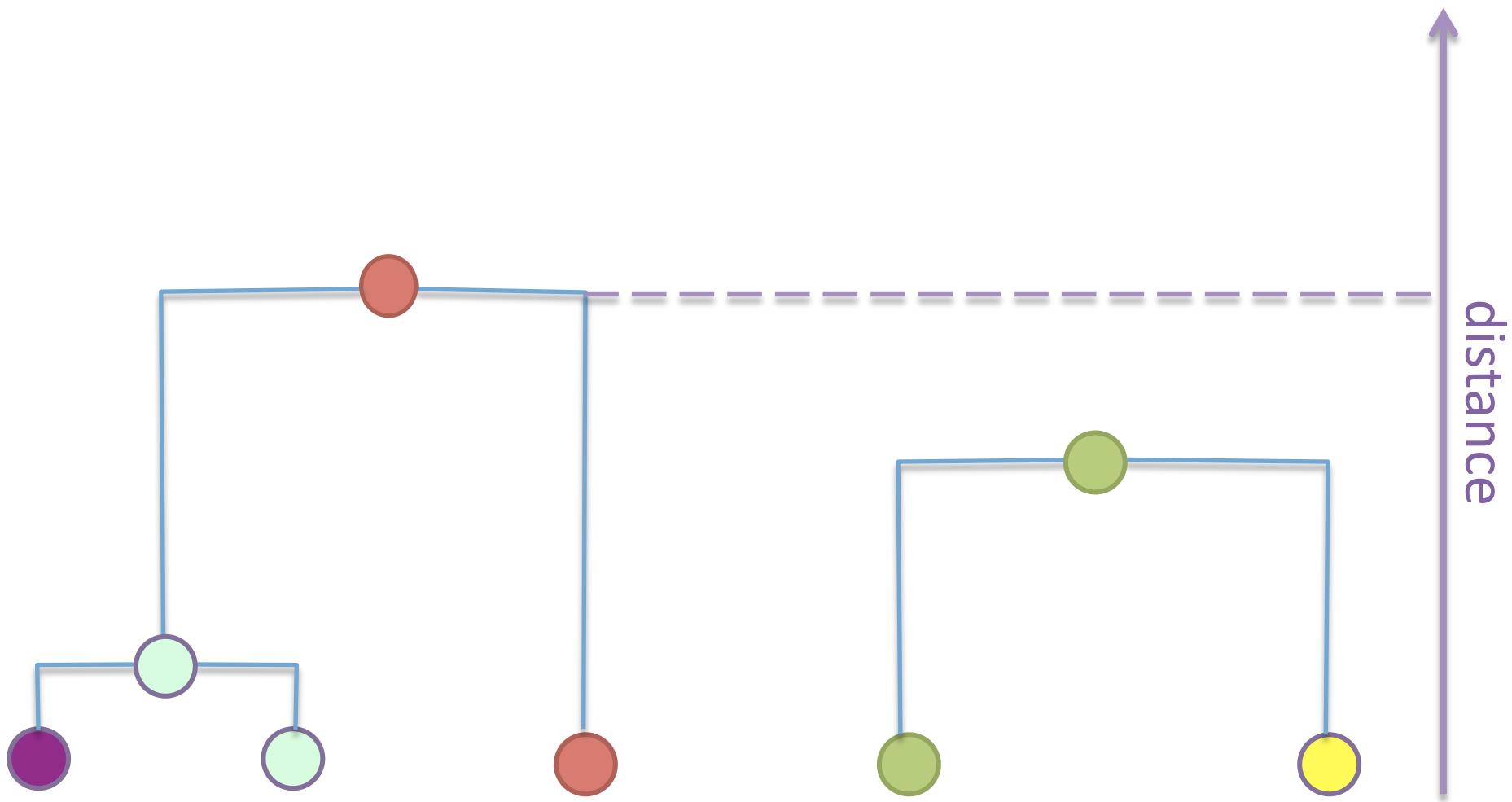
Agglomerative Hierarchical Clustering



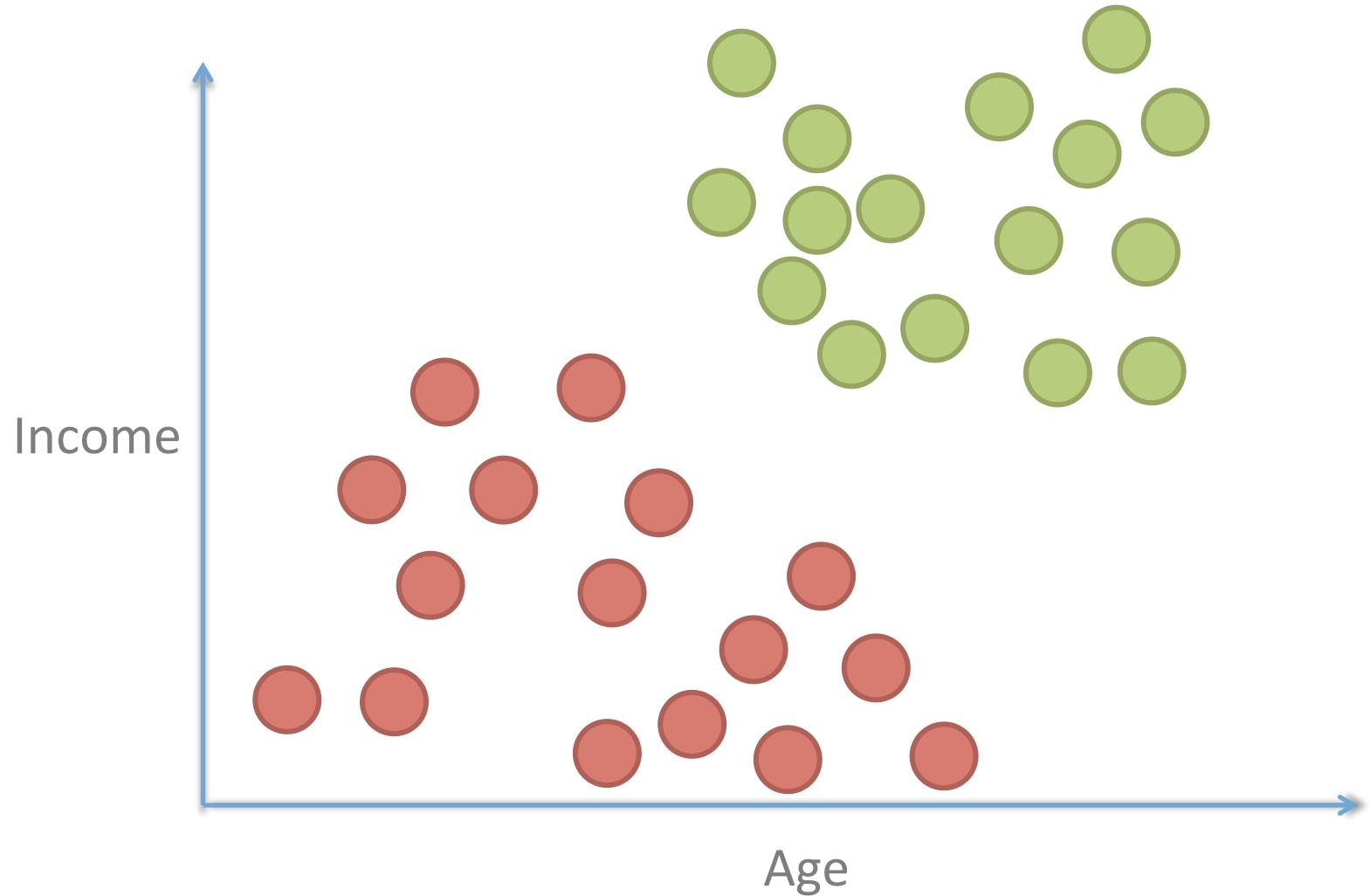
Agglomerative Hierarchical Clustering



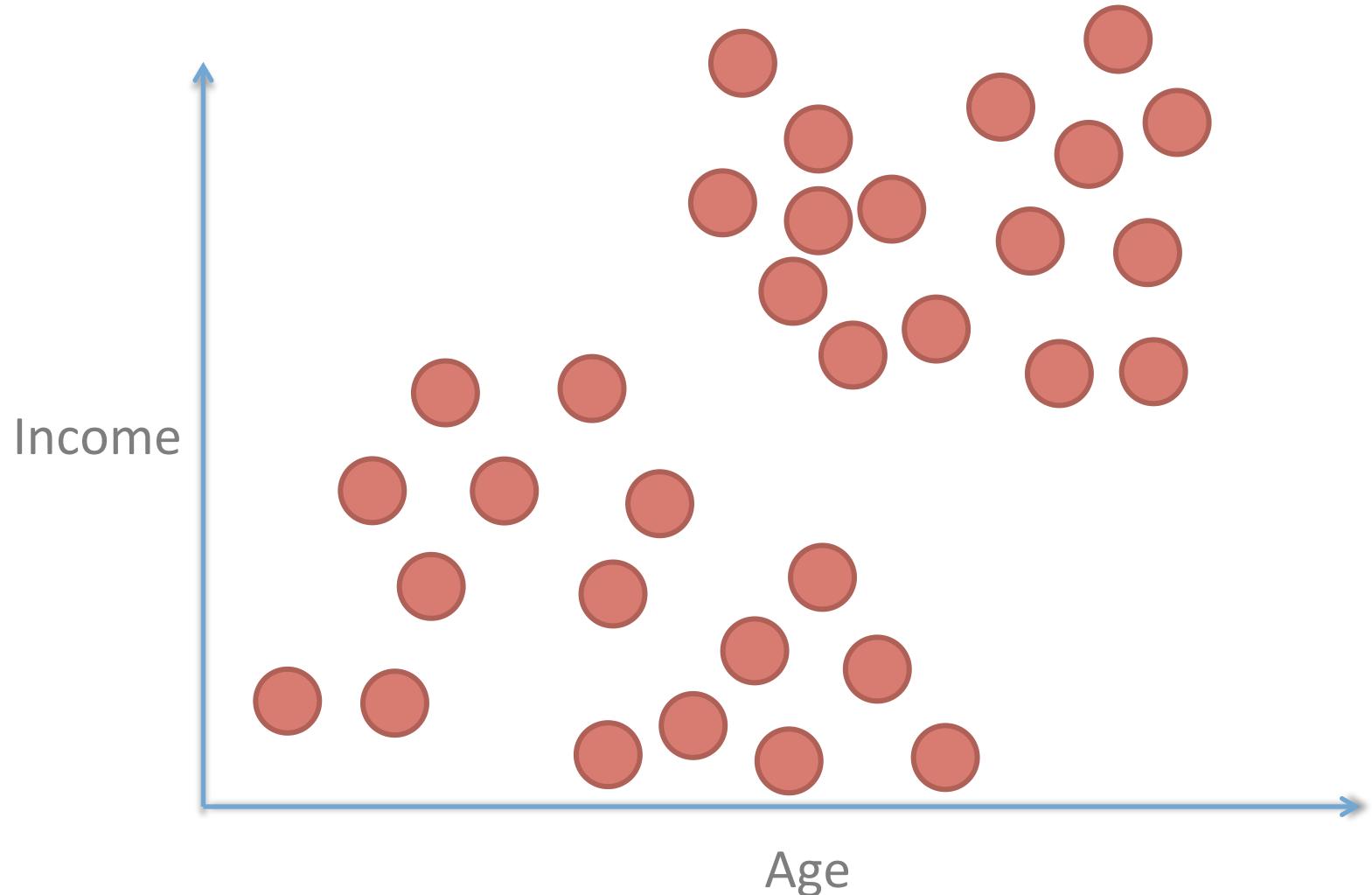
Agglomerative Hierarchical Clustering



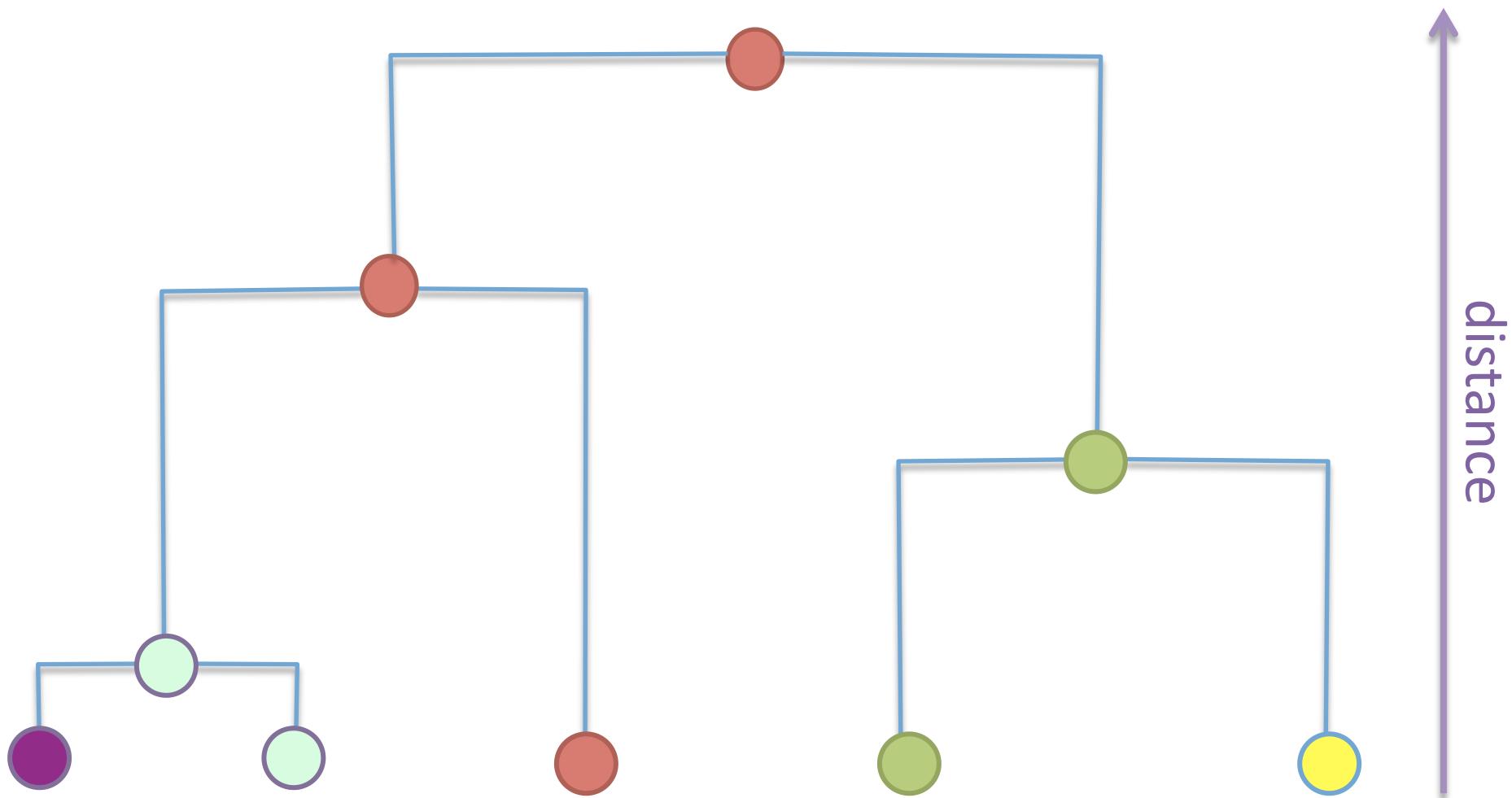
Agglomerative Hierarchical Clustering



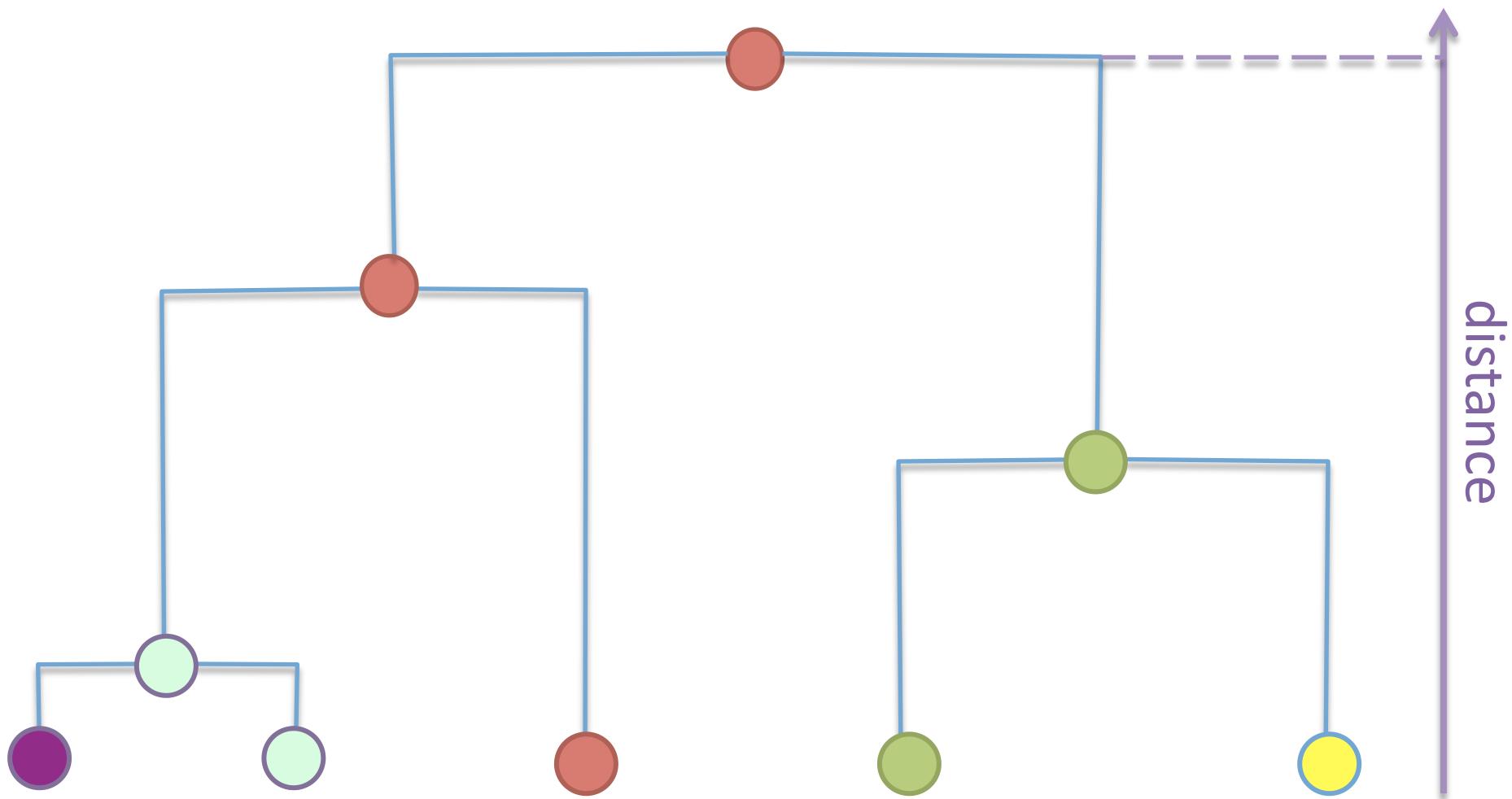
Agglomerative Hierarchical Clustering



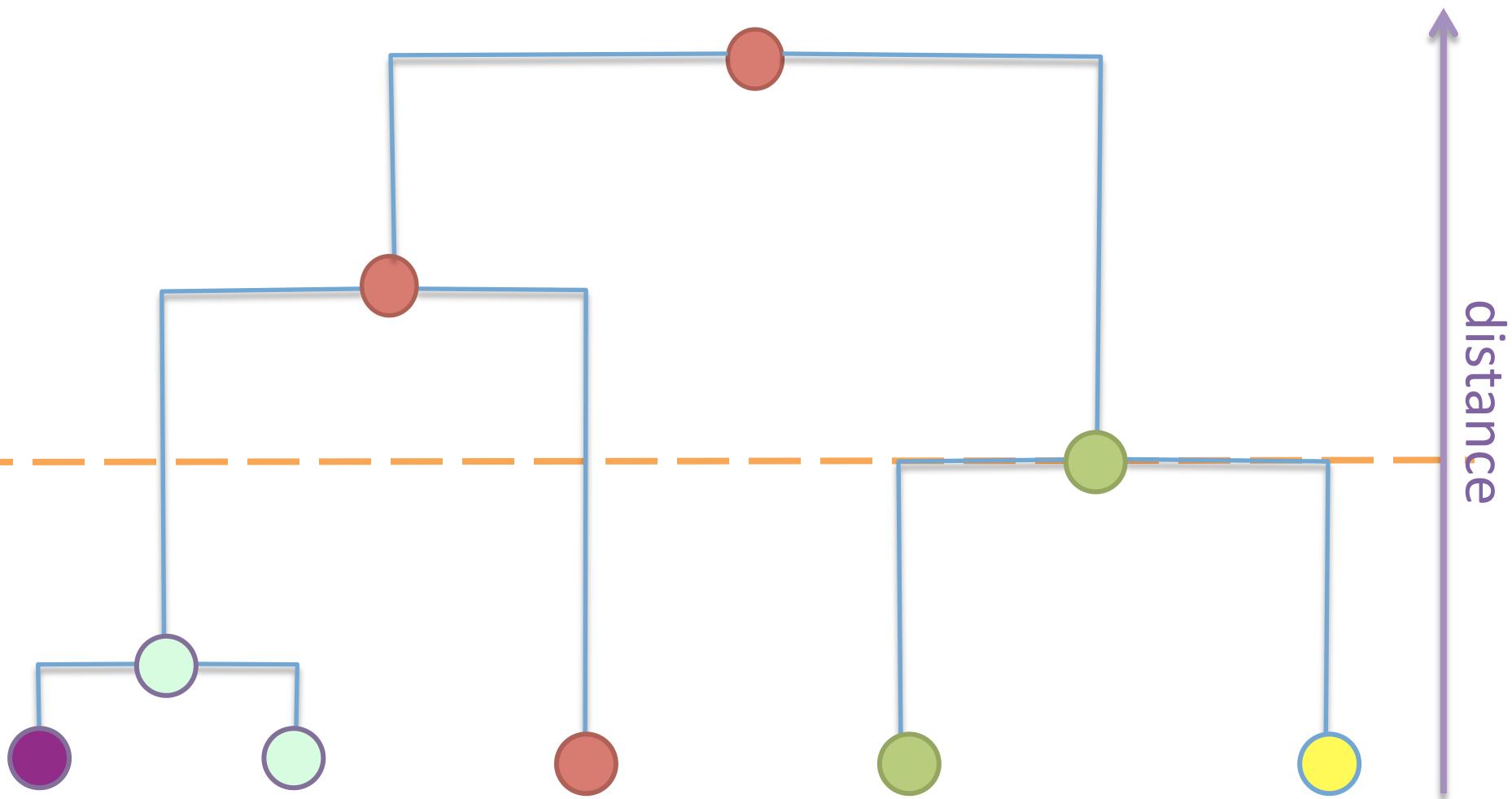
Agglomerative Hierarchical Clustering



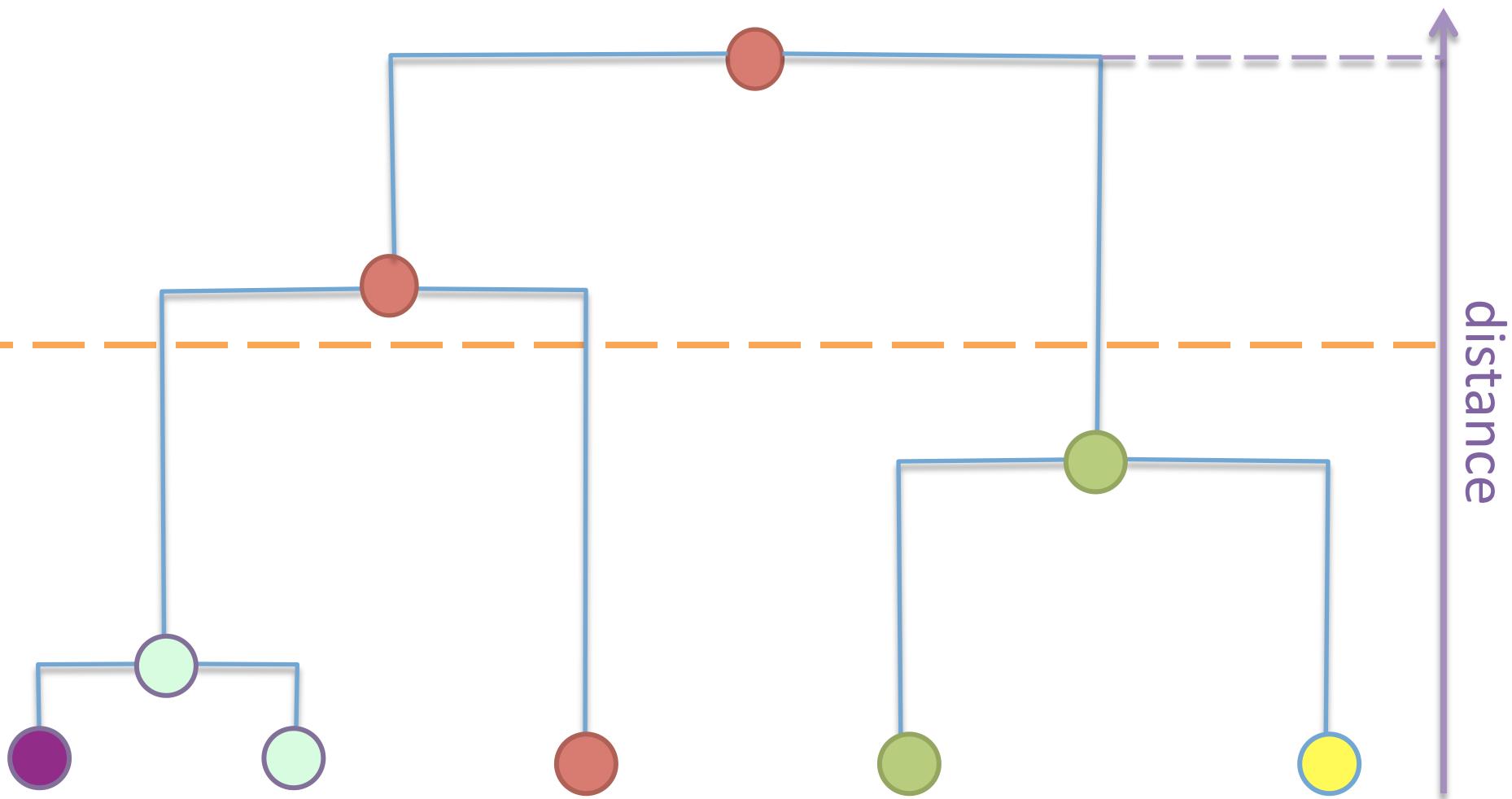
Agglomerative Hierarchical Clustering



Criterion: Stop at $k=3$ clusters!



Criterion: Stop at $\text{dist}=3\dots\dots!$



```
from sklearn.cluster import AgglomerativeClustering
```

```
AgglomerativeClustering(...,  
...,  
linkage = "ward",  
...,  
affinity = "euclidian",  
....)
```

```
from sklearn.cluster import AgglomerativeClustering
```

```
AgglomerativeClustering(....,
```

```
    ....,
```

```
    linkage = "ward",
```

```
    ....,
```

```
    affinity = "euclidian",
```

```
    ....)
```

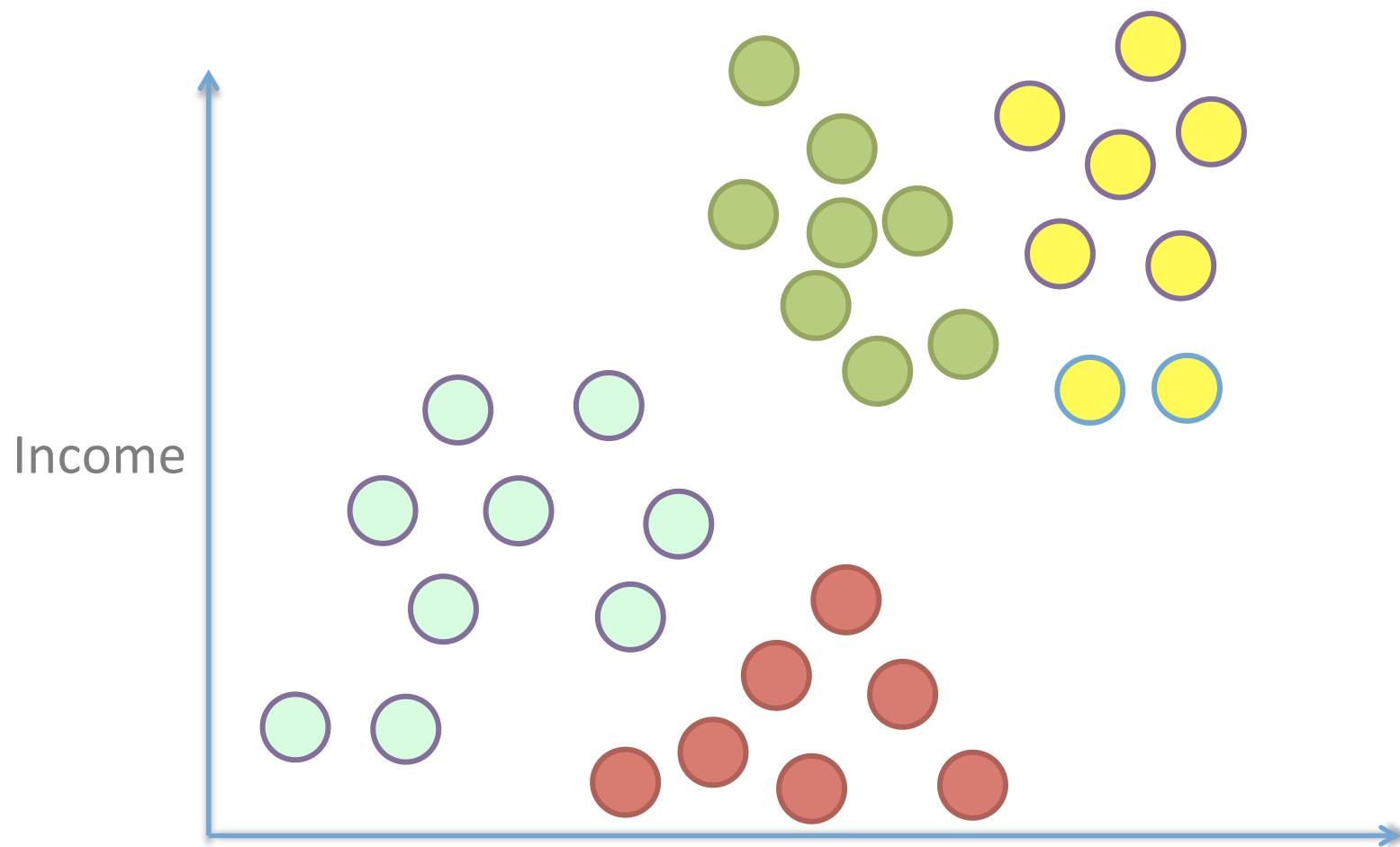
Agglomerative Clustering linkage

Which clusters to merge next.

Definition of “shortest distance”

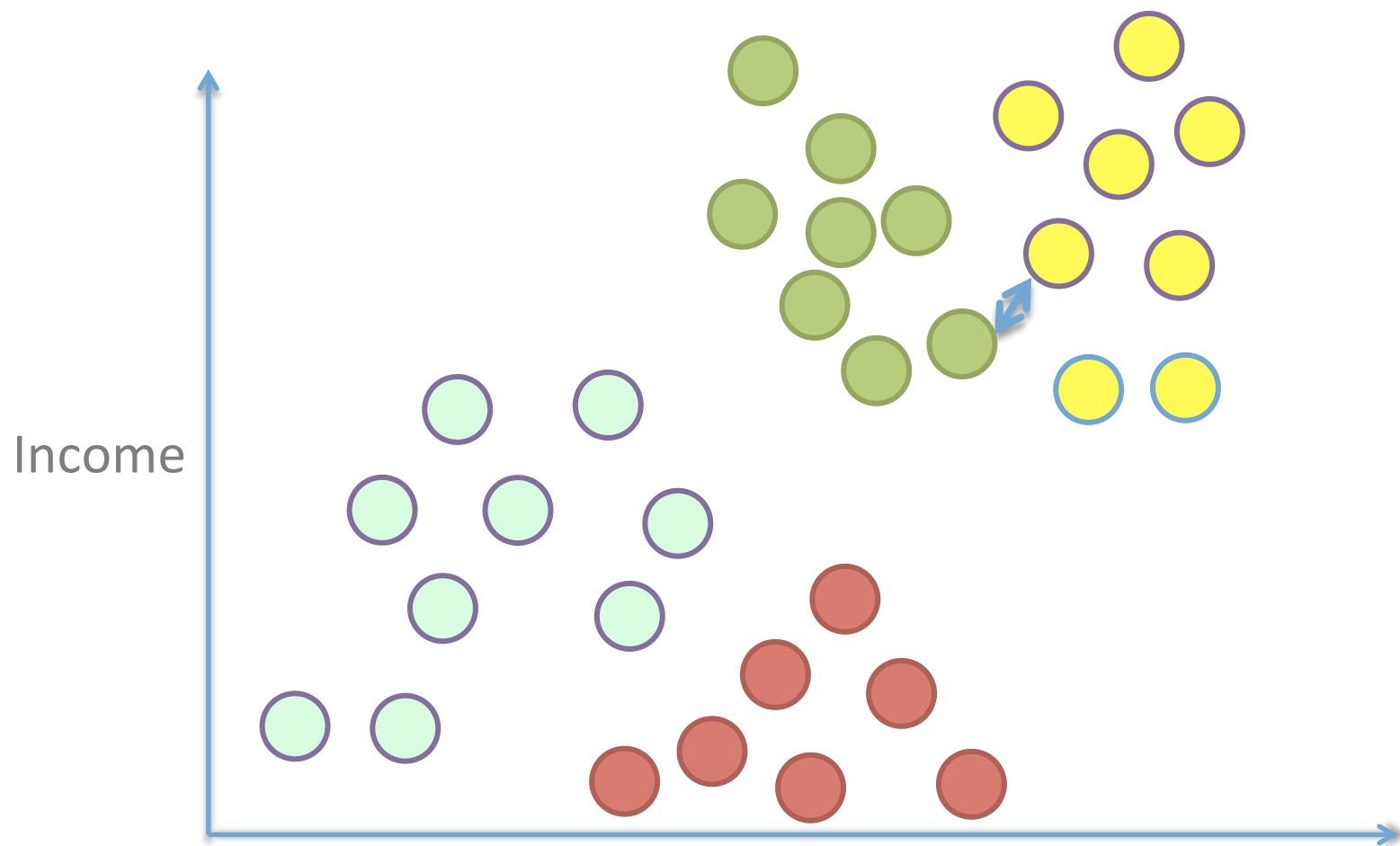
linkage = “single”

Distance between two cluster is the minimum pairwise distance



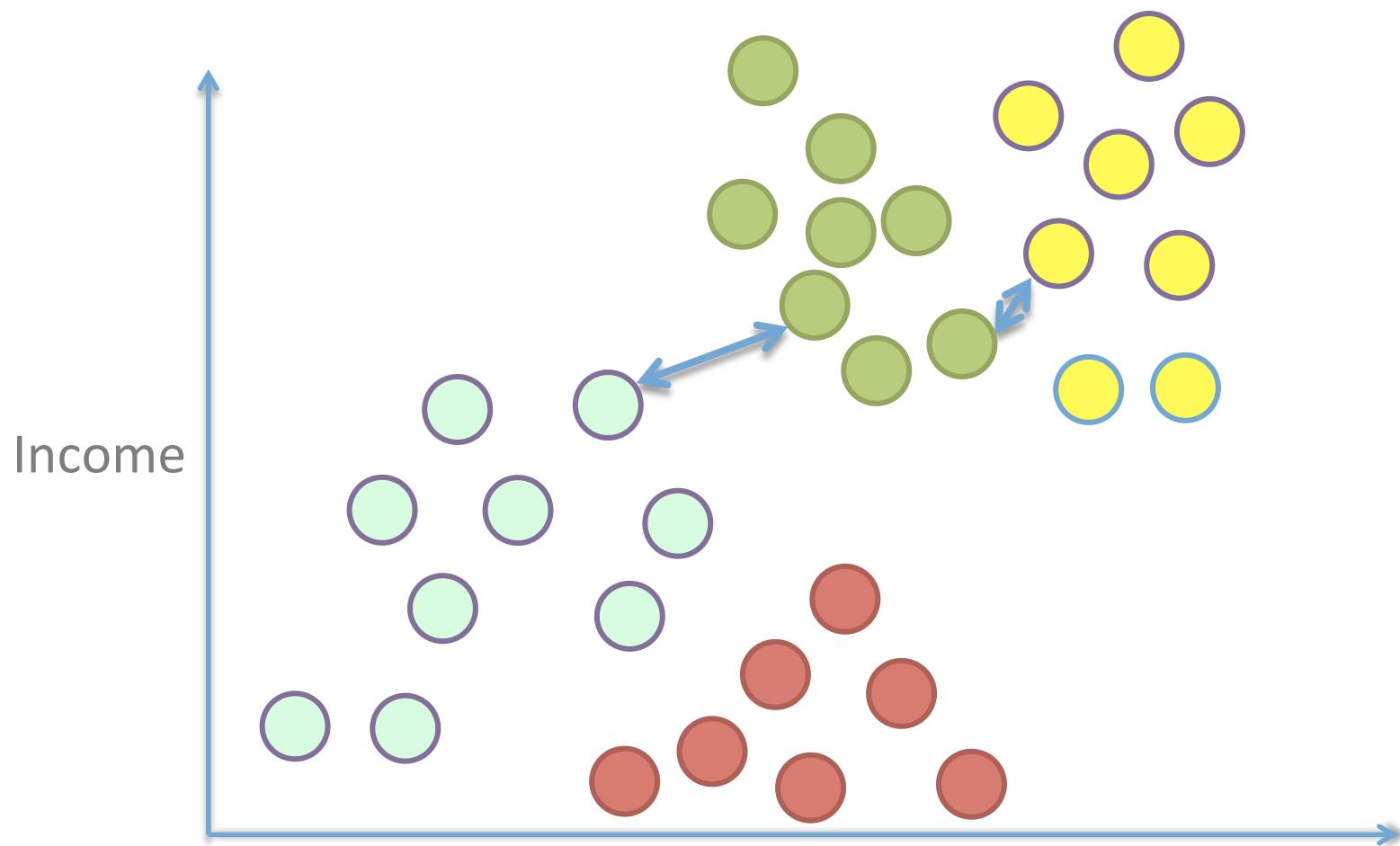
linkage = “single”

Distance between two cluster is the minimum pairwise distance



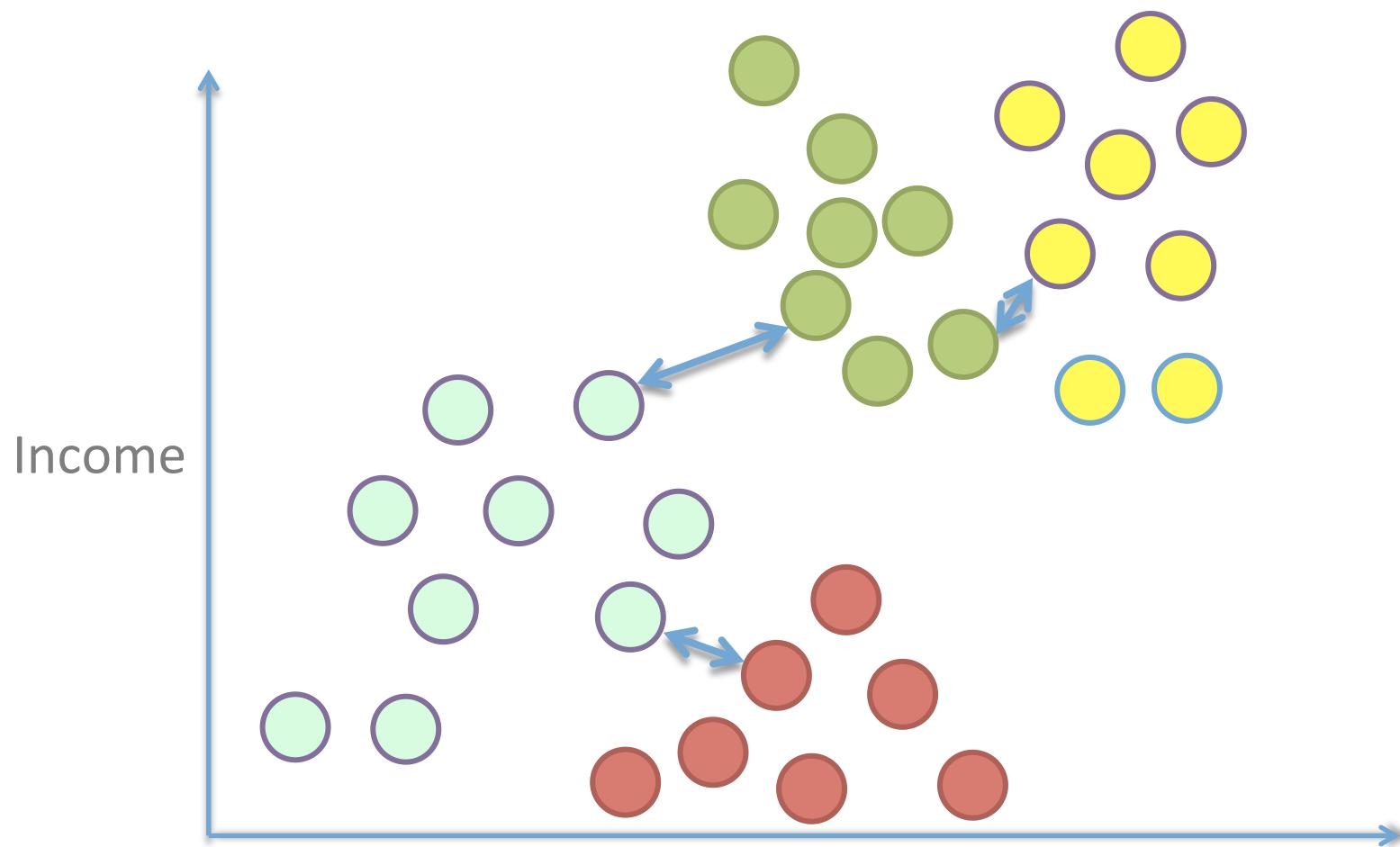
linkage = “single”

Distance between two cluster is the minimum pairwise distance



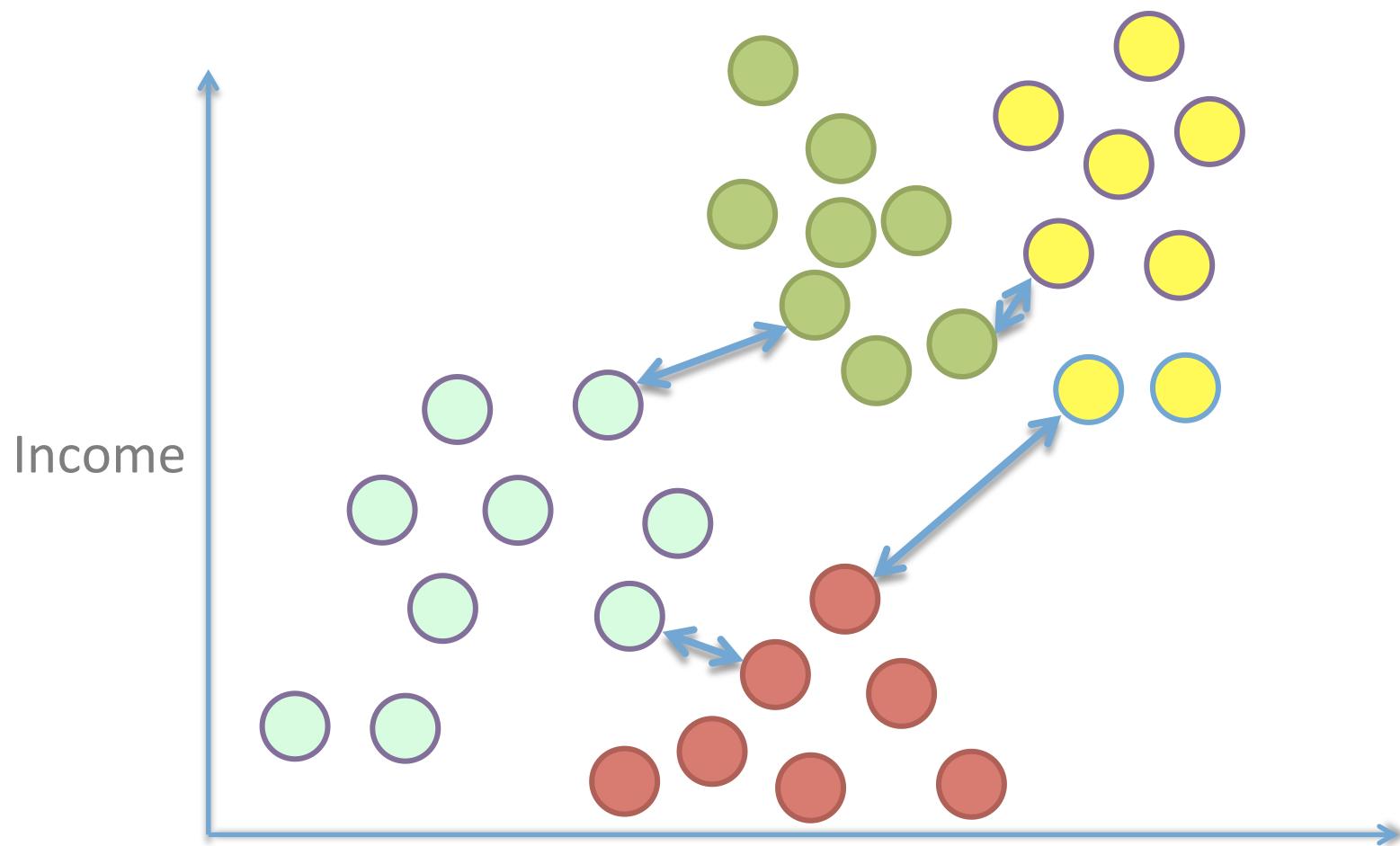
linkage = “single”

Distance between two cluster is the minimum pairwise distance



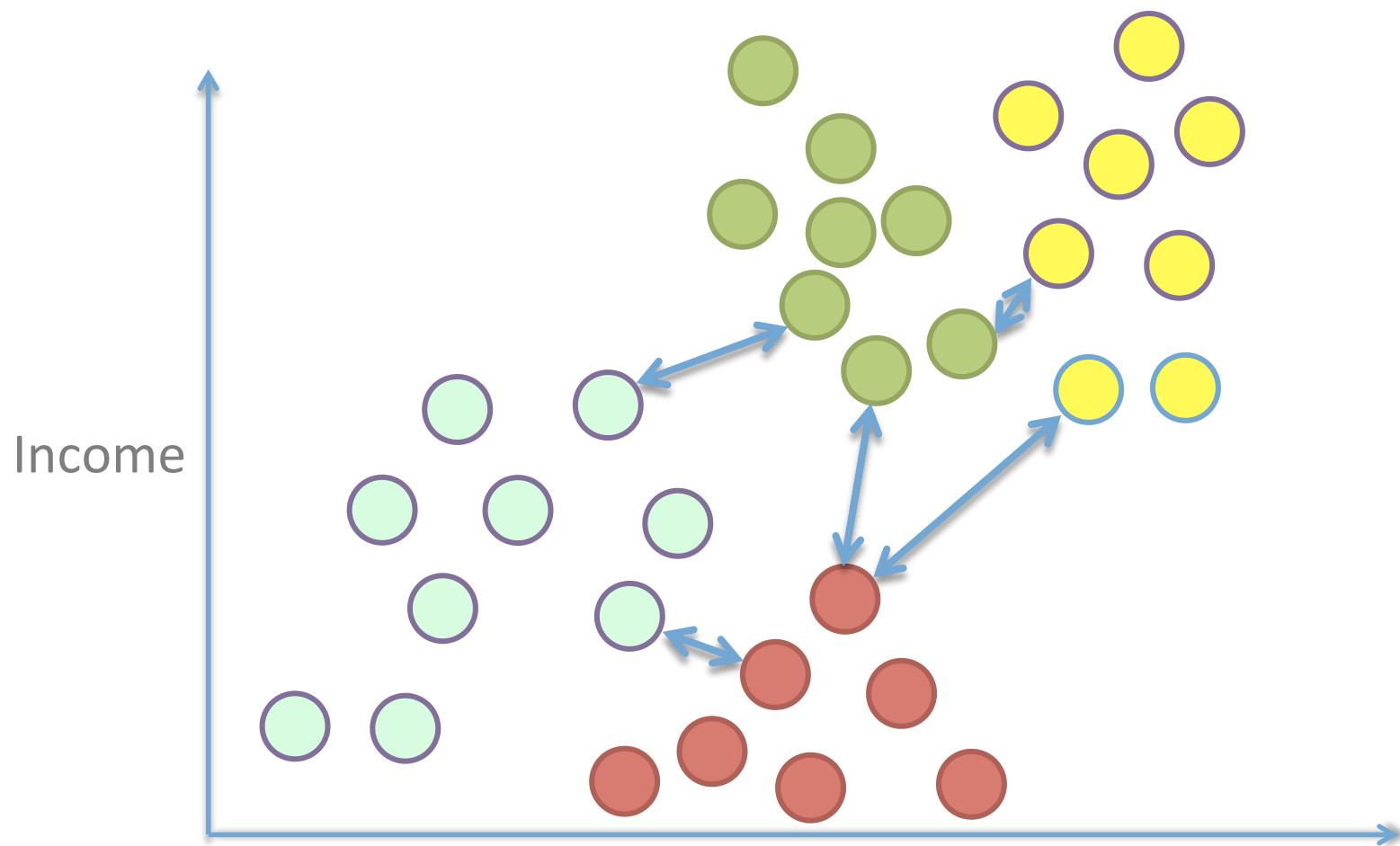
linkage = “single”

Distance between two cluster is the minimum pairwise distance



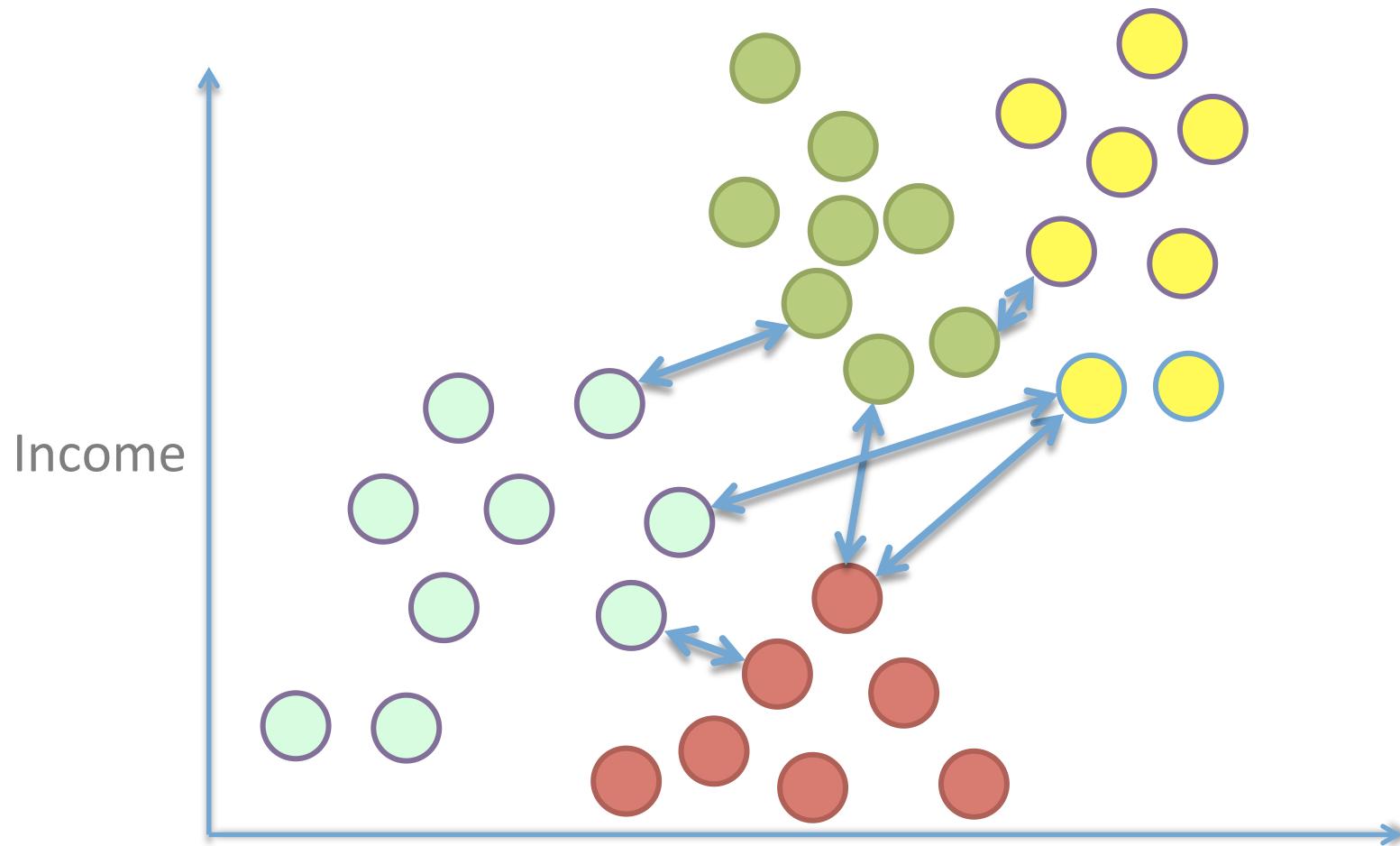
linkage = “single”

Distance between two cluster is the minimum pairwise distance



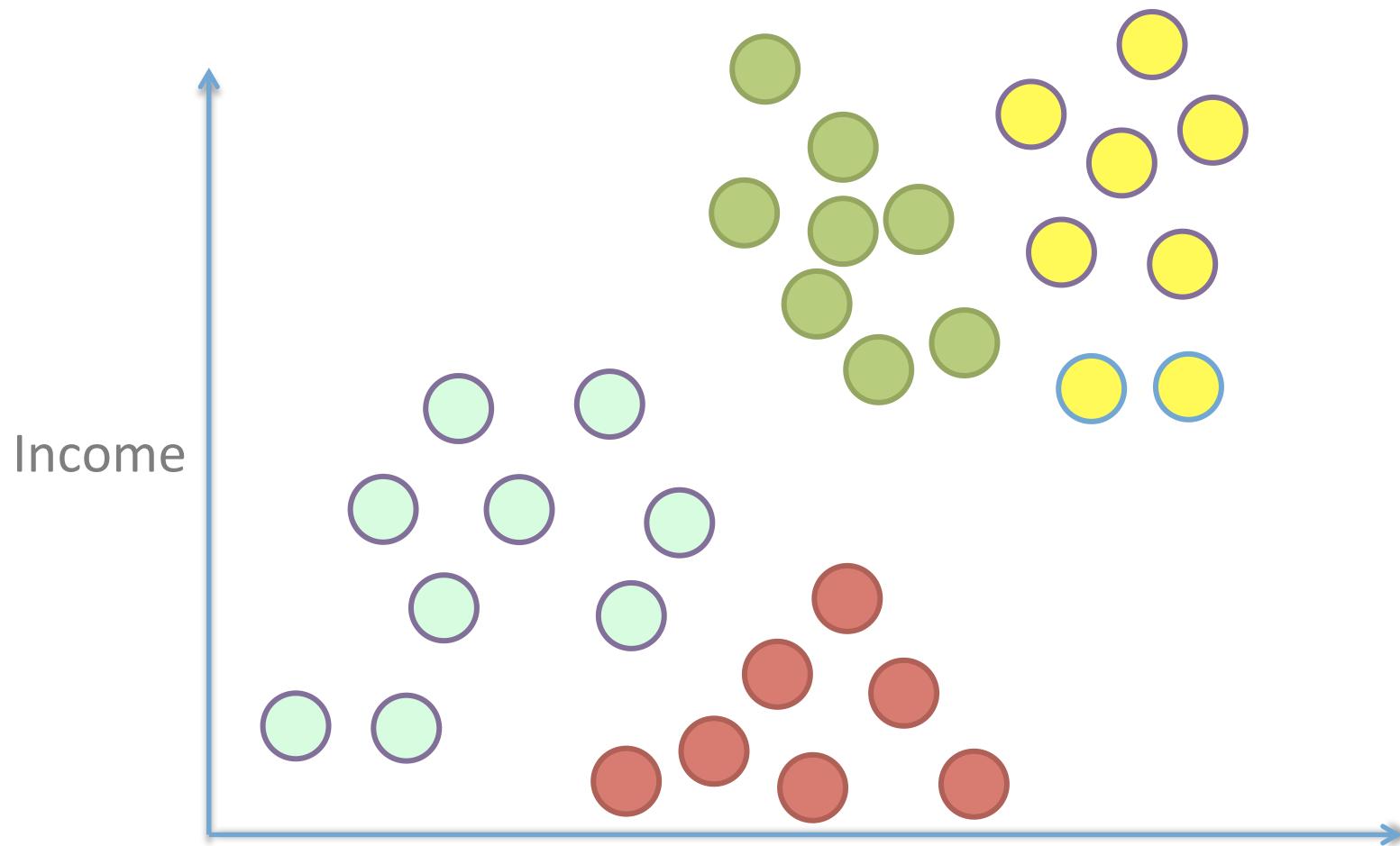
linkage = “single”

Fast. Can form long chains instead of clusters.



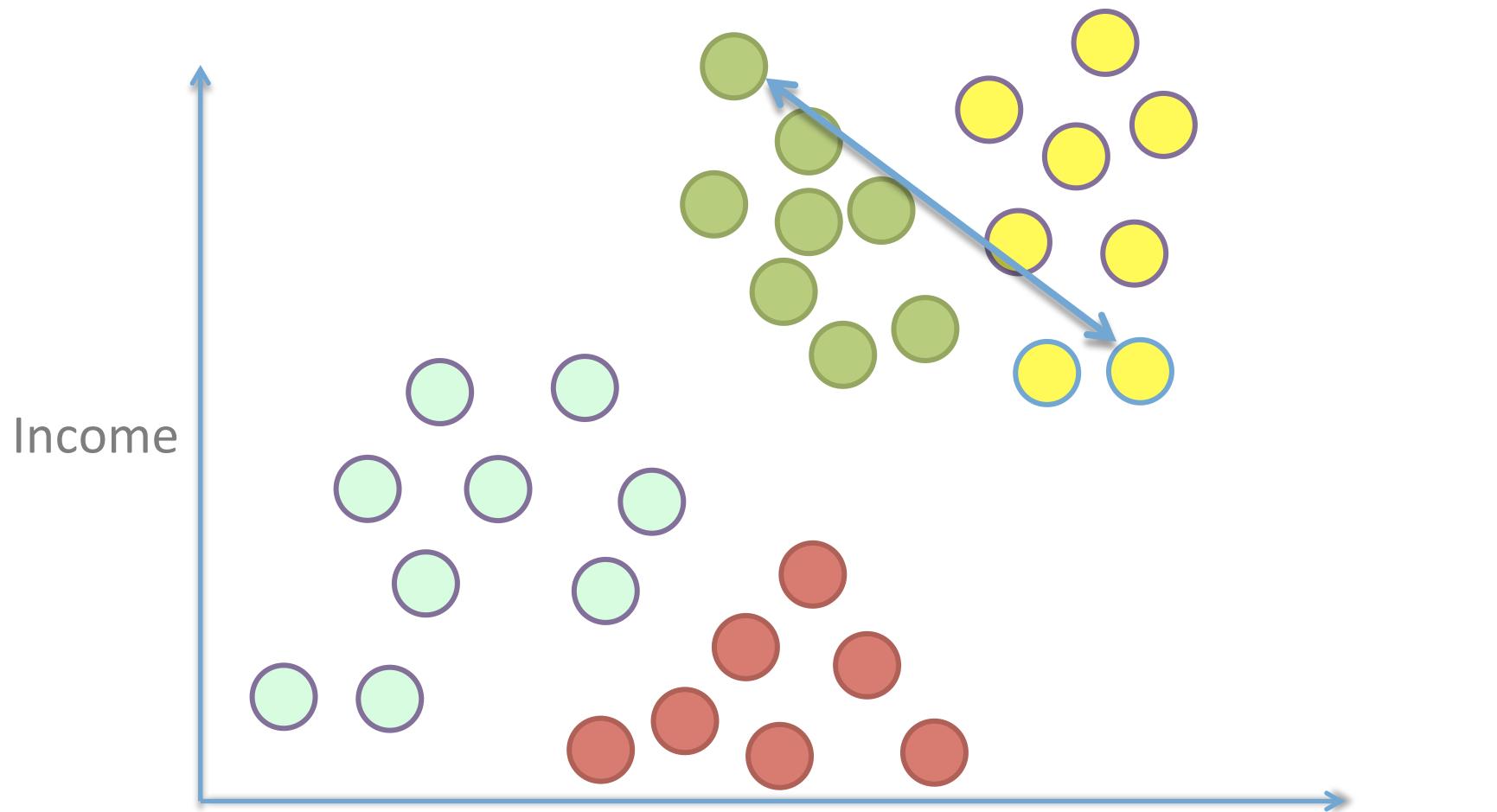
linkage = “complete”

Distance between two clusters is the maximum pairwise distance



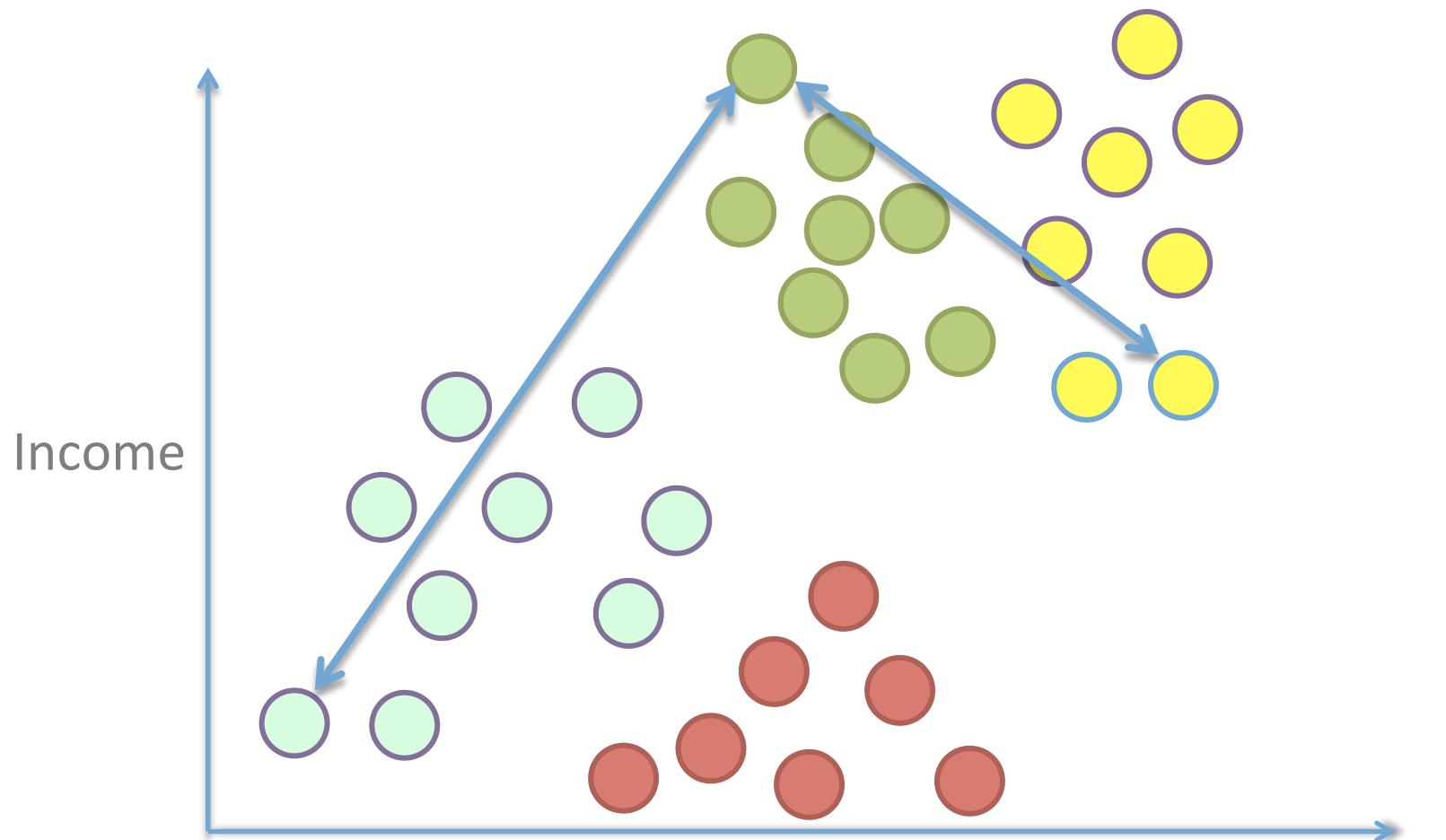
linkage = “complete”

Distance between two clusters is the maximum pairwise distance



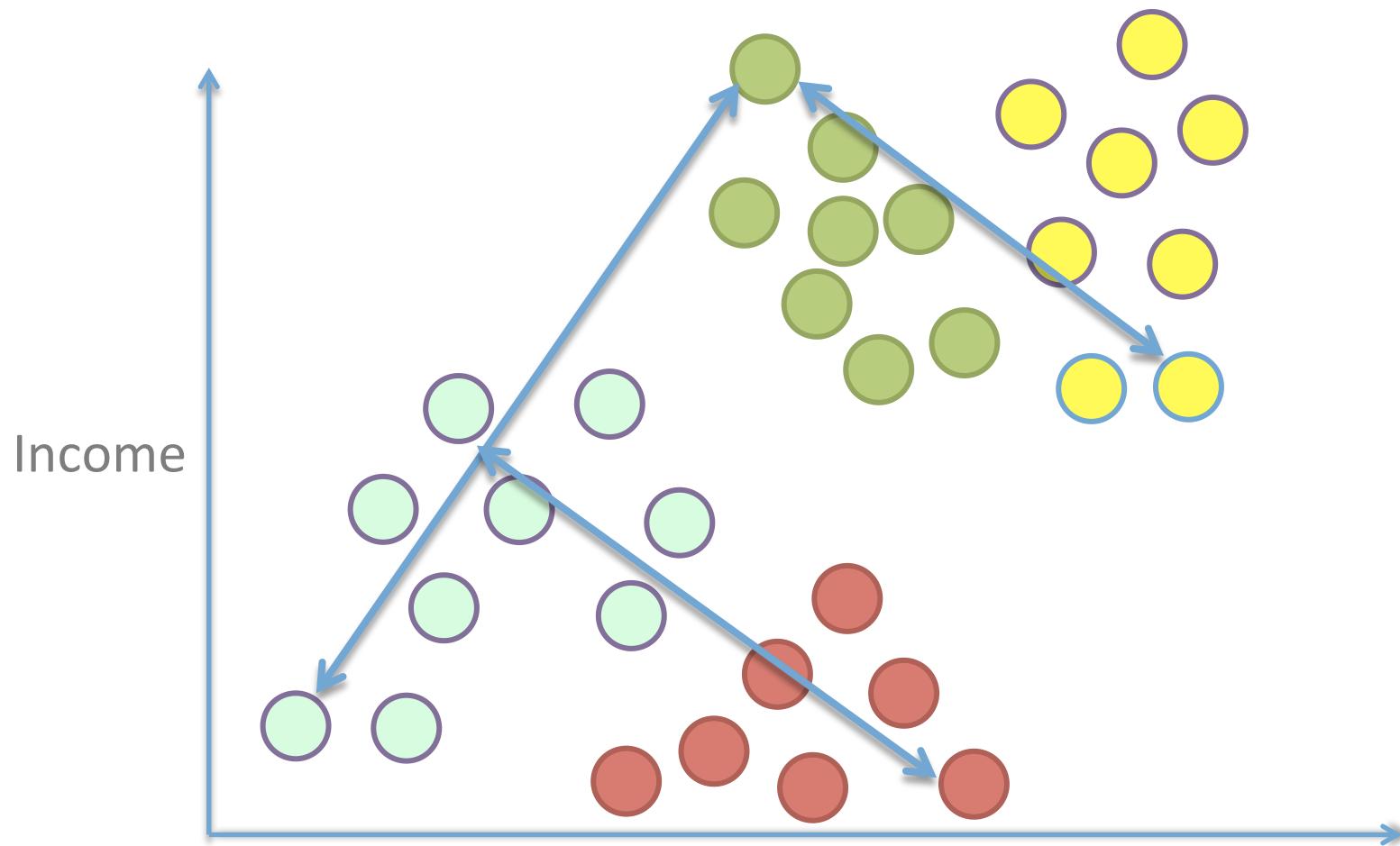
linkage = “complete”

Distance between two clusters is the maximum pairwise distance



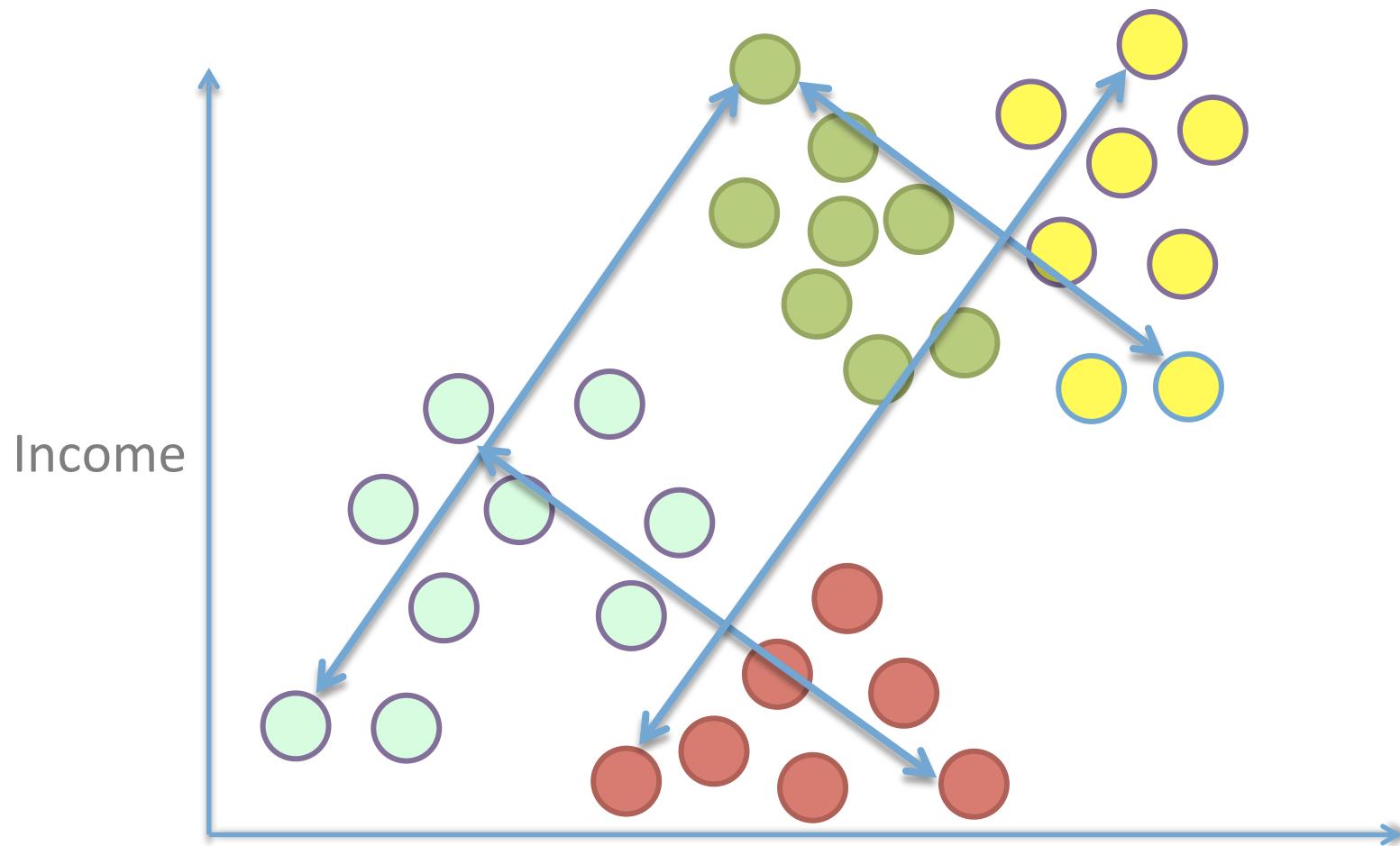
linkage = “complete”

Distance between two clusters is the maximum pairwise distance



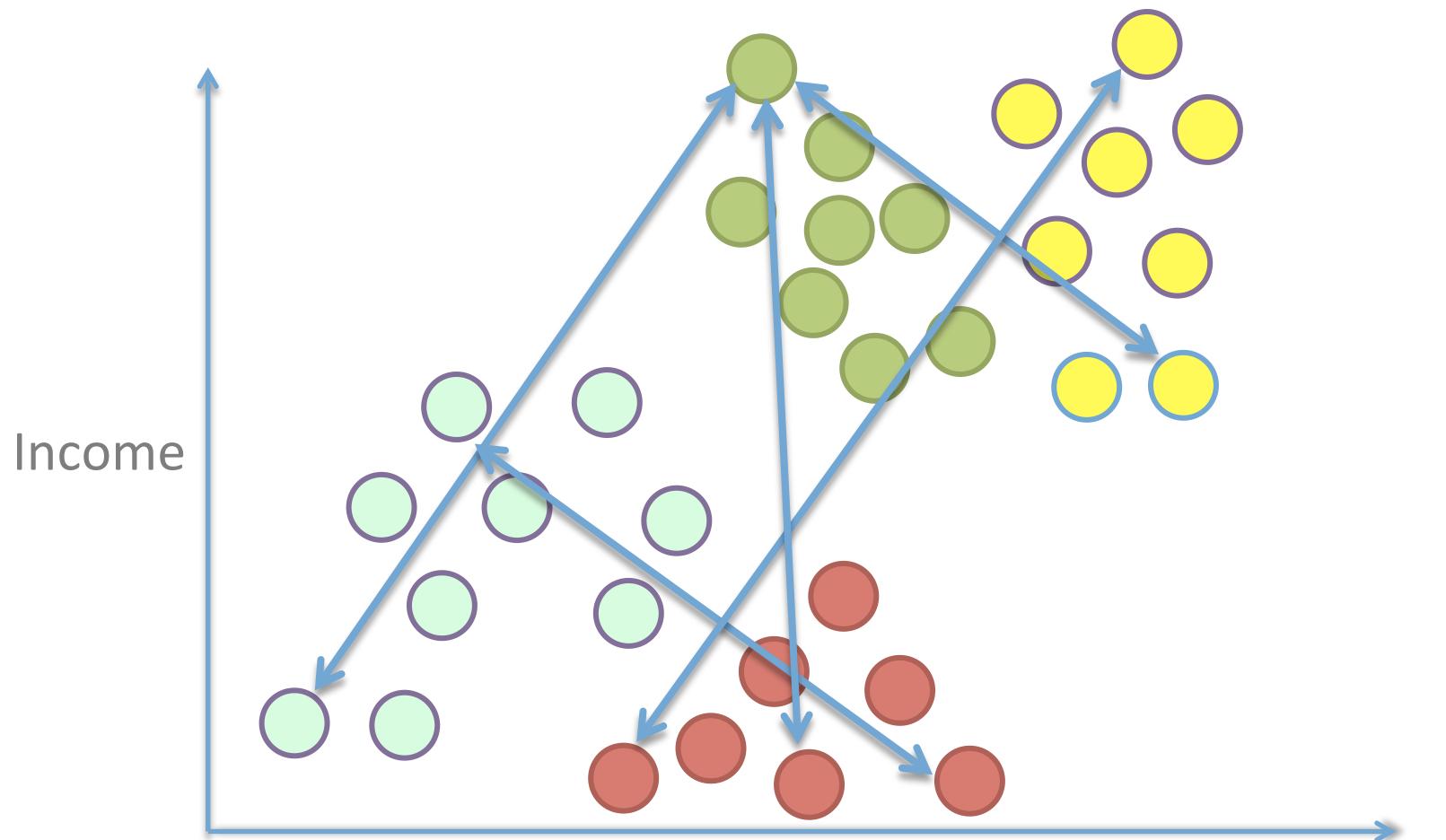
linkage = “complete”

Distance between two clusters is the maximum pairwise distance



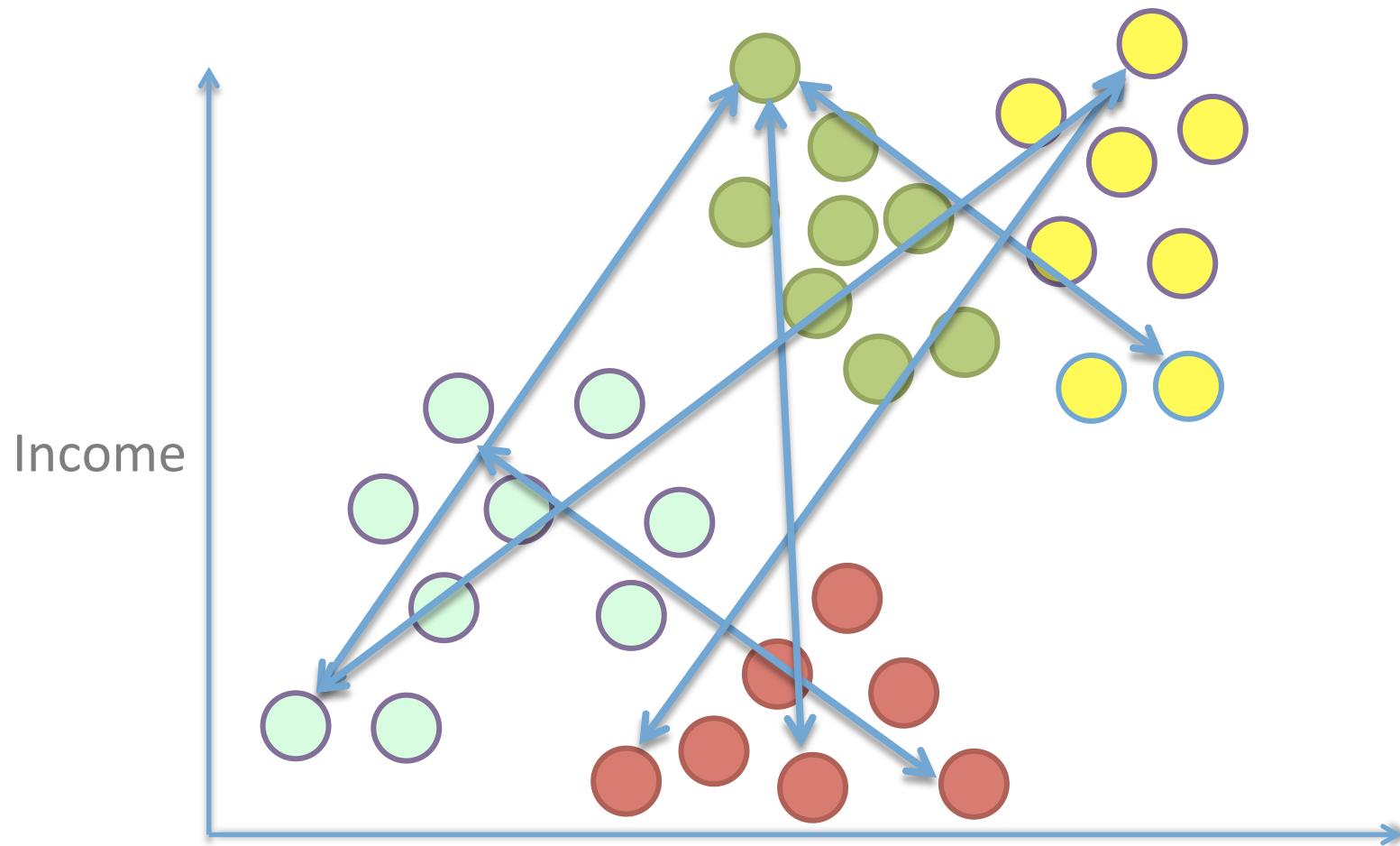
linkage = “complete”

Distance between two clusters is the maximum pairwise distance



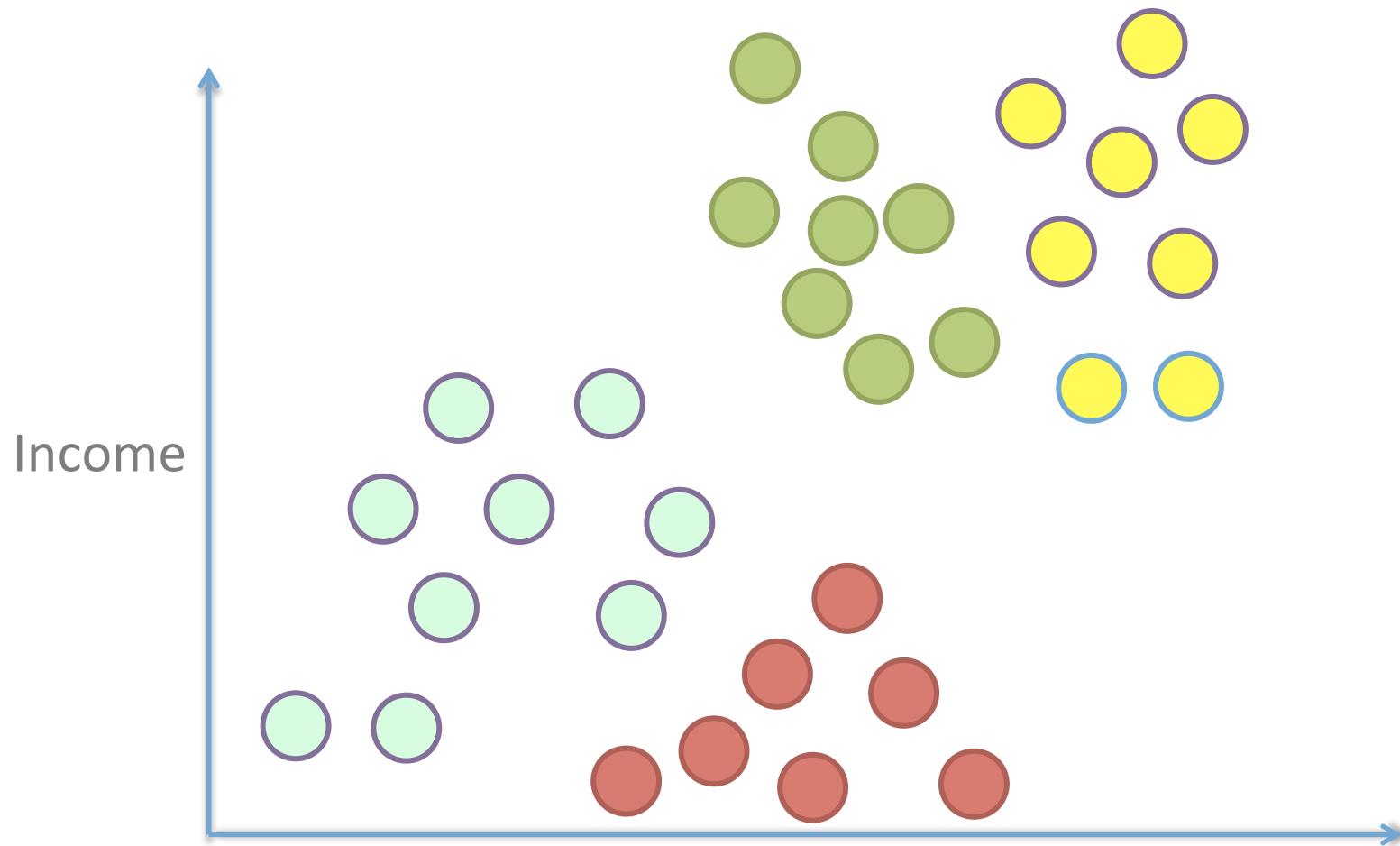
linkage = “complete”

Fast. No chains. Sensitive to outliers.



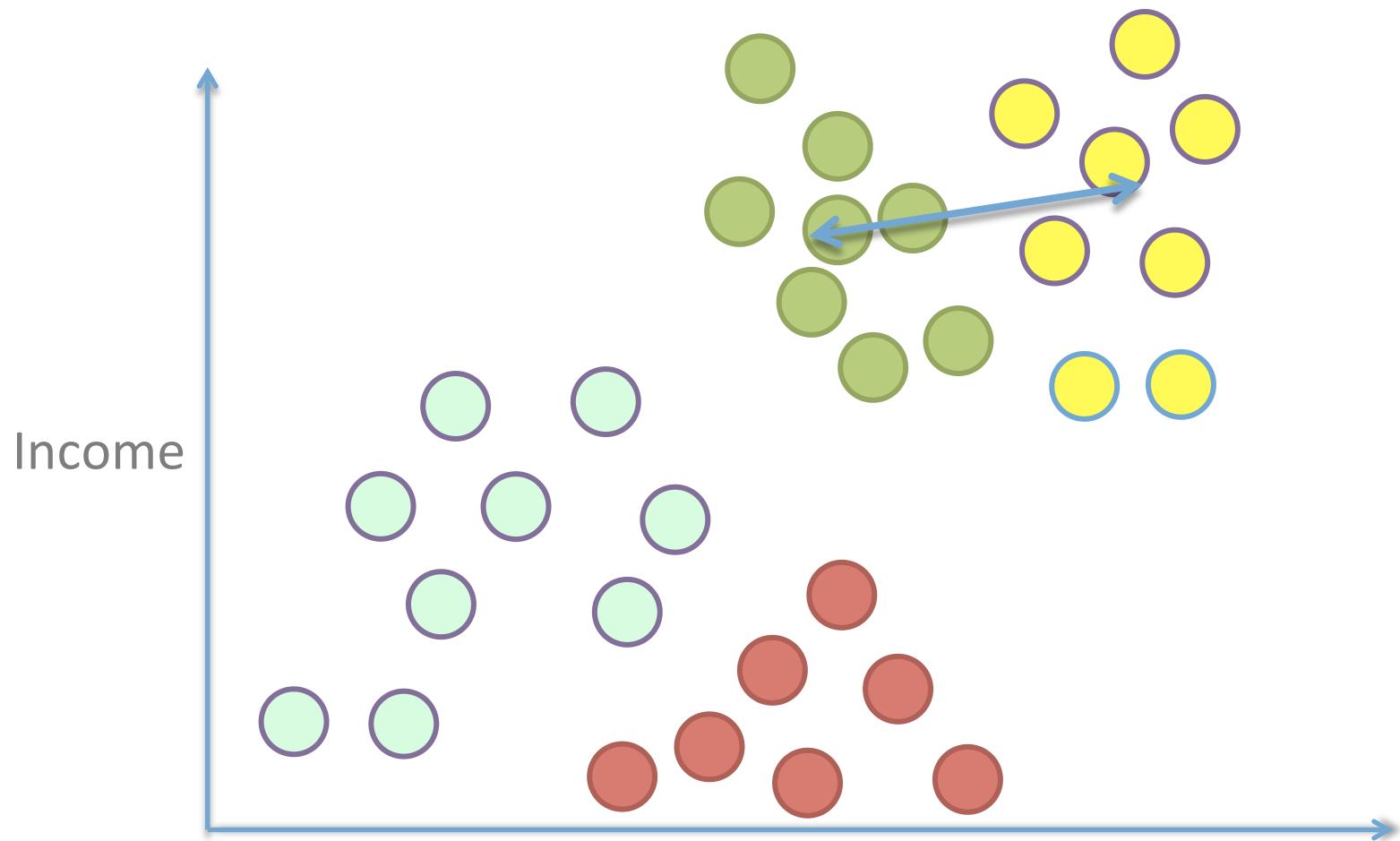
linkage = “average”

Distance between two cluster is average of all pairwise distance



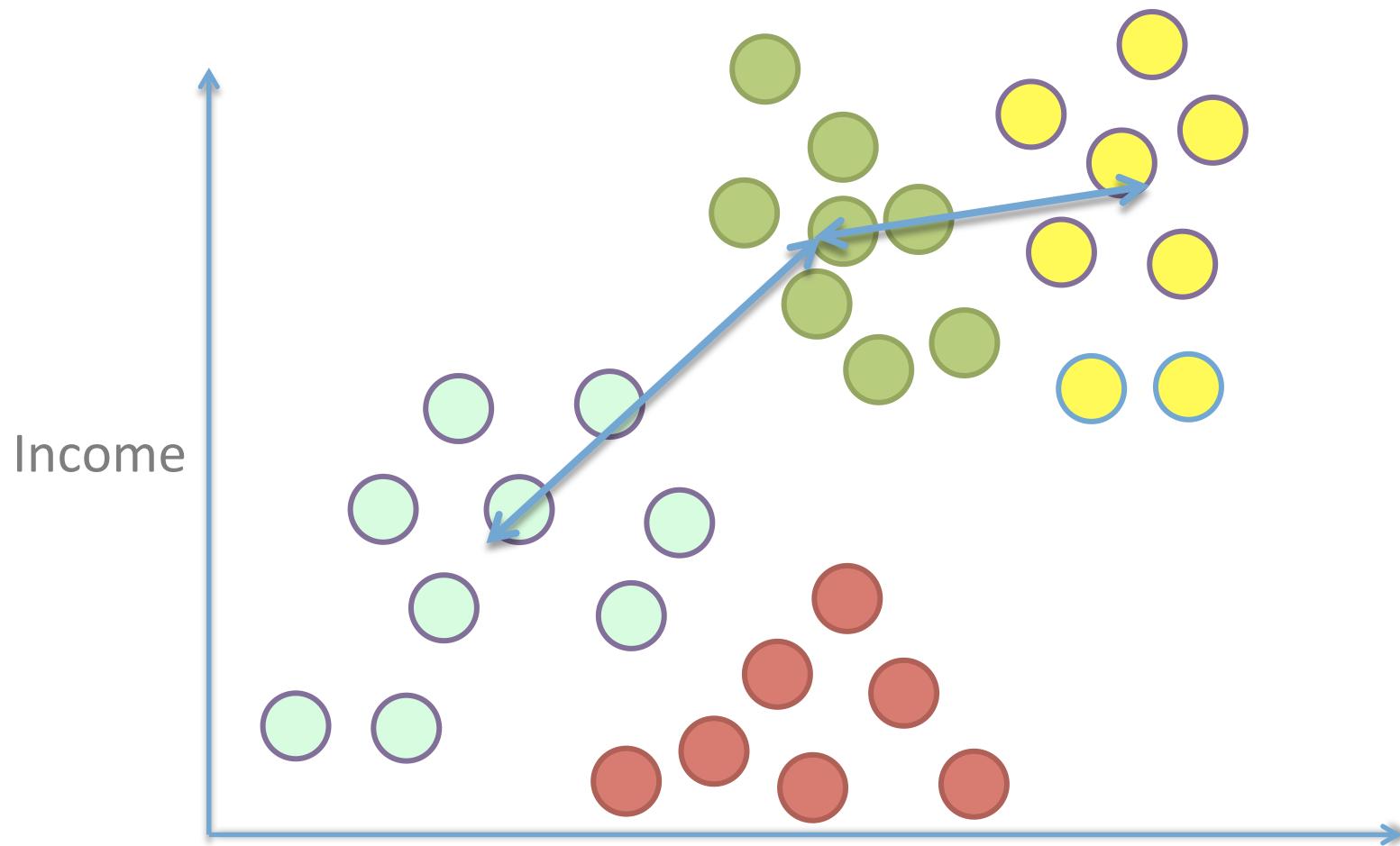
linkage = “average”

Distance between two cluster is average of all pairwise distance



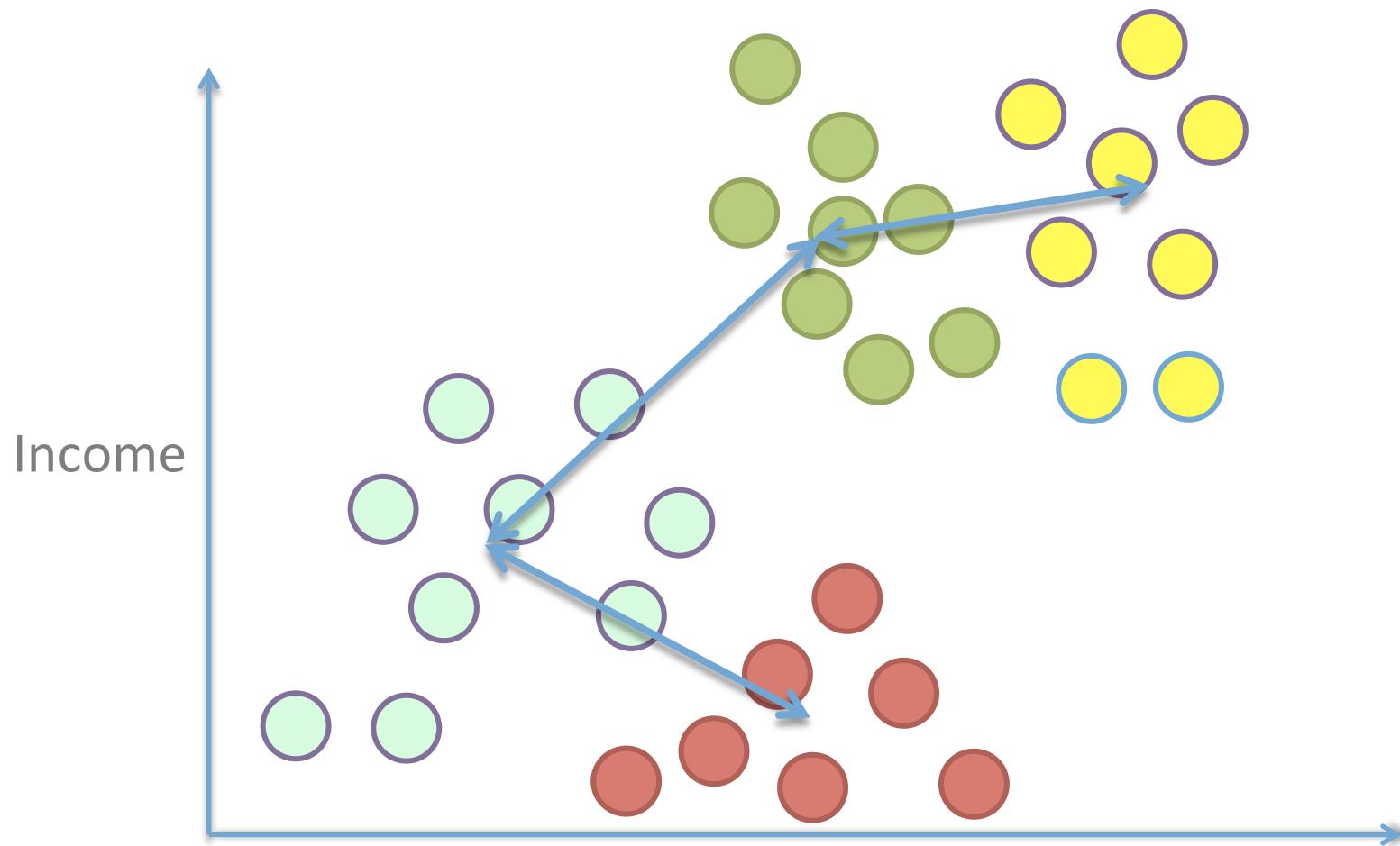
linkage = “average”

Distance between two cluster is average of all pairwise distance



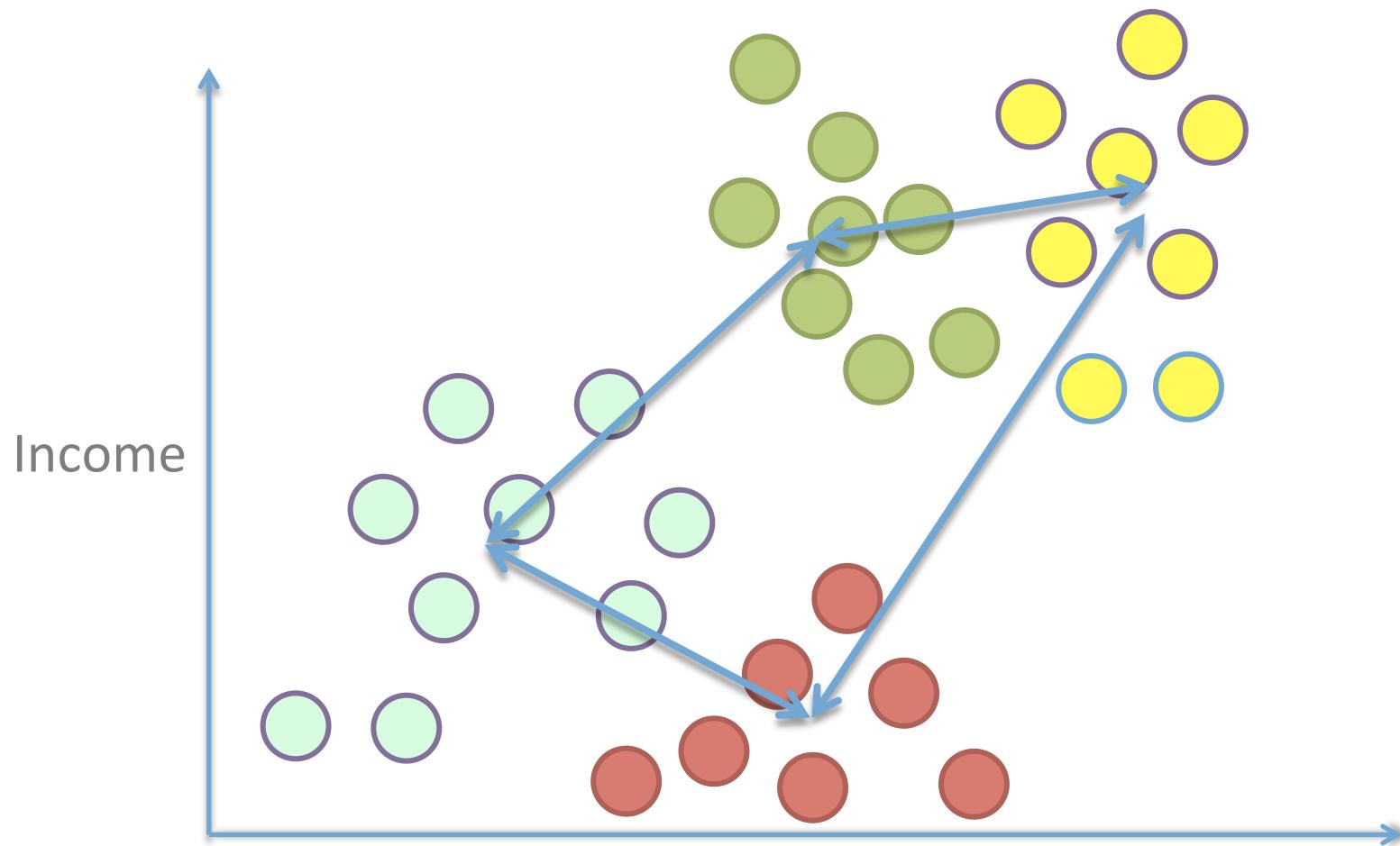
linkage = “average”

Distance between two cluster is average of all pairwise distance



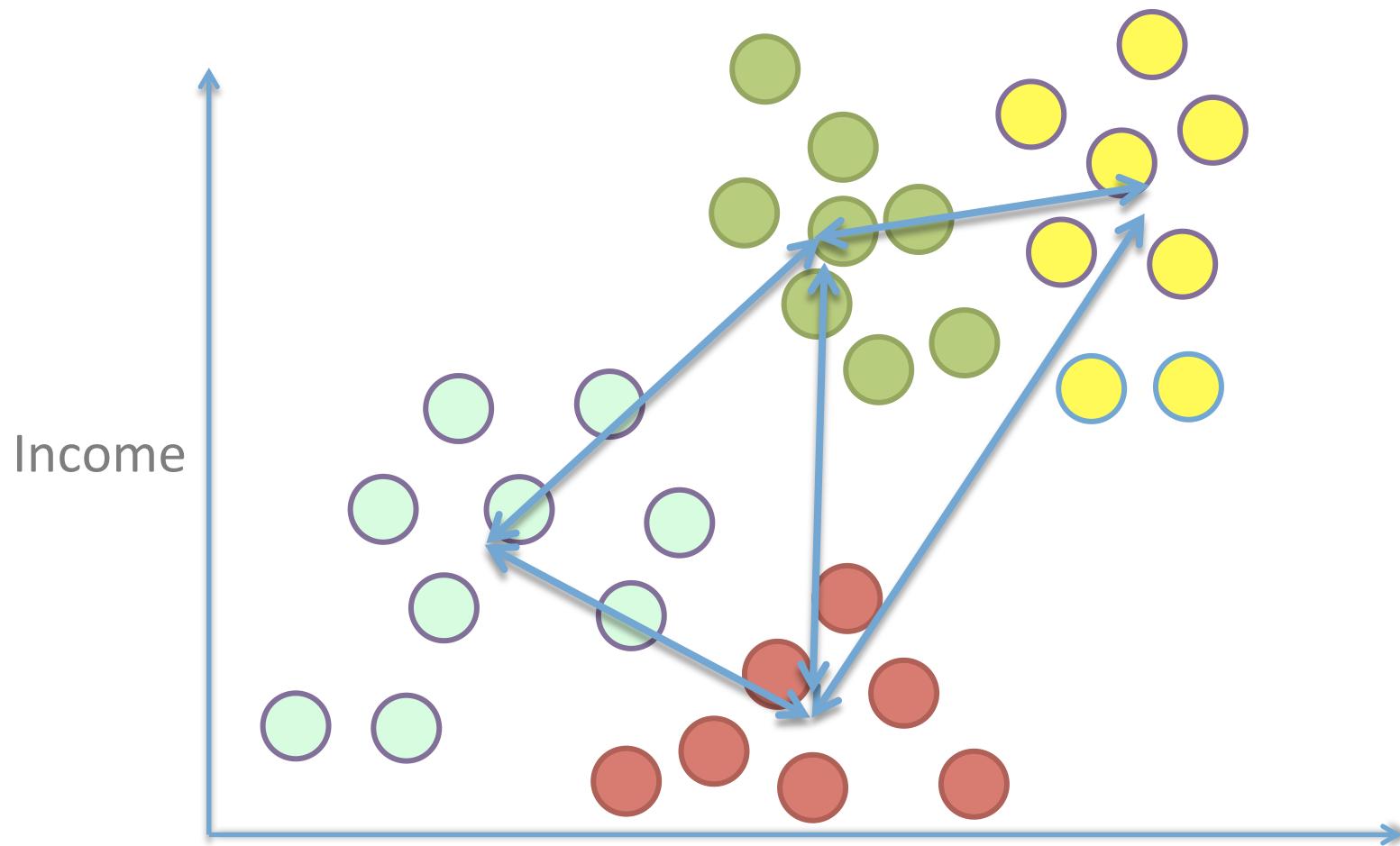
linkage = “average”

Distance between two cluster is average of all pairwise distance



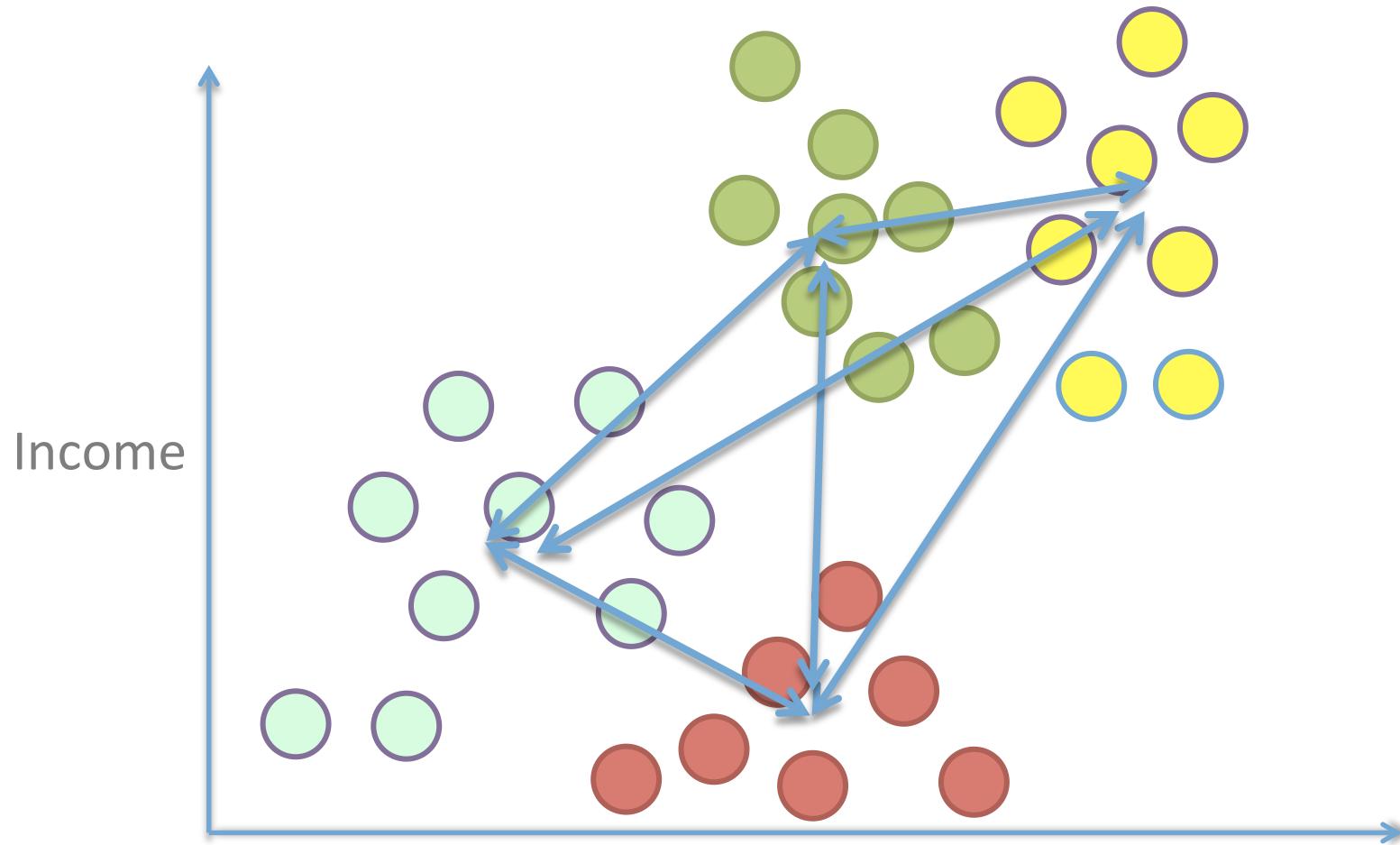
linkage = “average”

Distance between two cluster is average of all pairwise distance



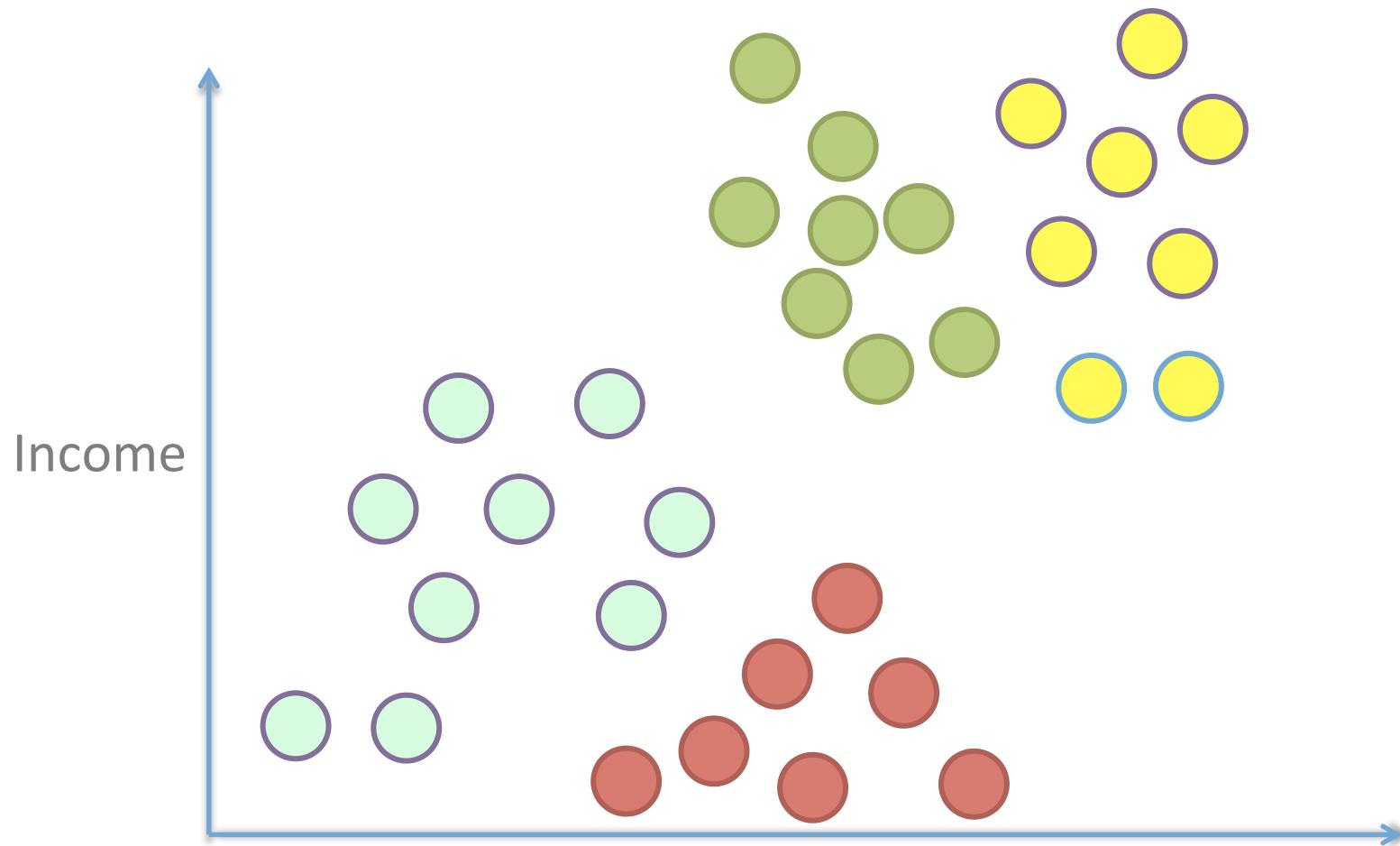
linkage = “average”

Slower. No chains. Not sensitive to outliers.



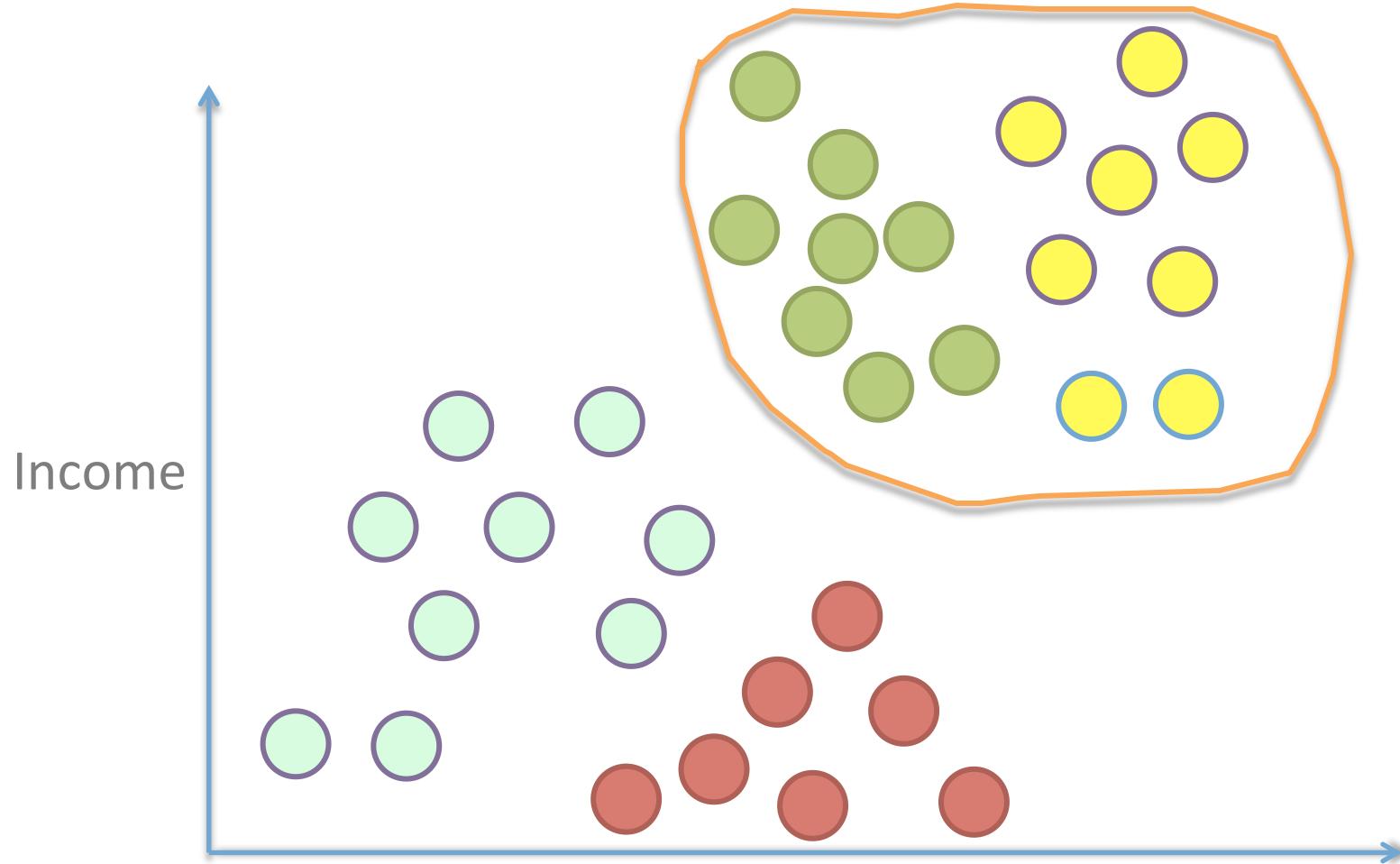
linkage = “ward”

Don't merge based on distance. Merge the two whose combination will give the best score (lowest inertia).



linkage = “ward”

This will choose the merge with the minimum within-cluster variance (high density clusters).



```
from sklearn.cluster import AgglomerativeClustering  
Agglomerative Clustering(linkage="Ward").fit_predict(X)
```

or

```
from sklearn.cluster import Ward  
Ward().fit_predict(X)
```

(Same thing)

```
from sklearn.cluster import AgglomerativeClustering
```

```
AgglomerativeClustering(...,  
...,  
linkage = "ward",  
...,  
affinity = "euclidian",  
....)
```

```
from sklearn.cluster import AgglomerativeClustering
```

```
AgglomerativeClustering(...,
```

```
....,
```

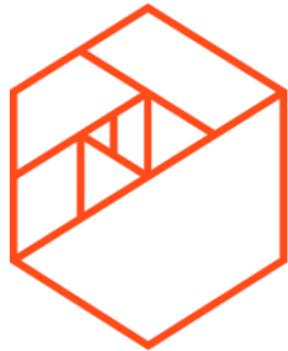
```
linkage = "ward",
```

```
....,
```

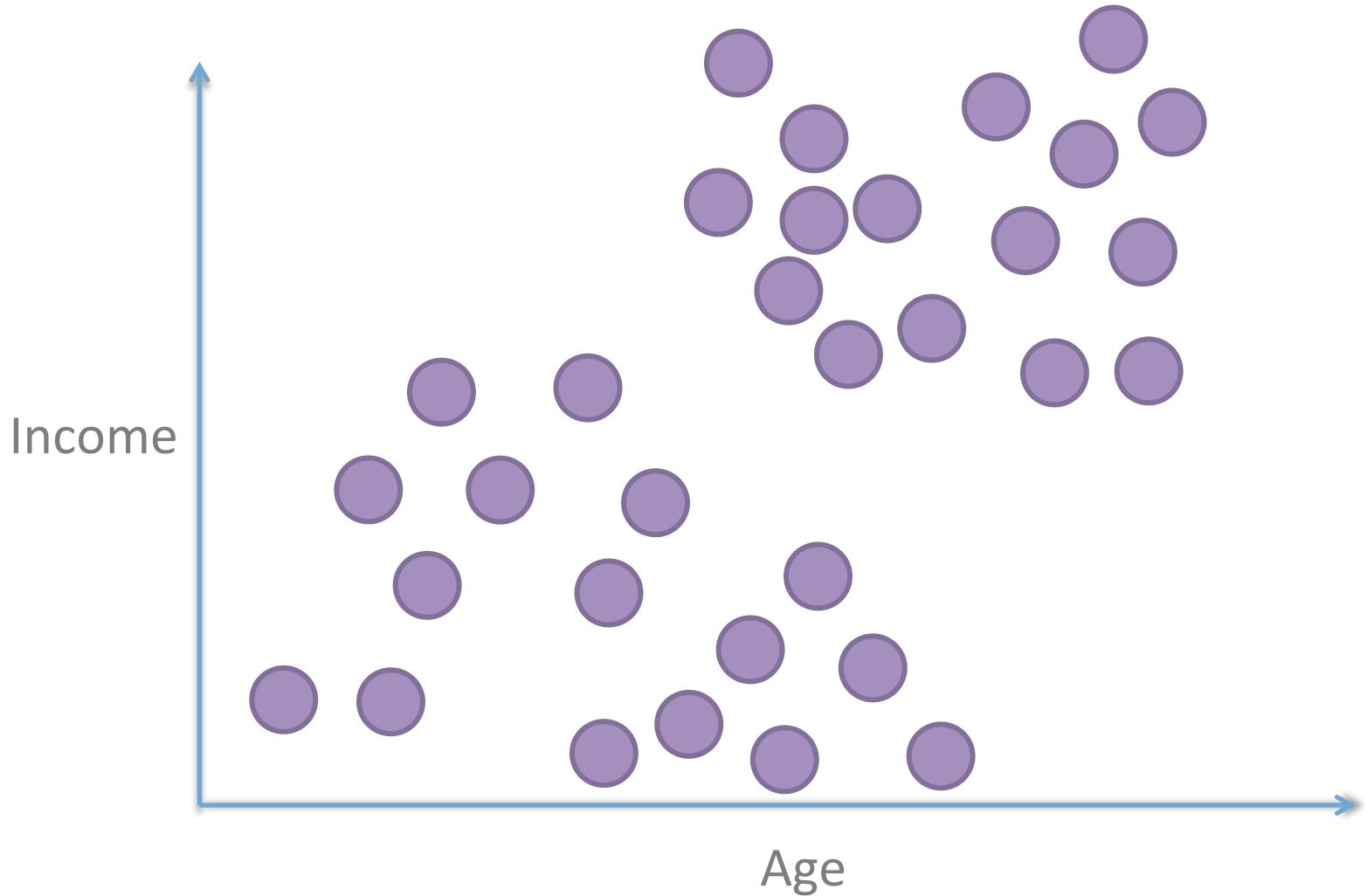
```
affinity = "euclidian",
```

```
....)
```

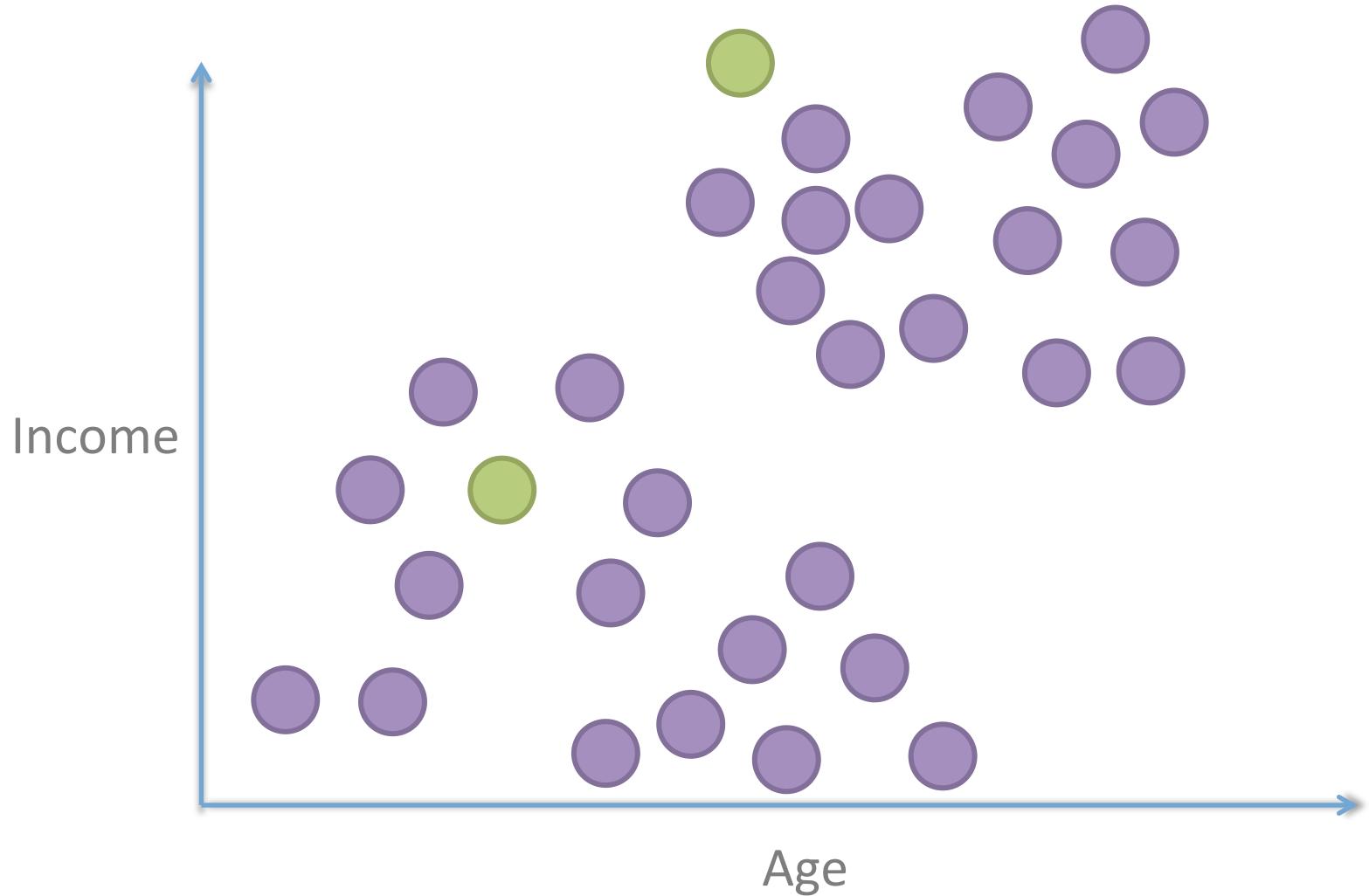
Distance Metrics



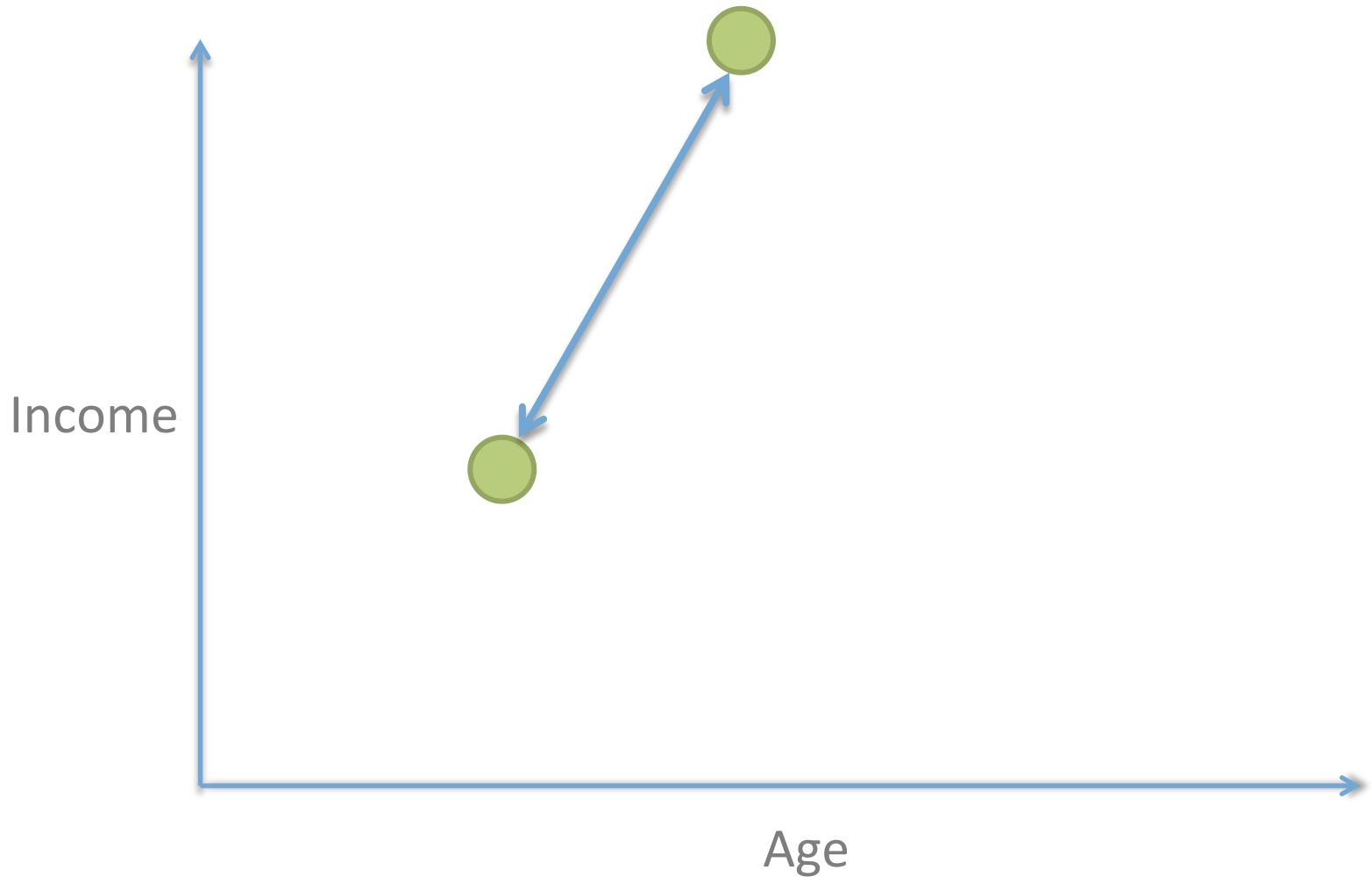
Euclidean Distance



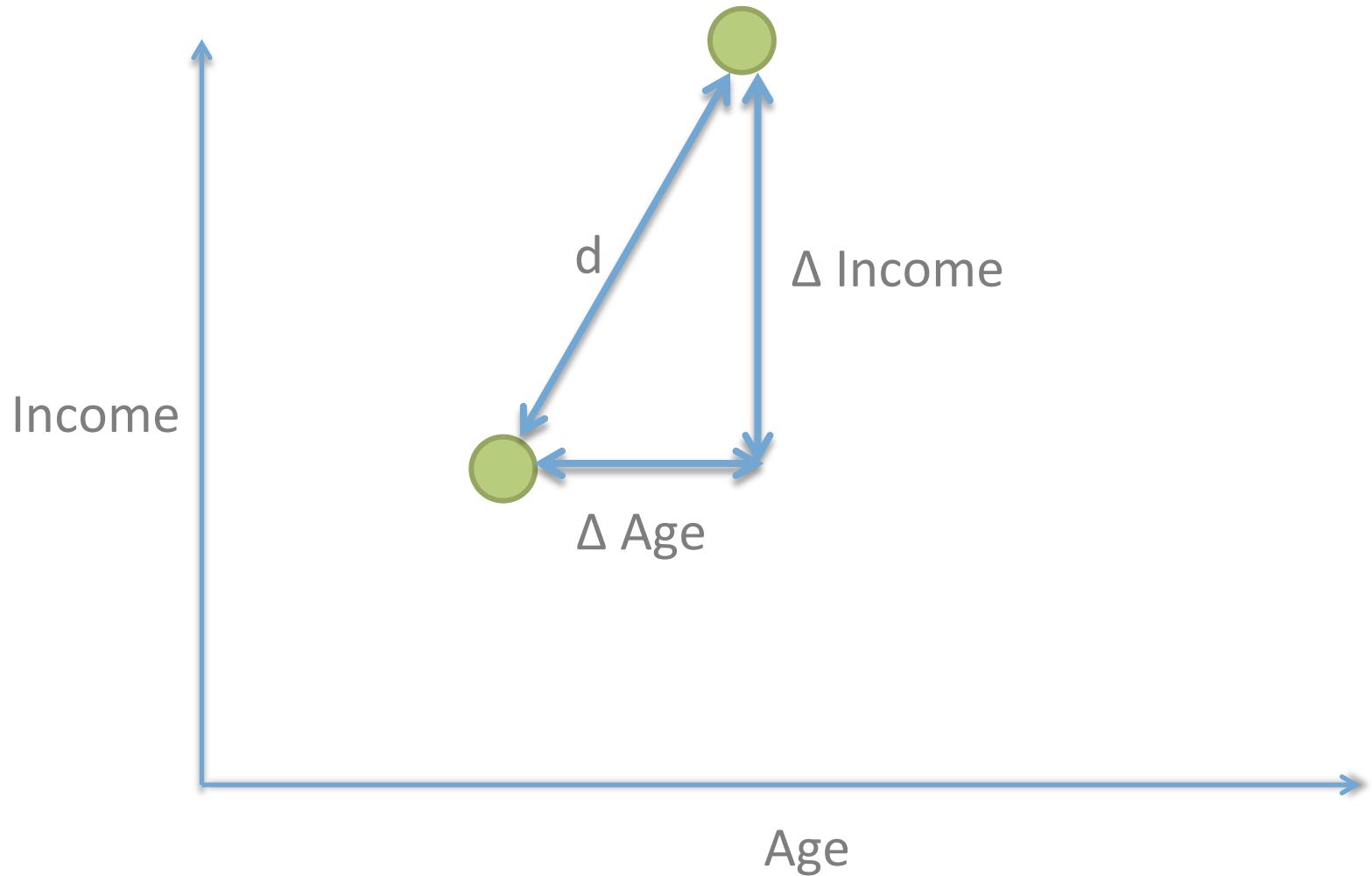
Euclidean Distance



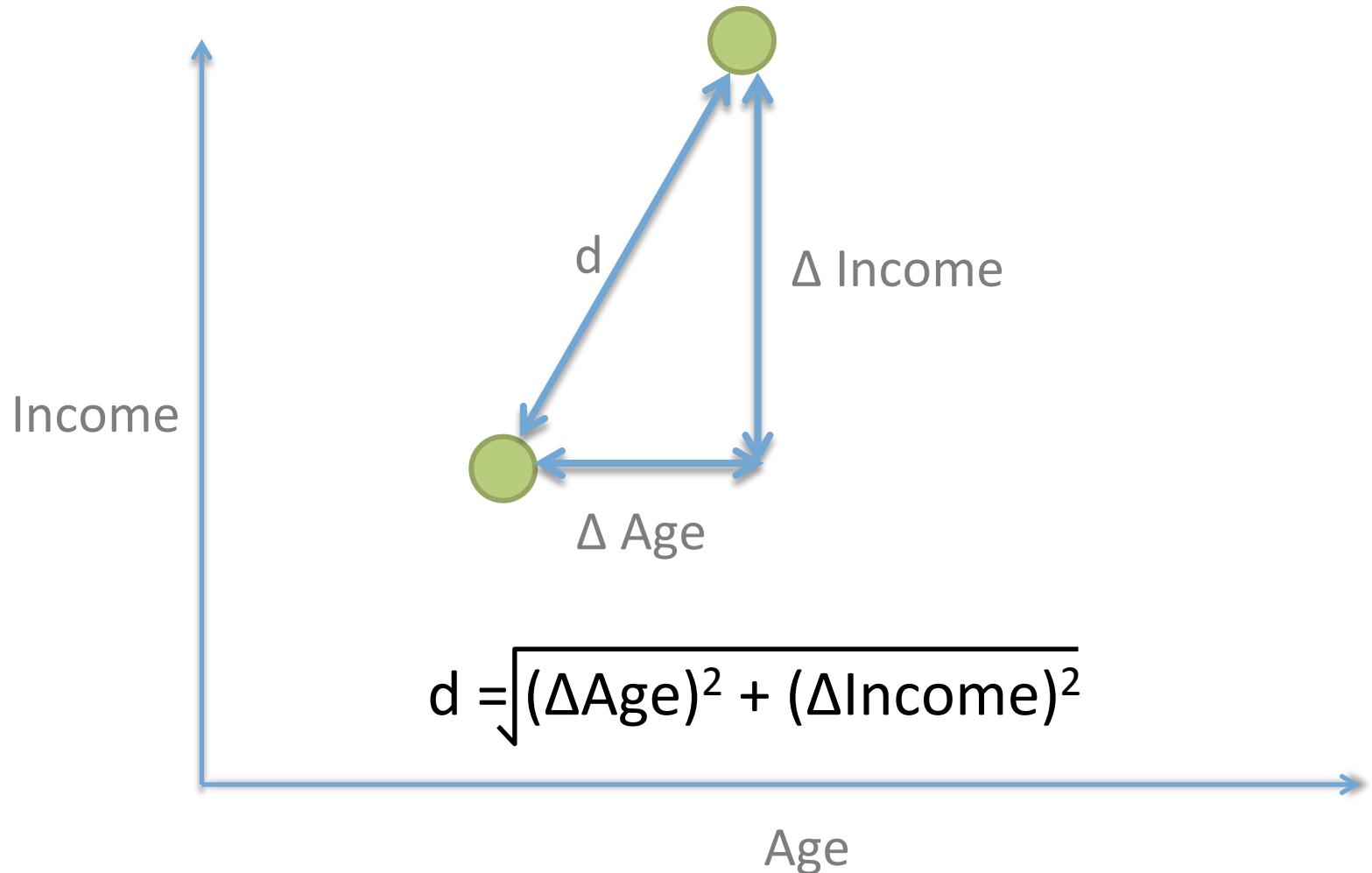
Euclidean Distance



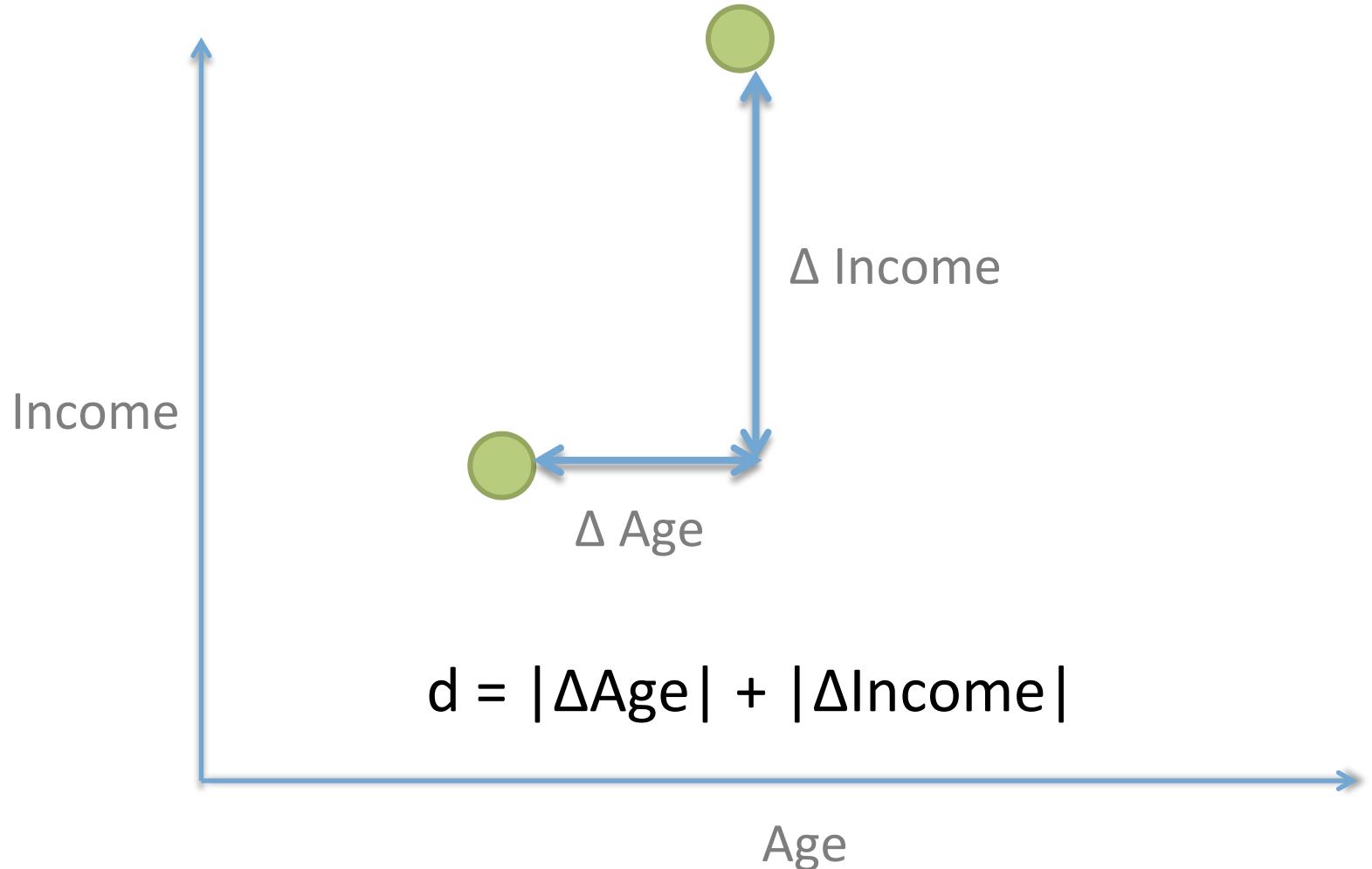
Euclidean Distance



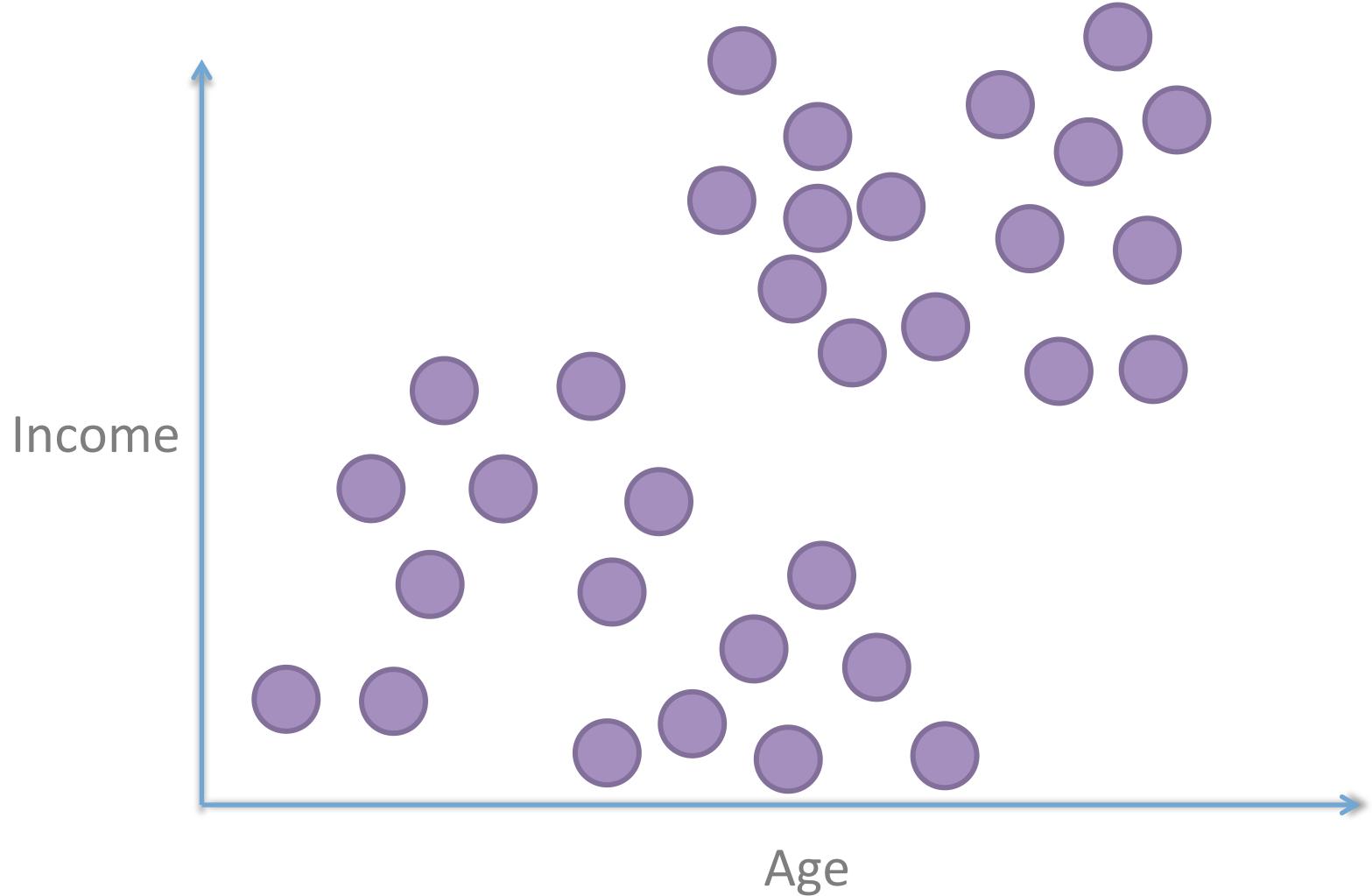
Euclidean Distance (L2 dist)



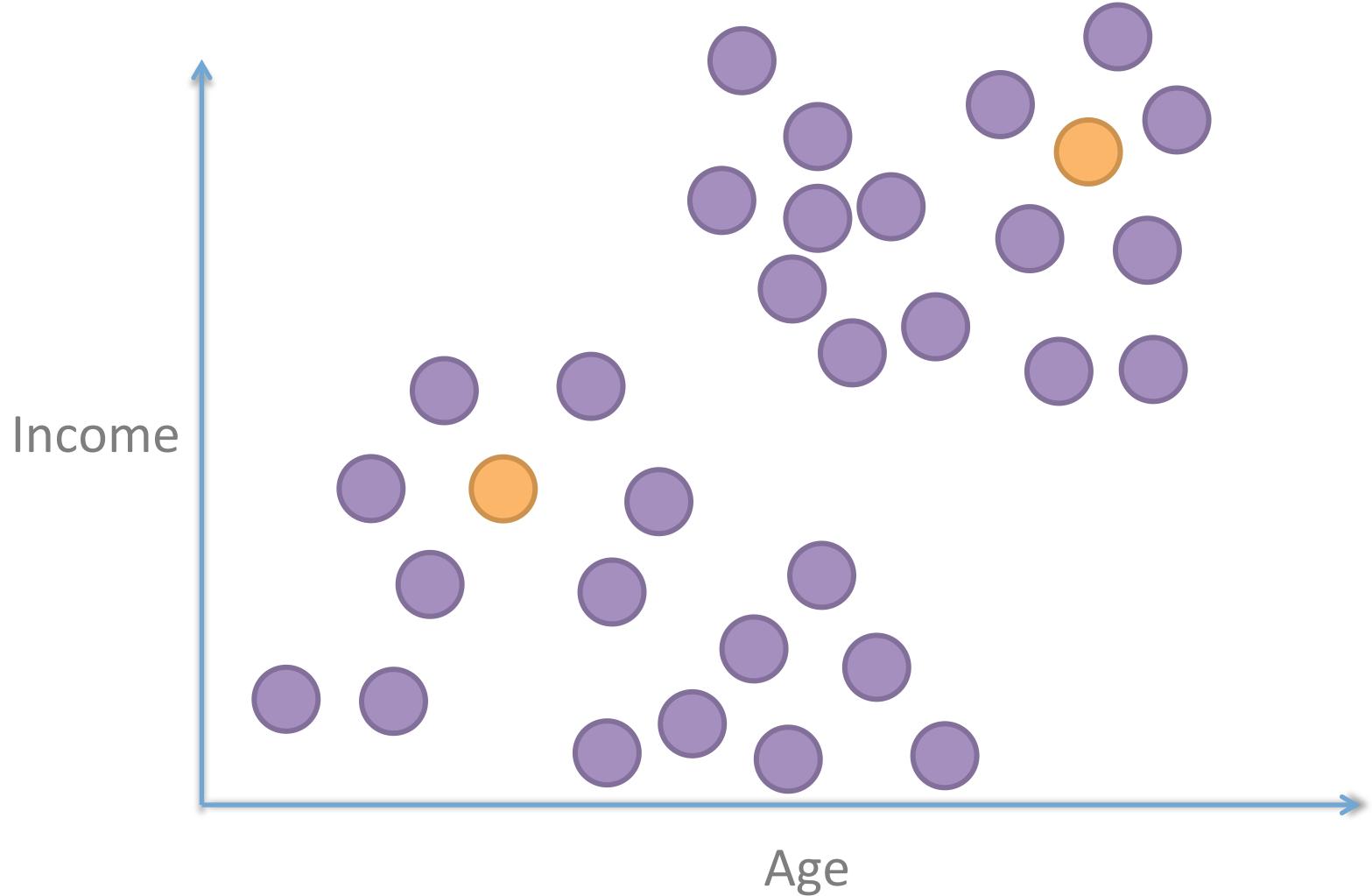
Manhattan Distance (Cityblock dist, L1 dist)



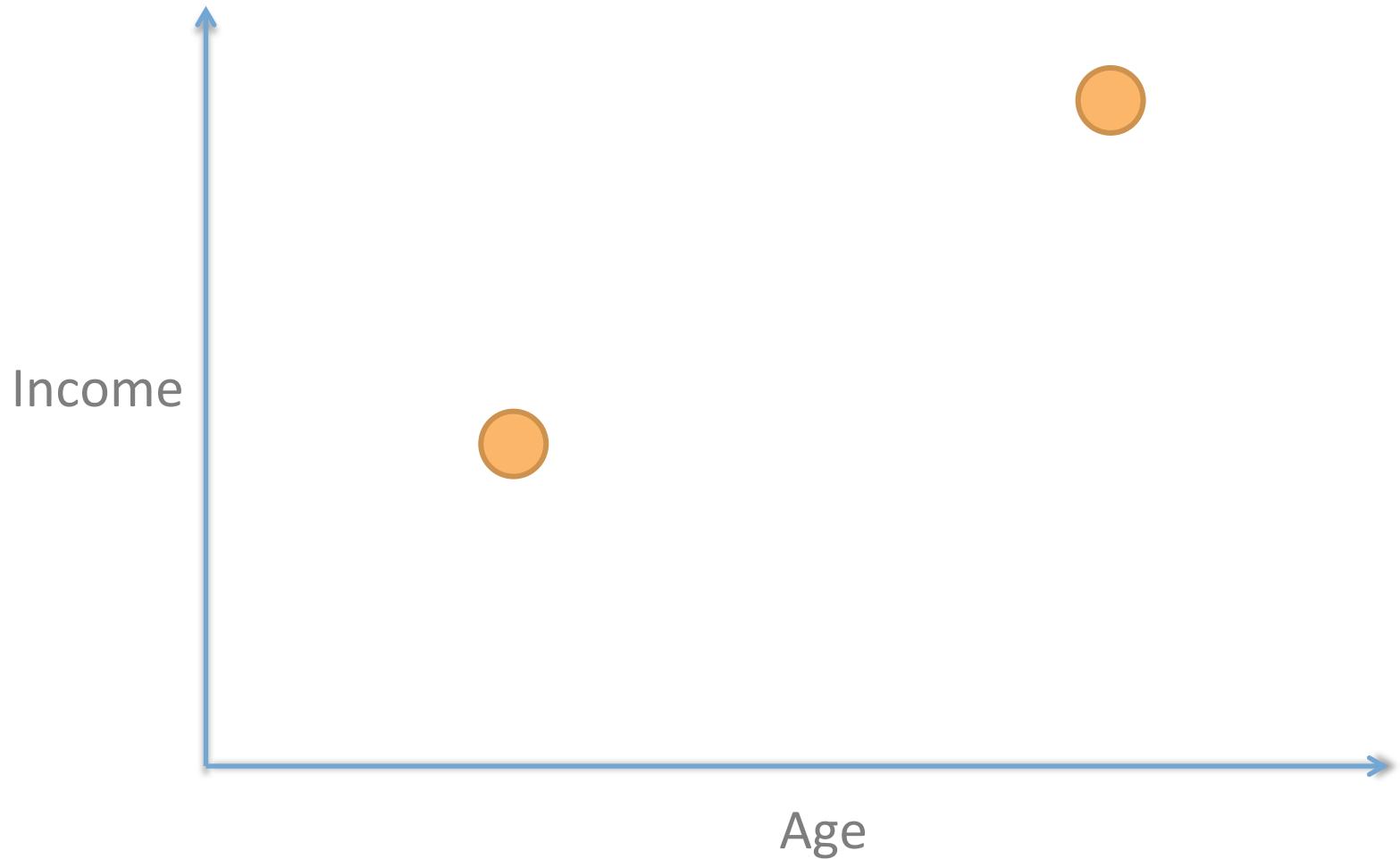
Cosine Distance



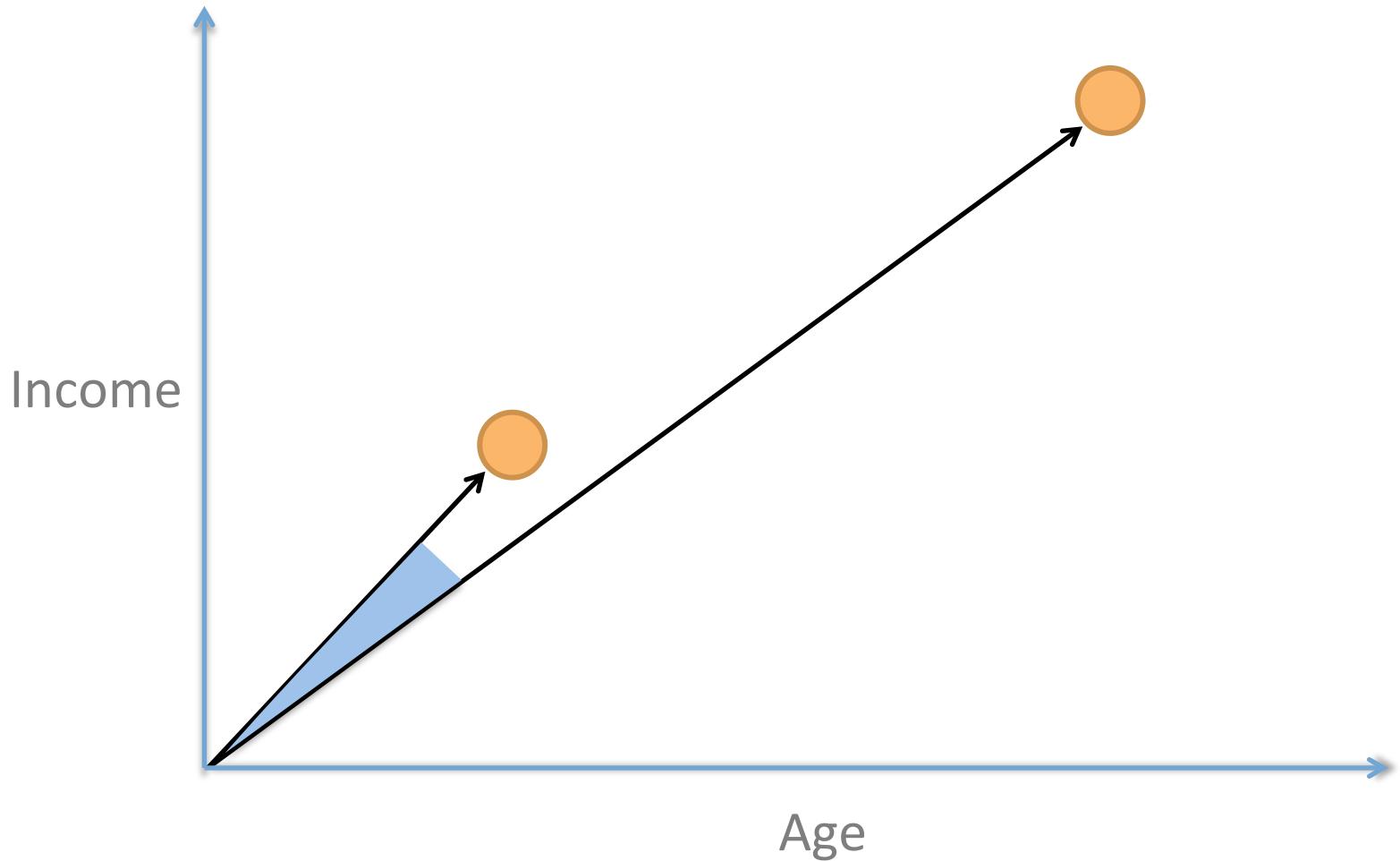
Cosine Distance



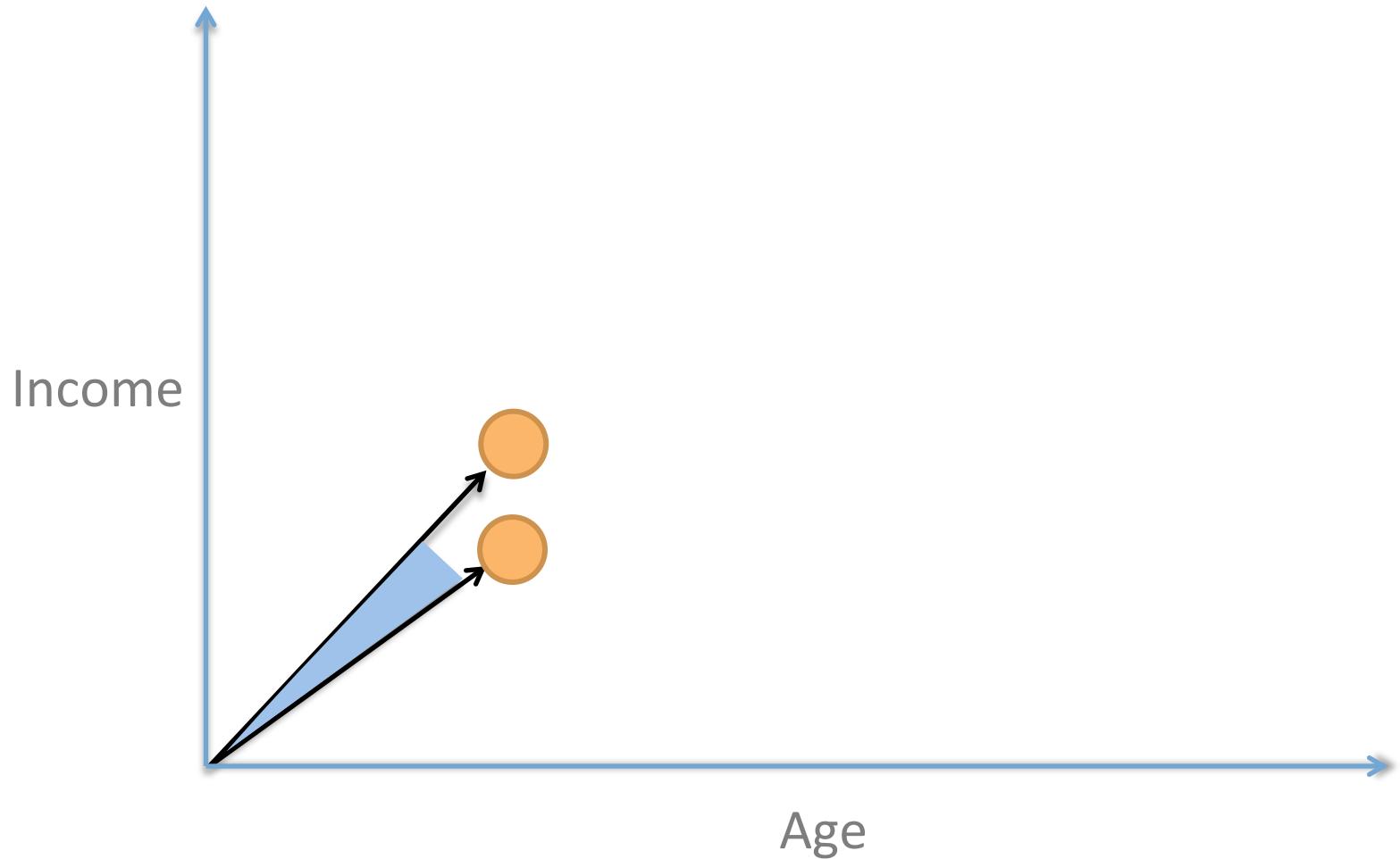
Cosine Distance



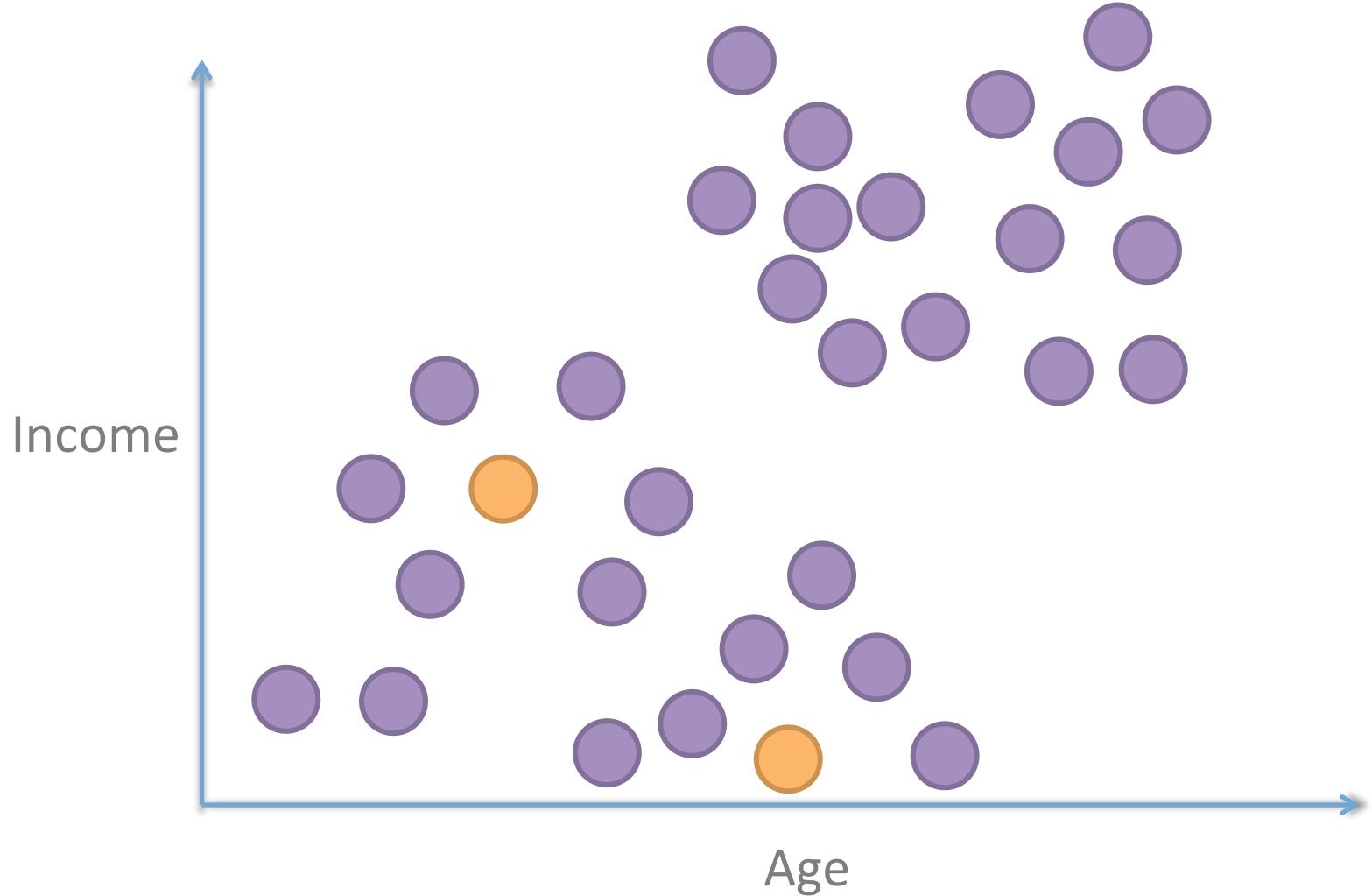
Cosine Distance



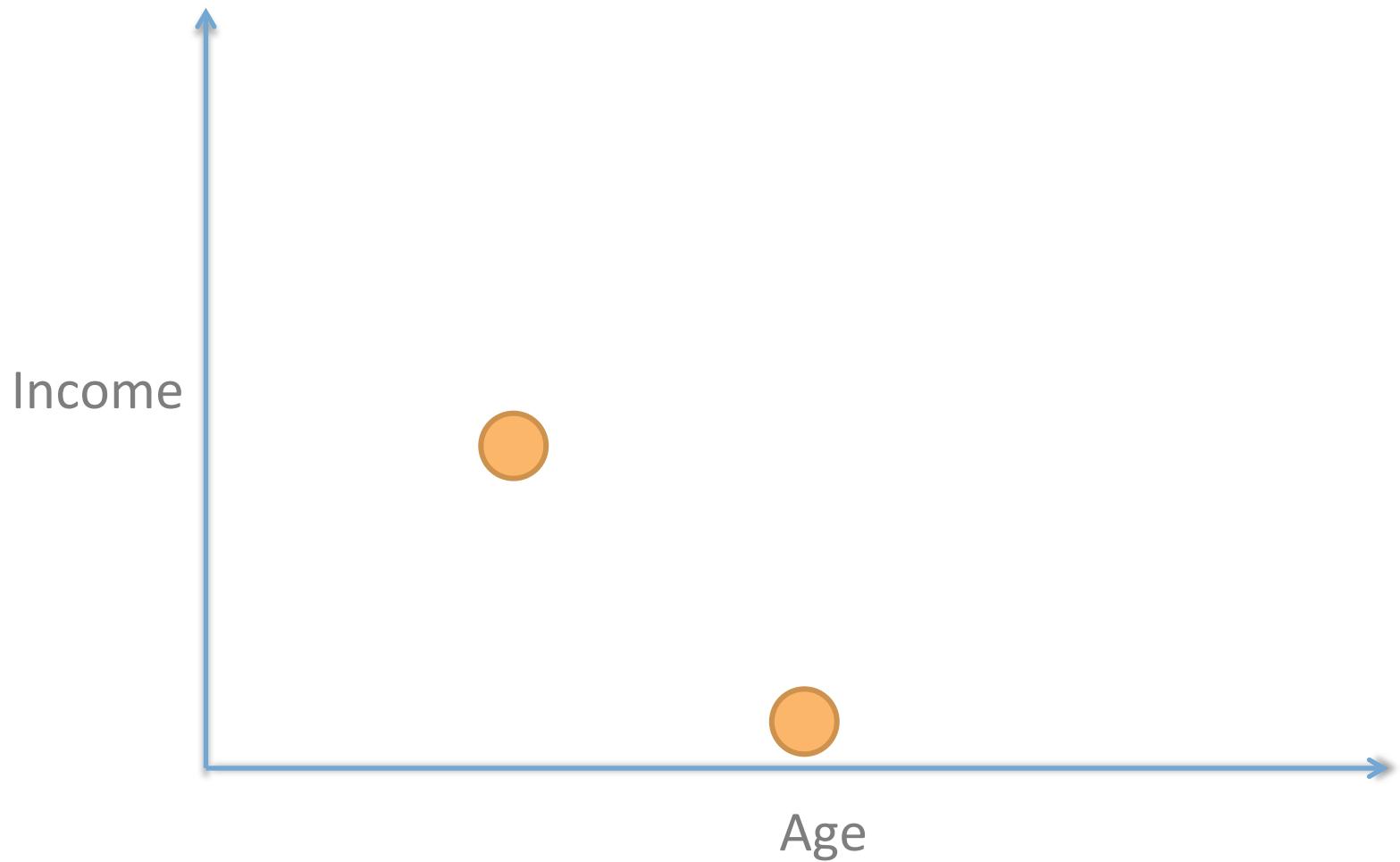
Cosine Distance



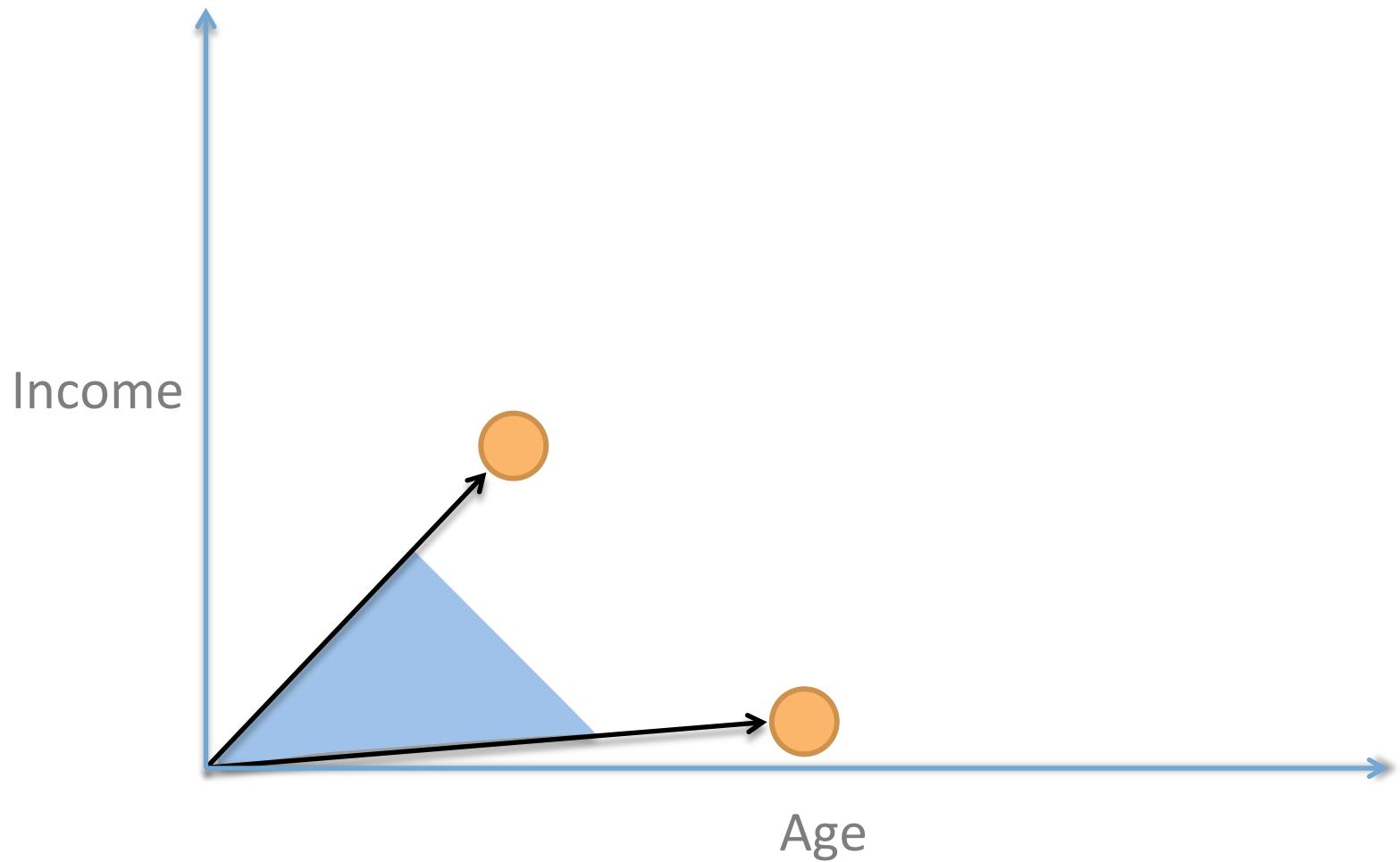
Cosine Distance



Cosine Distance

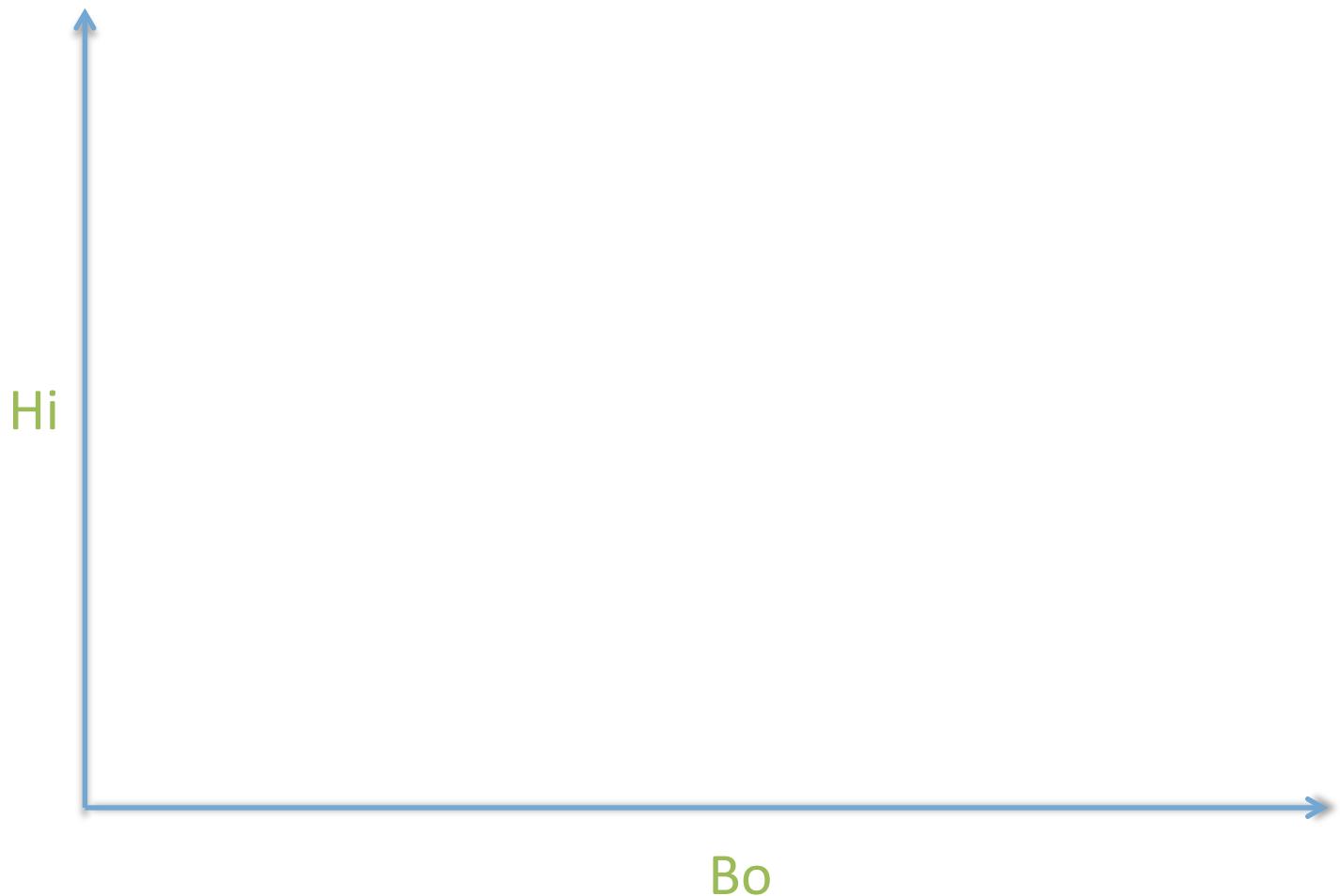


Cosine Distance



Cosine Distance is better for text data

Euclidean Distance is sensitive to doc. length



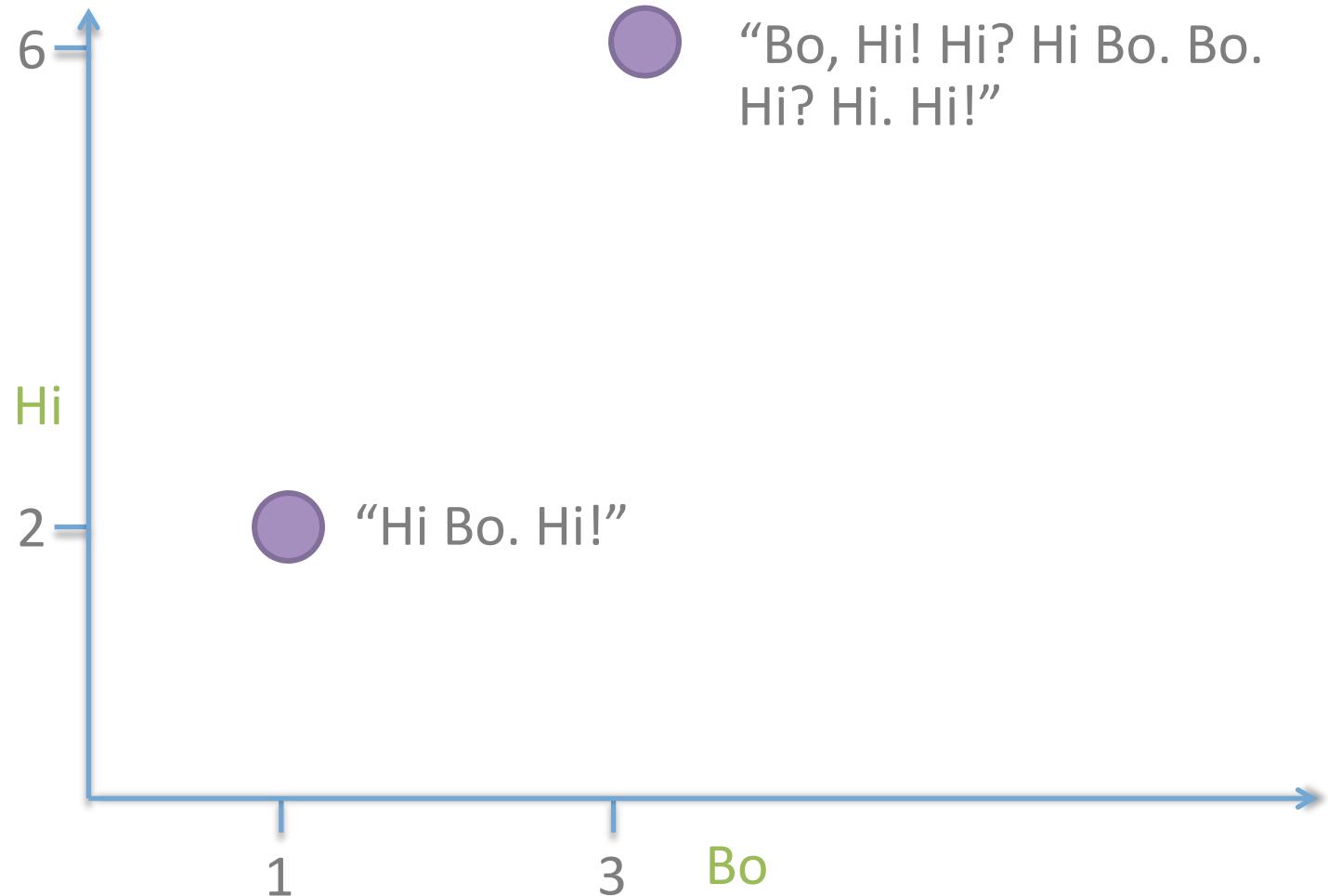
Cosine Distance is better for text data

Euclidean Distance is sensitive to doc. length



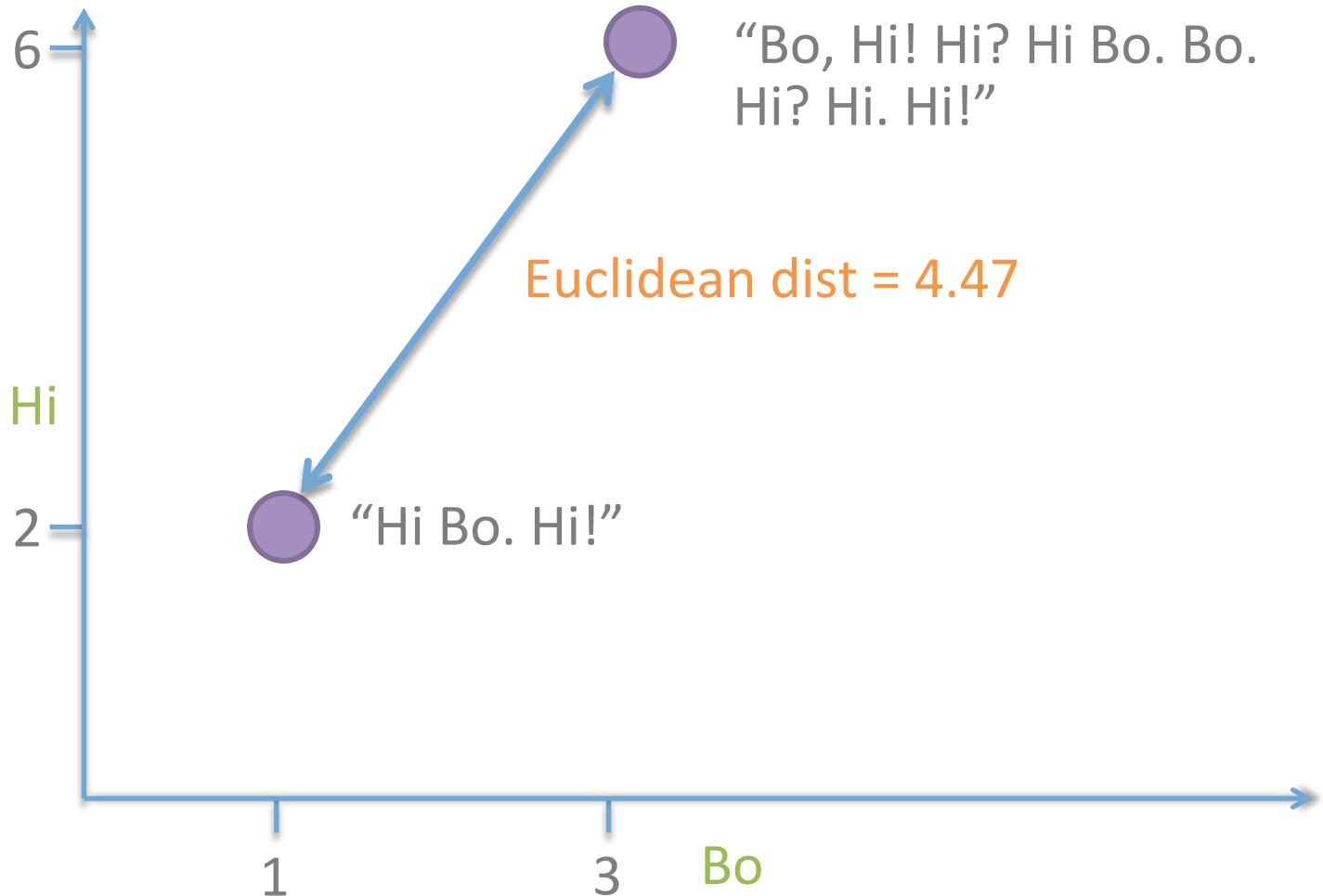
Cosine Distance is better for text data

Euclidean Distance is sensitive to doc. length



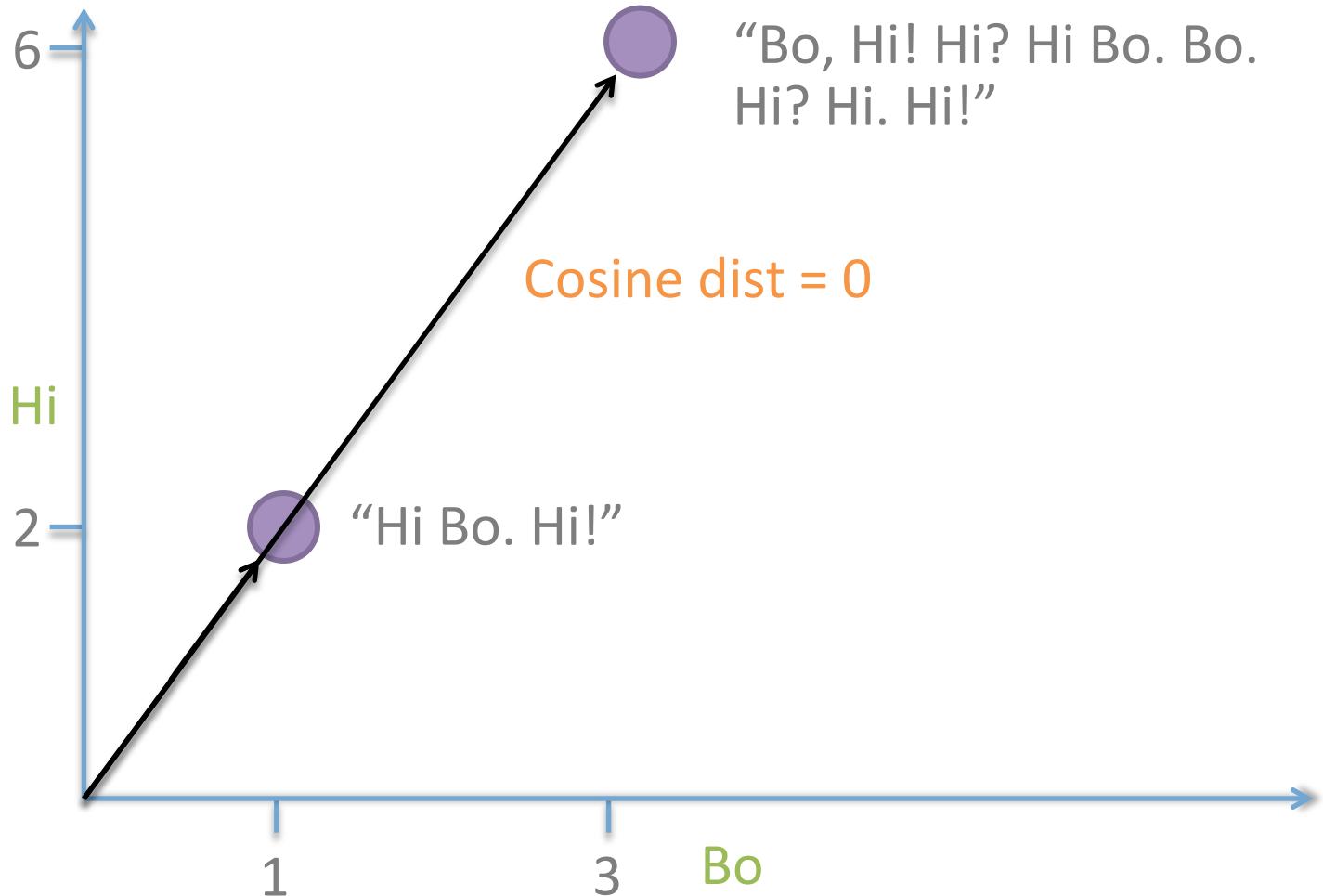
Cosine Distance is better for text data

Euclidean Distance is sensitive to doc. length

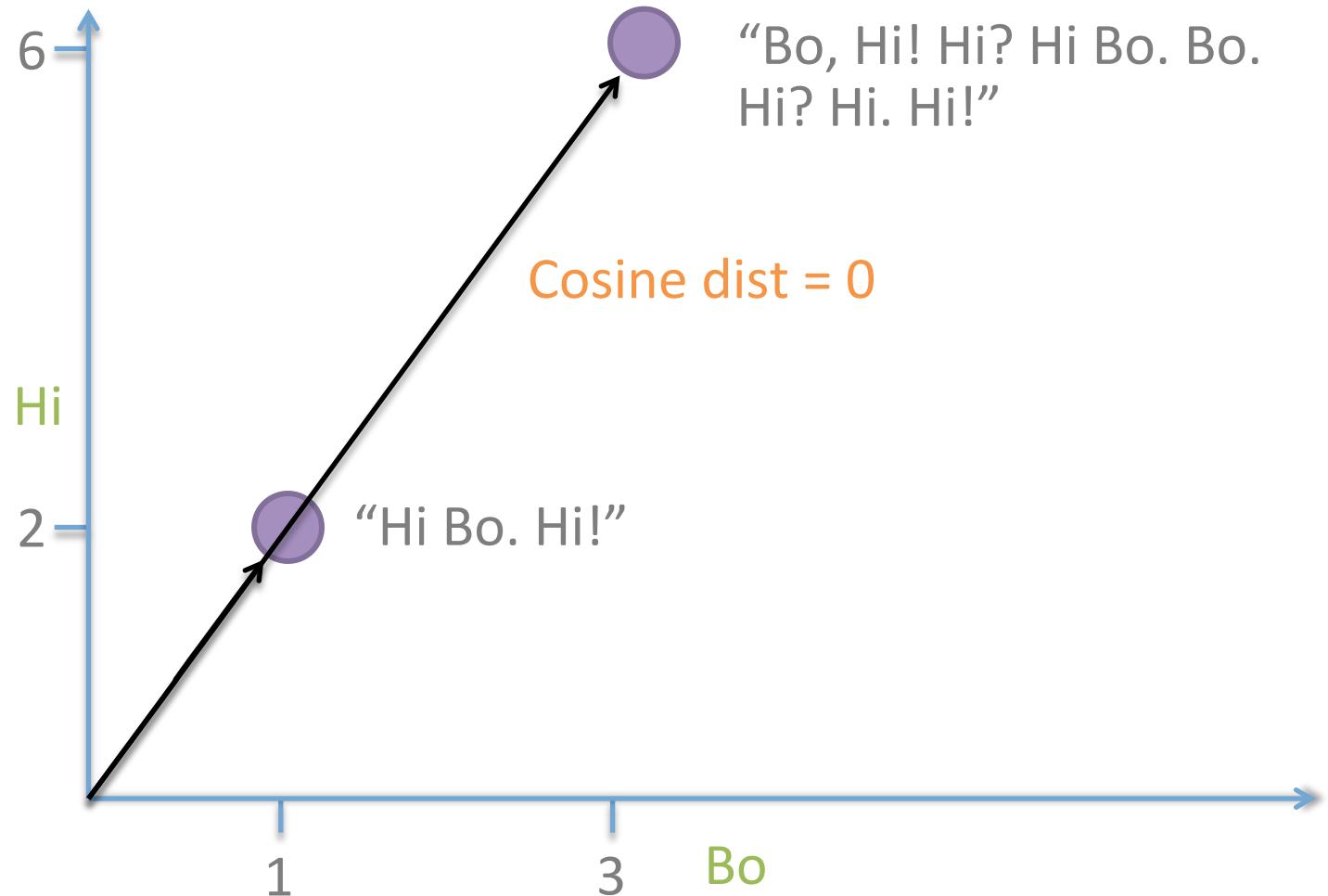


Cosine Distance is better for text data

Euclidean Distance is sensitive to doc. length



Cosine Distance captures the “meaning” distance better in a bag of words system



Jaccard Distance

Applies to sets (like bag of words)

Jaccard Distance

Applies to sets (like bag of words)

Sentence A: “I like chocolate ice cream.”

set A = {I, like, chocolate, ice, cream}

Sentence B: “Do I want chocolate cream or vanilla cream?”

set B = {Do, I, want, chocolate, cream, or, vanilla}

Jaccard dist

Jaccard Distance

Applies to sets (like bag of words)

Sentence A: “I like chocolate ice cream.”

set A = {I, like, chocolate, ice, cream}

Sentence B: “Do I want chocolate cream or vanilla cream?”

set B = {Do, I, want, chocolate, cream, or, vanilla}

Jaccard dist $1 - \frac{A \cap B}{A \cup B} = 1 - \frac{\text{len(shared)}}{\text{len(all)}}$

Jaccard Distance

Applies to sets (like bag of words)

Sentence A: “I like chocolate ice cream.”

set A = {I, like, chocolate, ice, cream}

Sentence B: “Do I want chocolate cream or vanilla cream?”

set B = {Do, I, want, chocolate, cream, or, vanilla}

Jaccard dist $1 - \frac{A \cap B}{A \cup B} = 1 - \frac{\text{len(shared)}}{\text{len(all)}}$

Jaccard Distance

Applies to sets (like bag of words)

Sentence A: “I like chocolate ice cream.”

set A = {I, like, chocolate, ice, cream}

Sentence B: “Do I want chocolate cream or vanilla cream?”

set B = {Do, I, want, chocolate, cream, or, vanilla}

Jaccard dist

$$1 - \frac{\text{len}(\{I, chocolate, cream\})}{\text{len}(\{I, like, chocolate, ice, cream, do, want, or, vanilla\})}$$

Jaccard Distance

Applies to sets (like bag of words)

Sentence A: “I like chocolate ice cream.”

set A = {I, like, chocolate, ice, cream}

Sentence B: “Do I want chocolate cream or vanilla cream?”

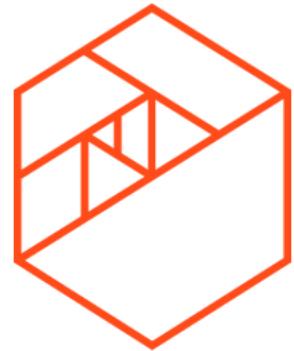
set B = {Do, I, want, chocolate, cream, or, vanilla}

Jaccard dist $=1 - 3/9 / =1 - 0.333 = 0.667$

Back to Other Clustering Algorithms

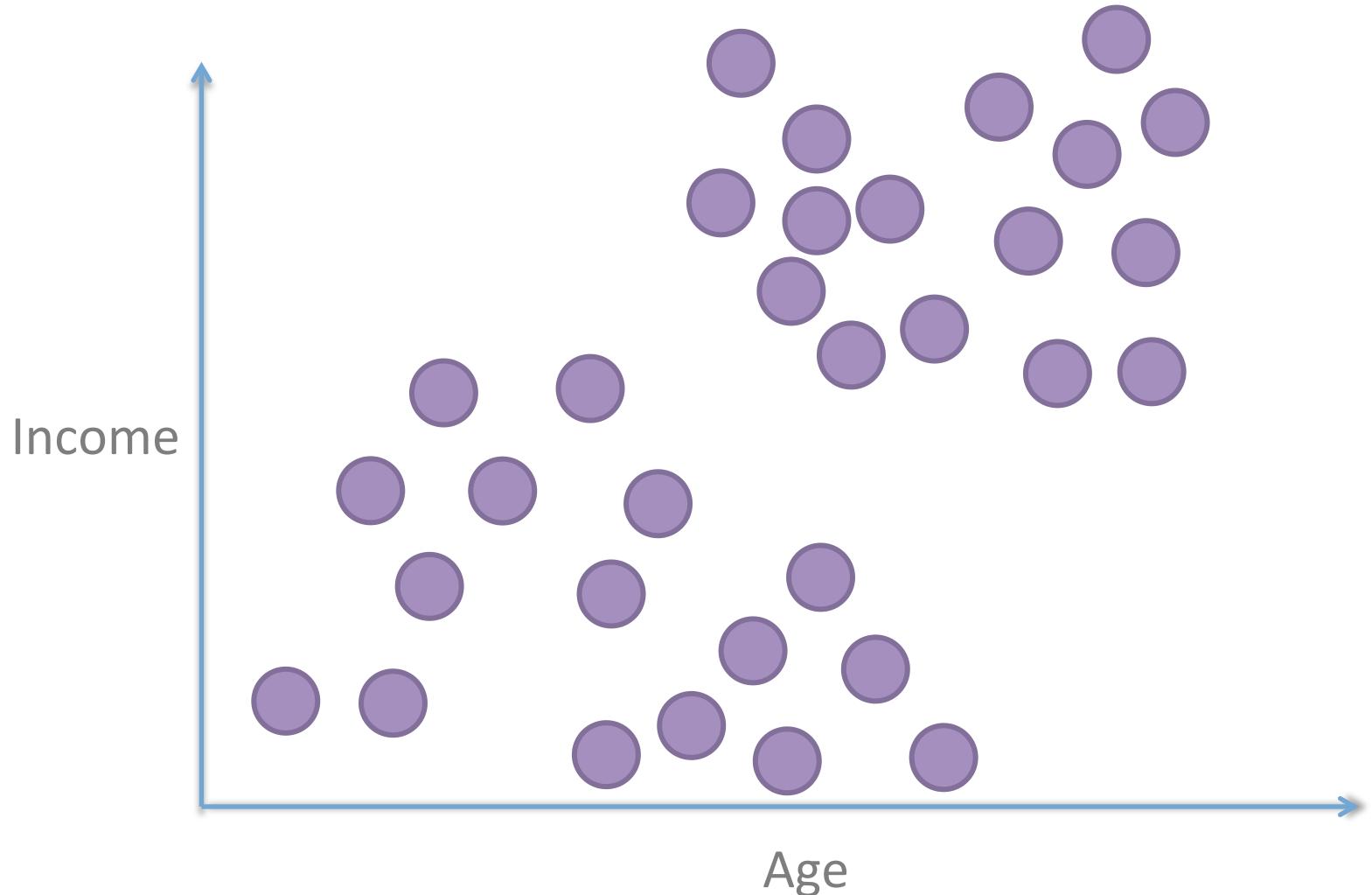


DBSCAN



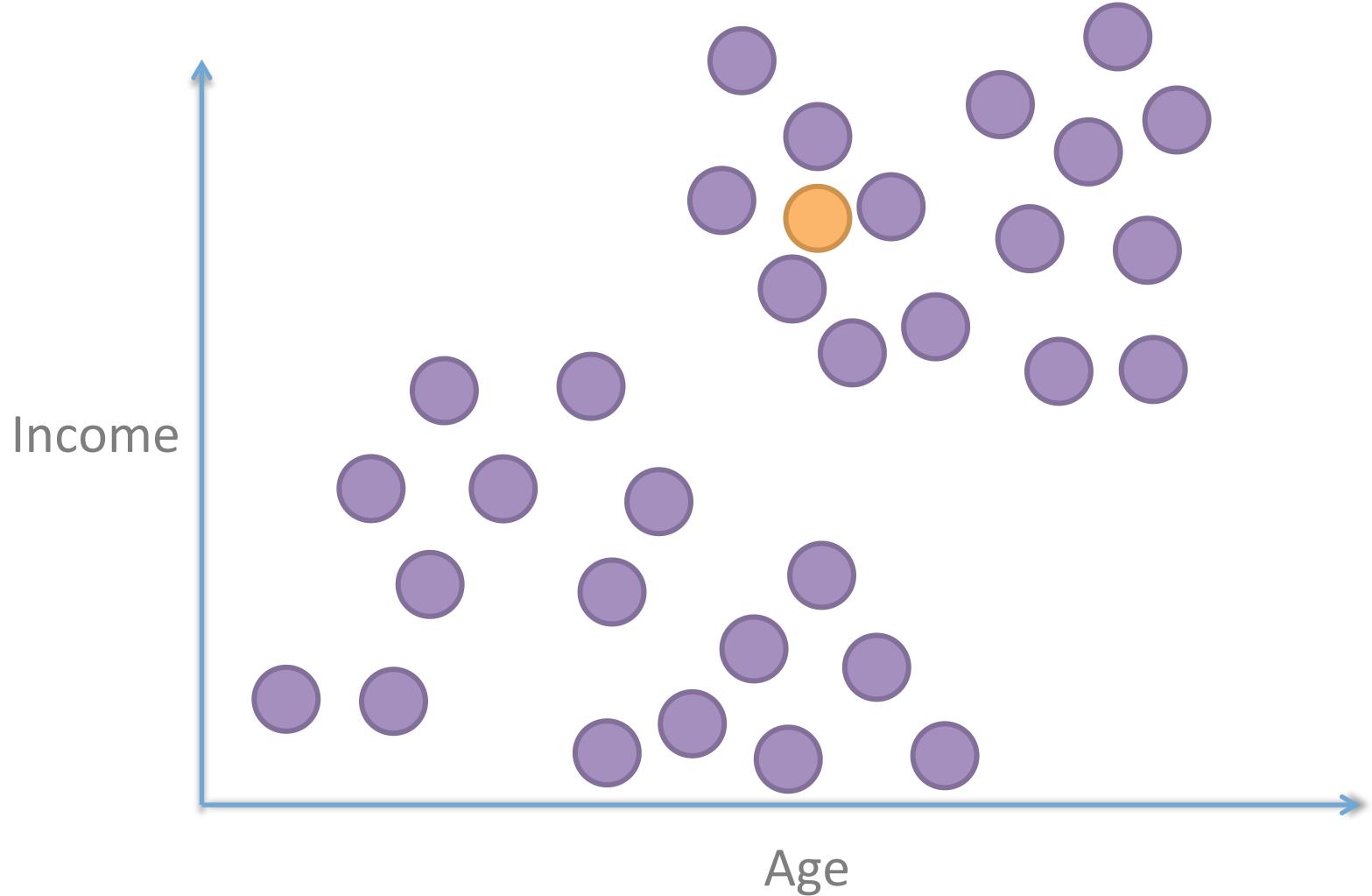
DBSCAN

Density---based spatial clustering of applications with noise



DBSCAN

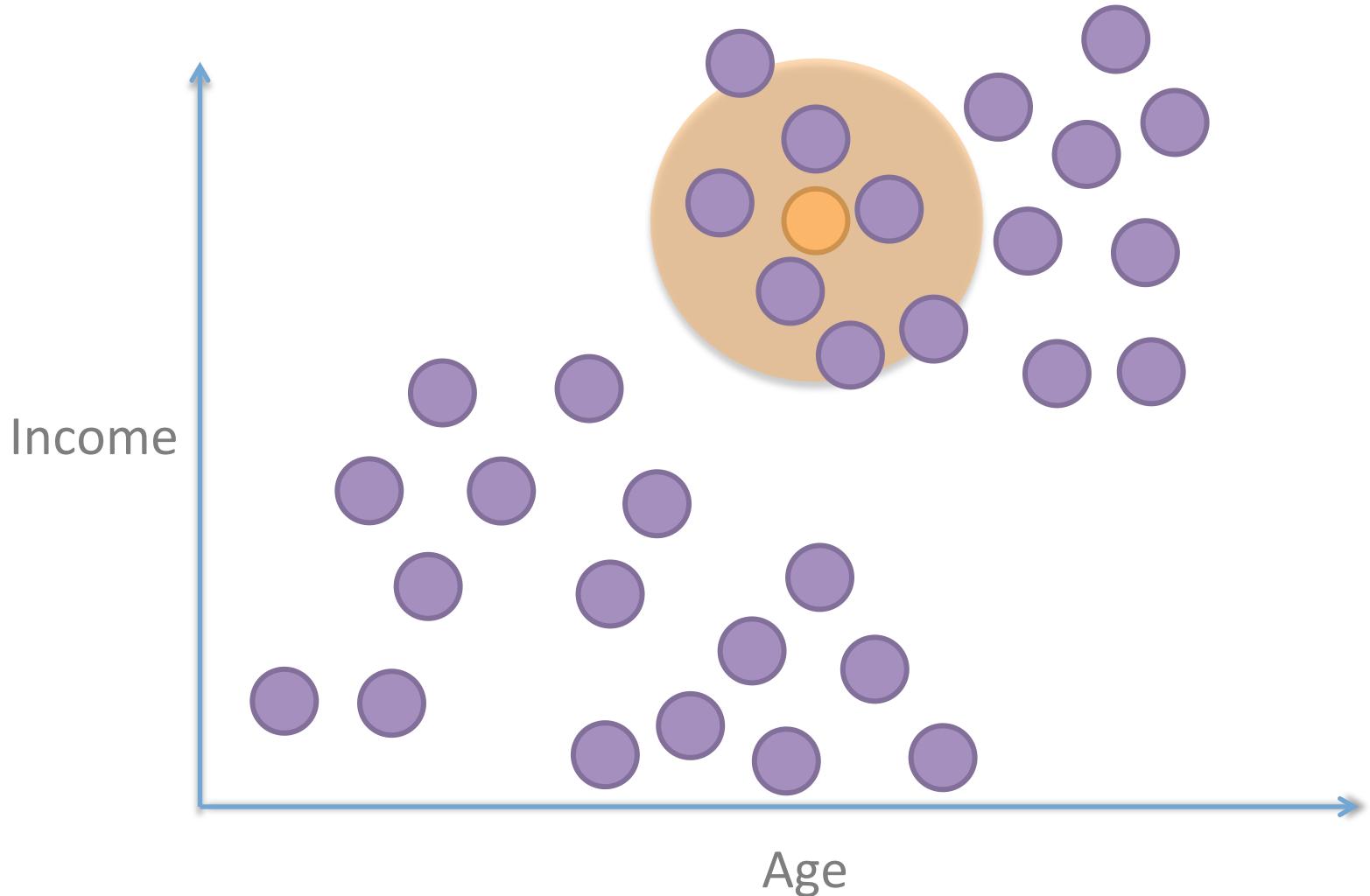
Start at a random point



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

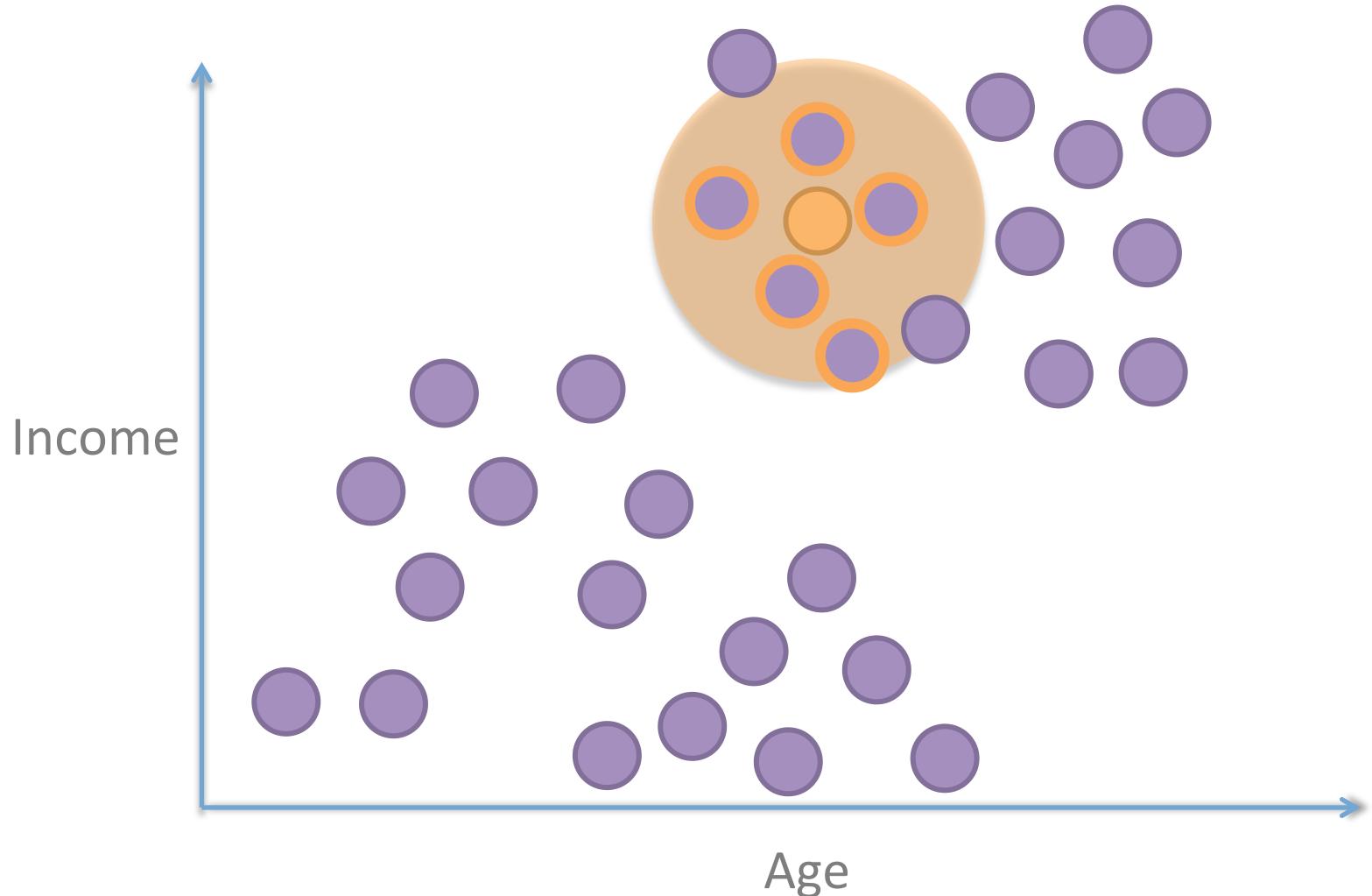
Look at the radius epsilon around point
If enough points (n_{clu}) within circle, start cluster



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

Everybody within epsilon are part of the cluster.
Process each the same way.

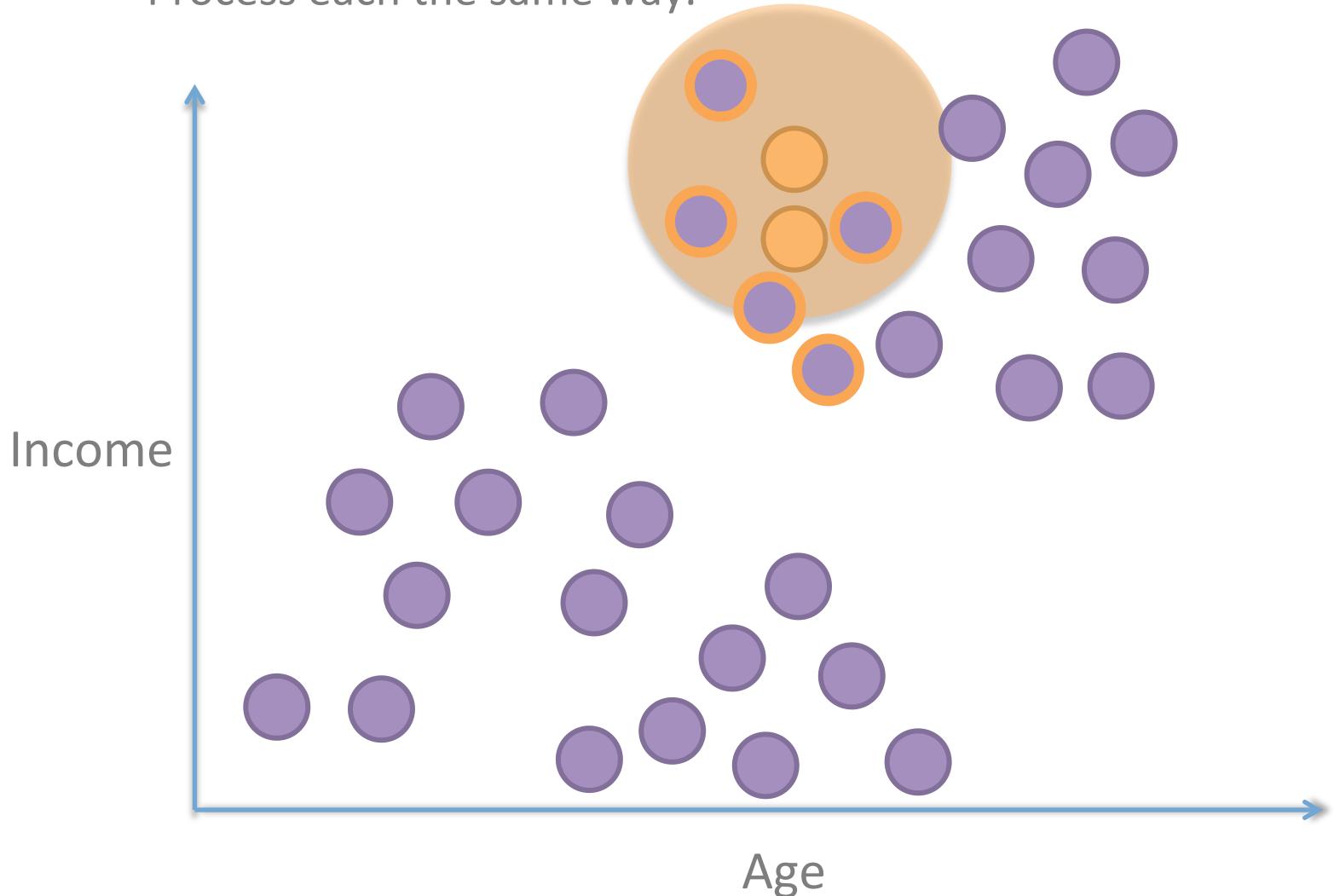


$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

Everybody within epsilon are part of the cluster.

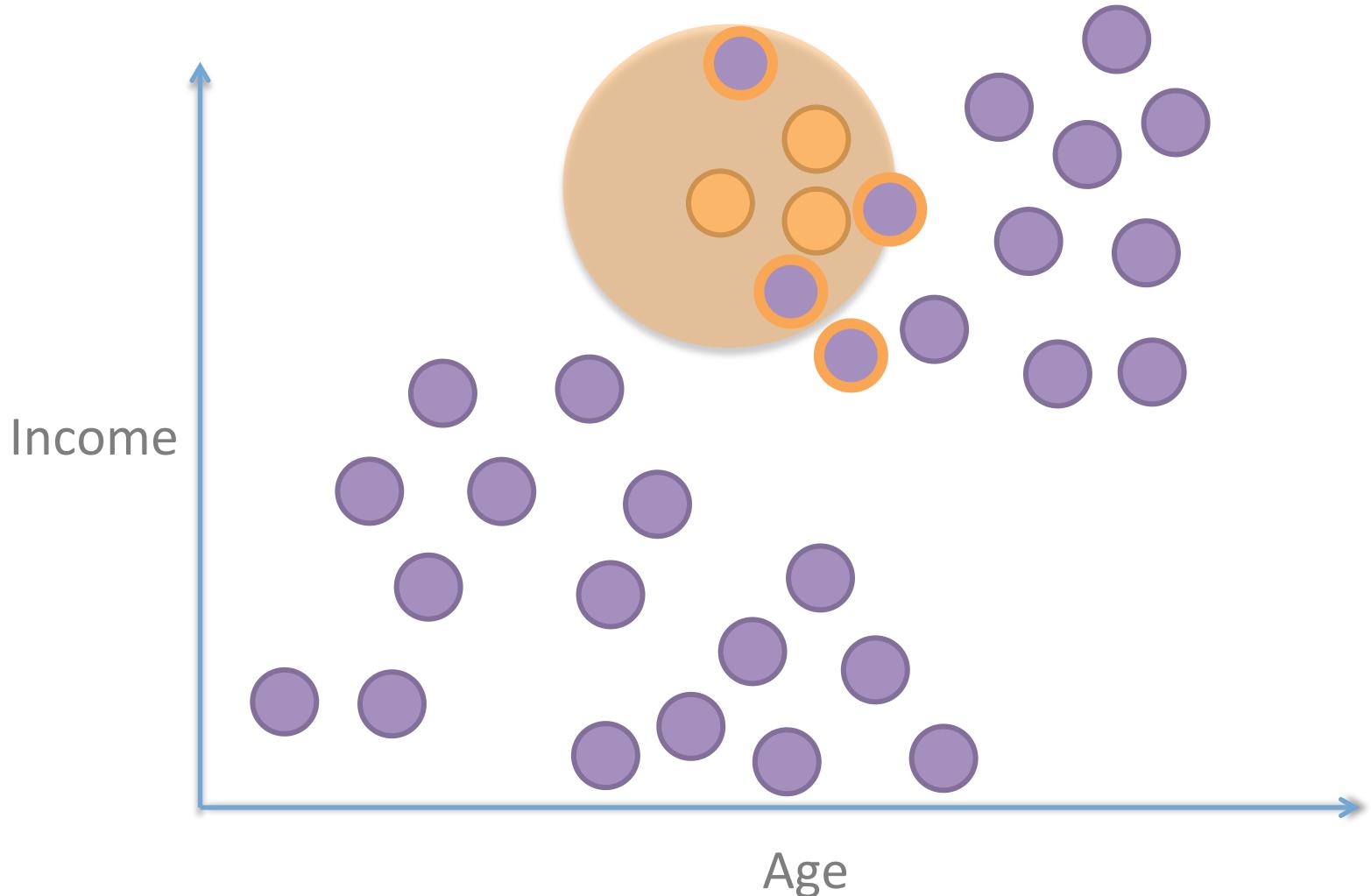
Process each the same way.



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

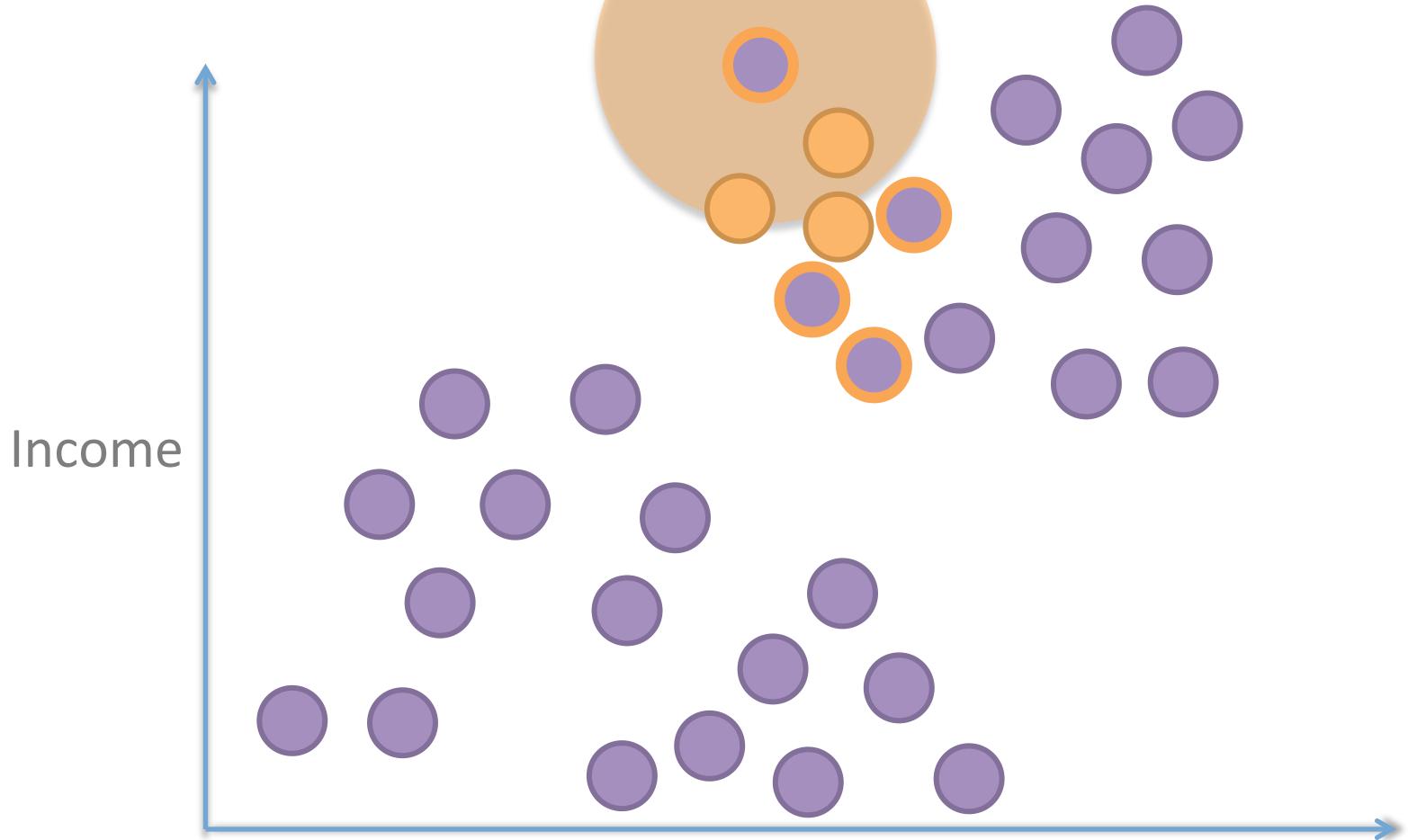
Everybody within epsilon are part of the cluster.
Process each the same way.



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

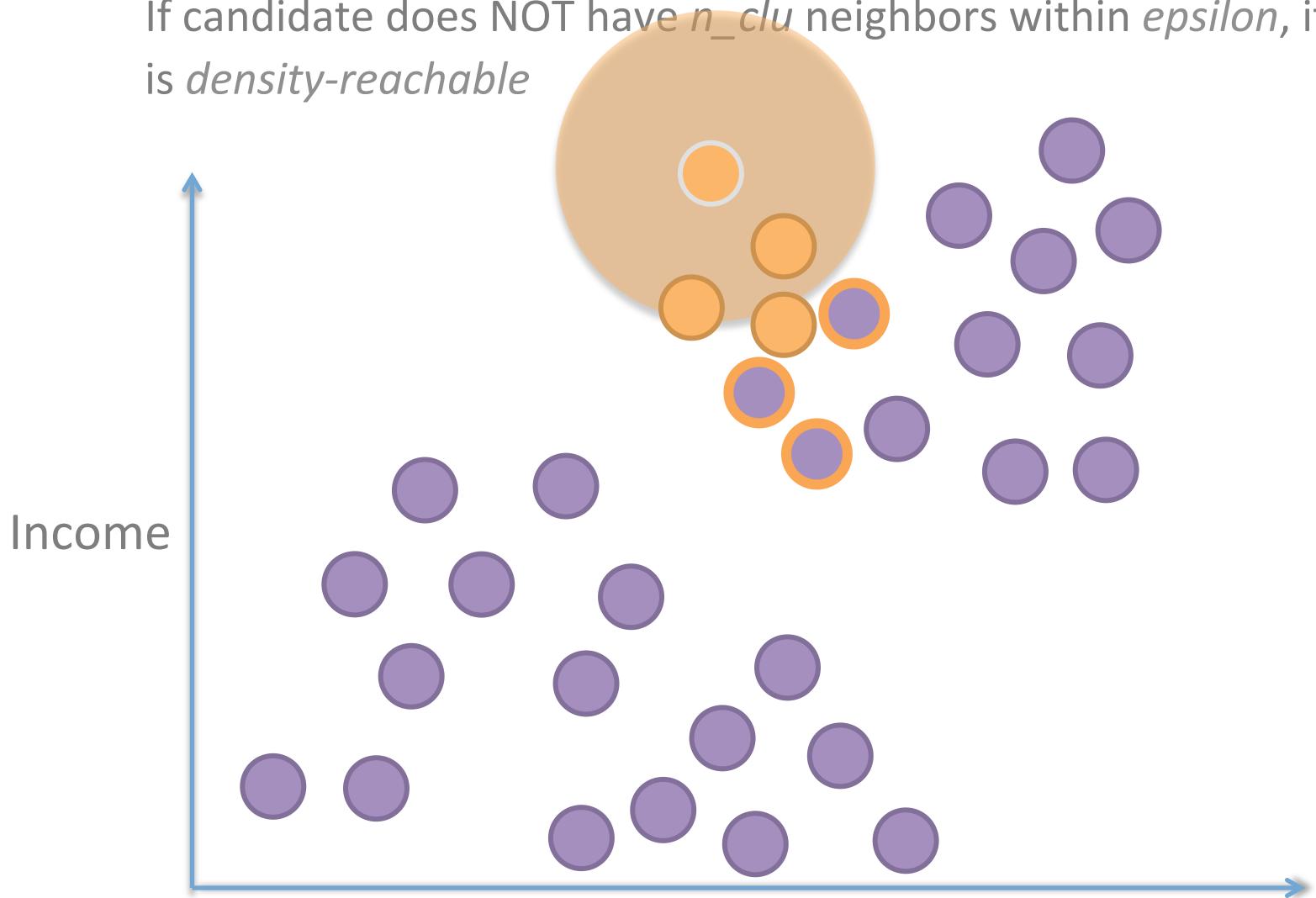
If candidate does NOT have n_{clu} neighbors within epsilon , it is *density-reachable*



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

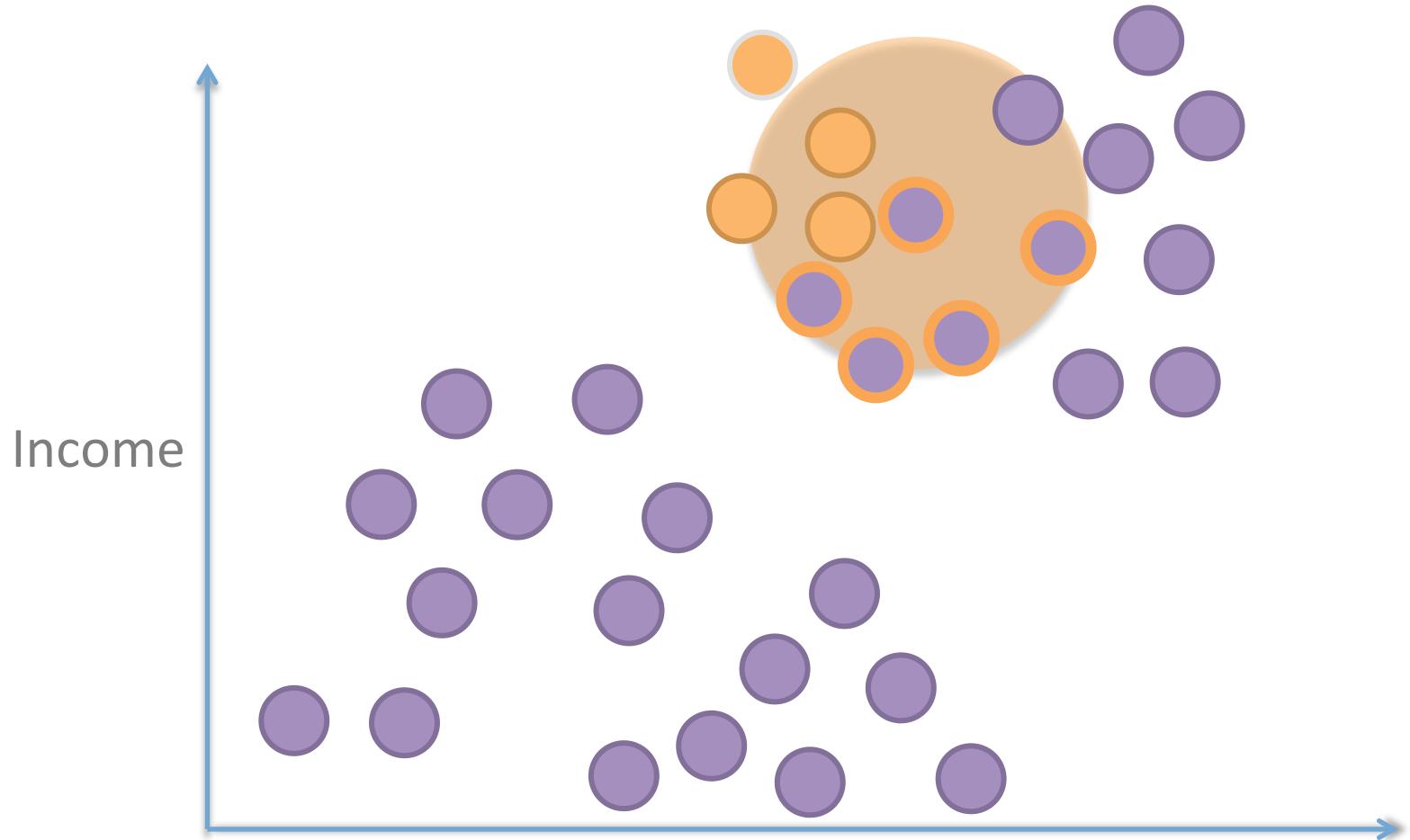
If candidate does NOT have n_{clu} neighbors within epsilon , it is *density-reachable*



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

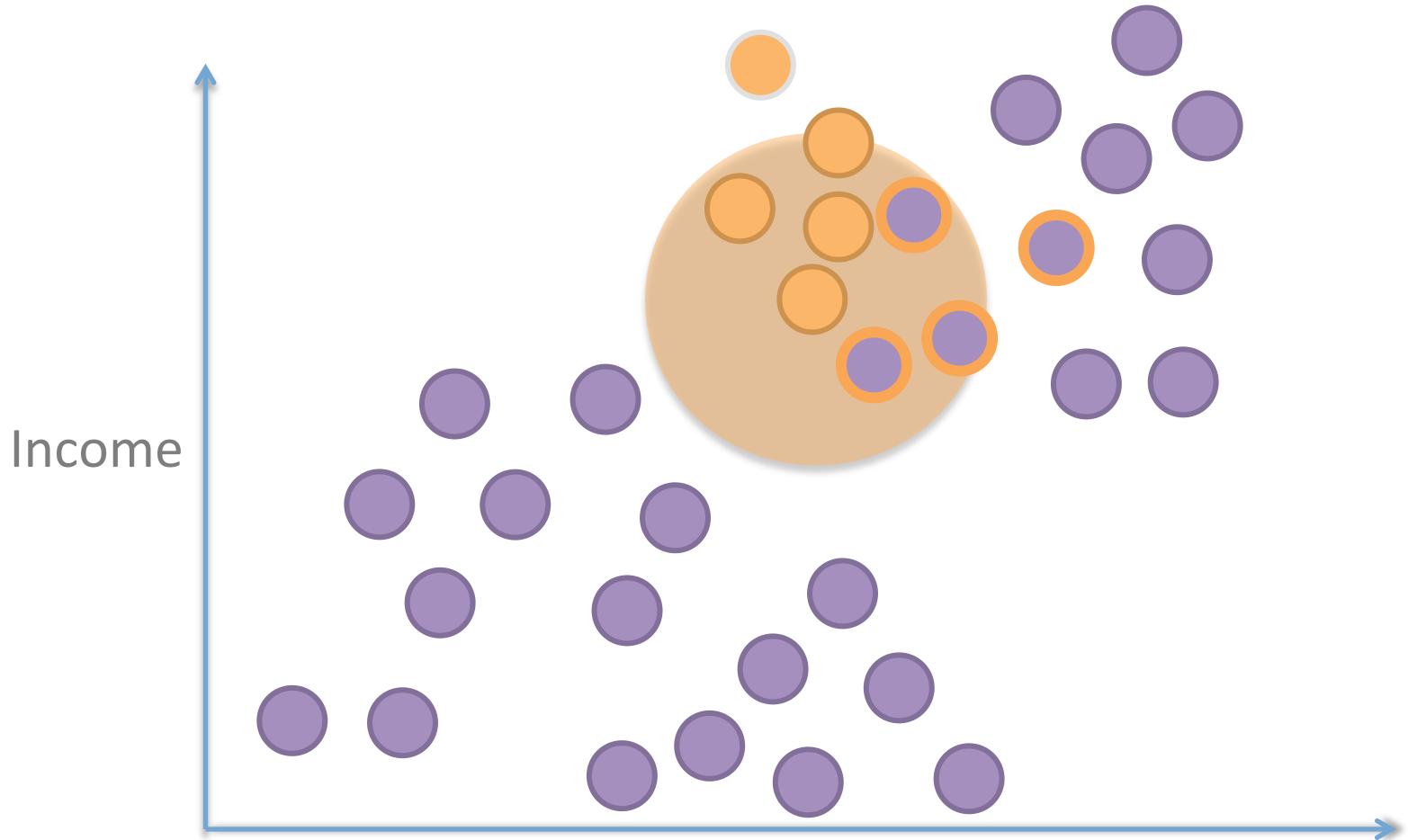
Keep going until you find the entire cluster and no point is left unvisited by this chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

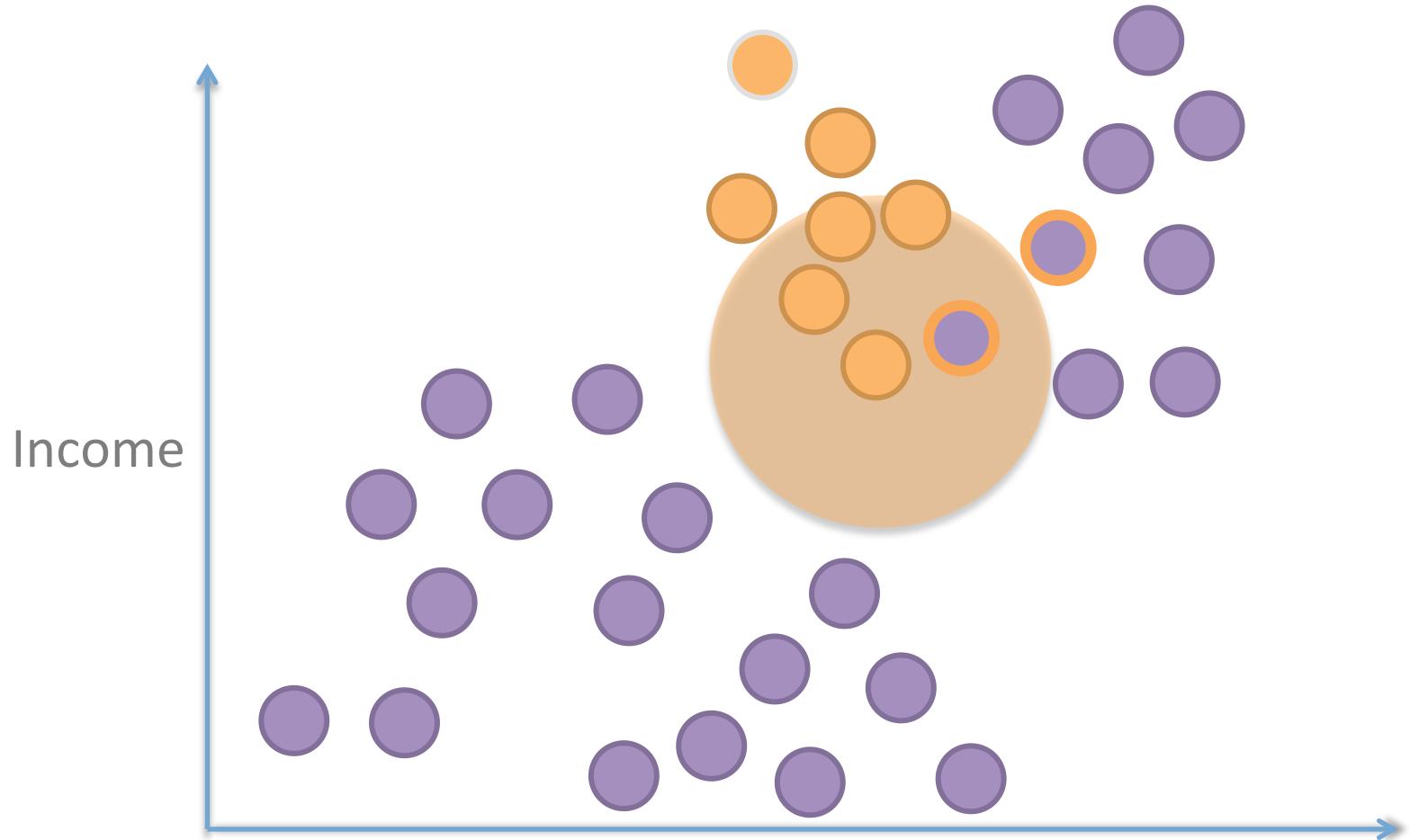
Keep going until you find the entire cluster and no point is left unvisited by this chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

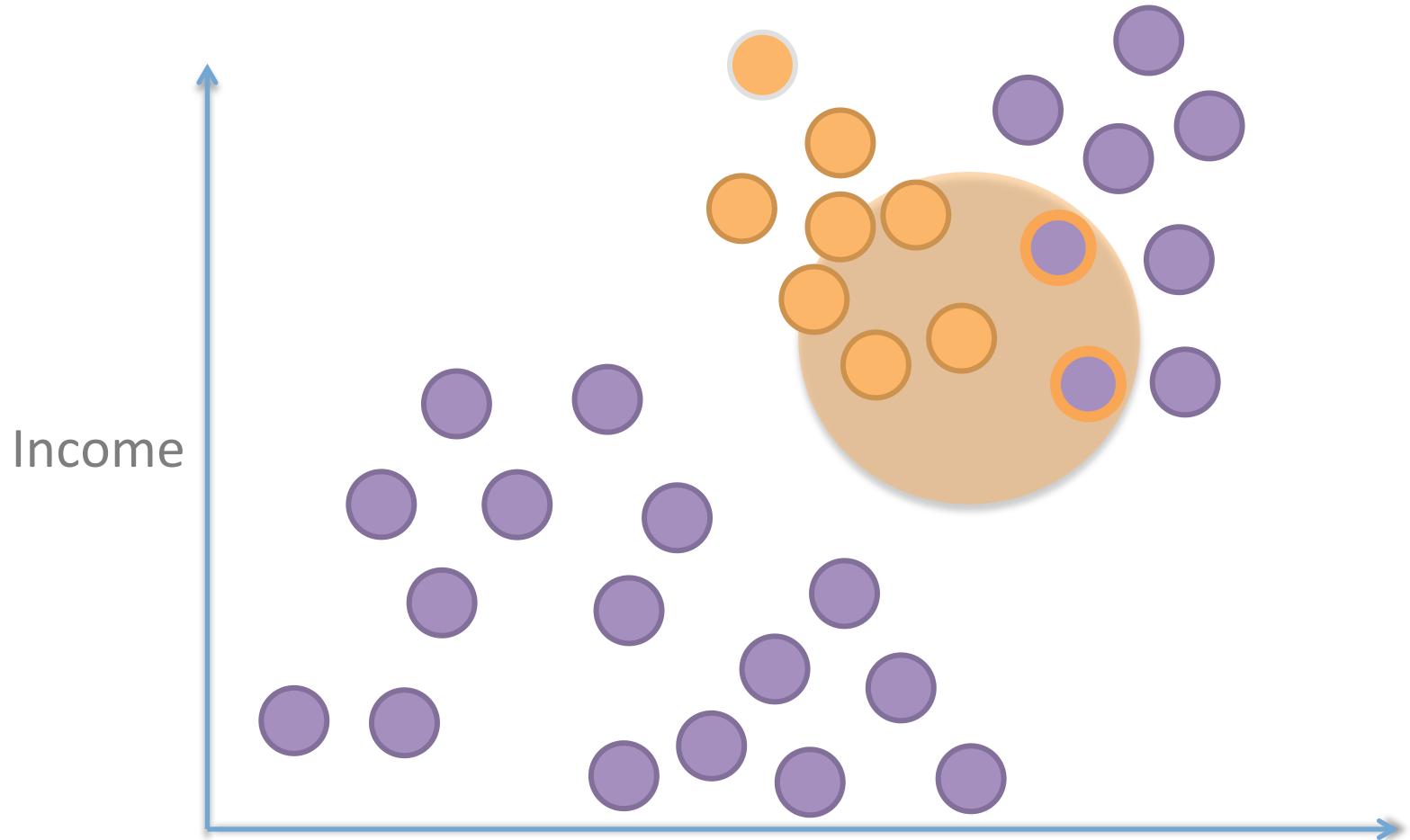
Keep going until you find the entire cluster and no point is left unvisited by this chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

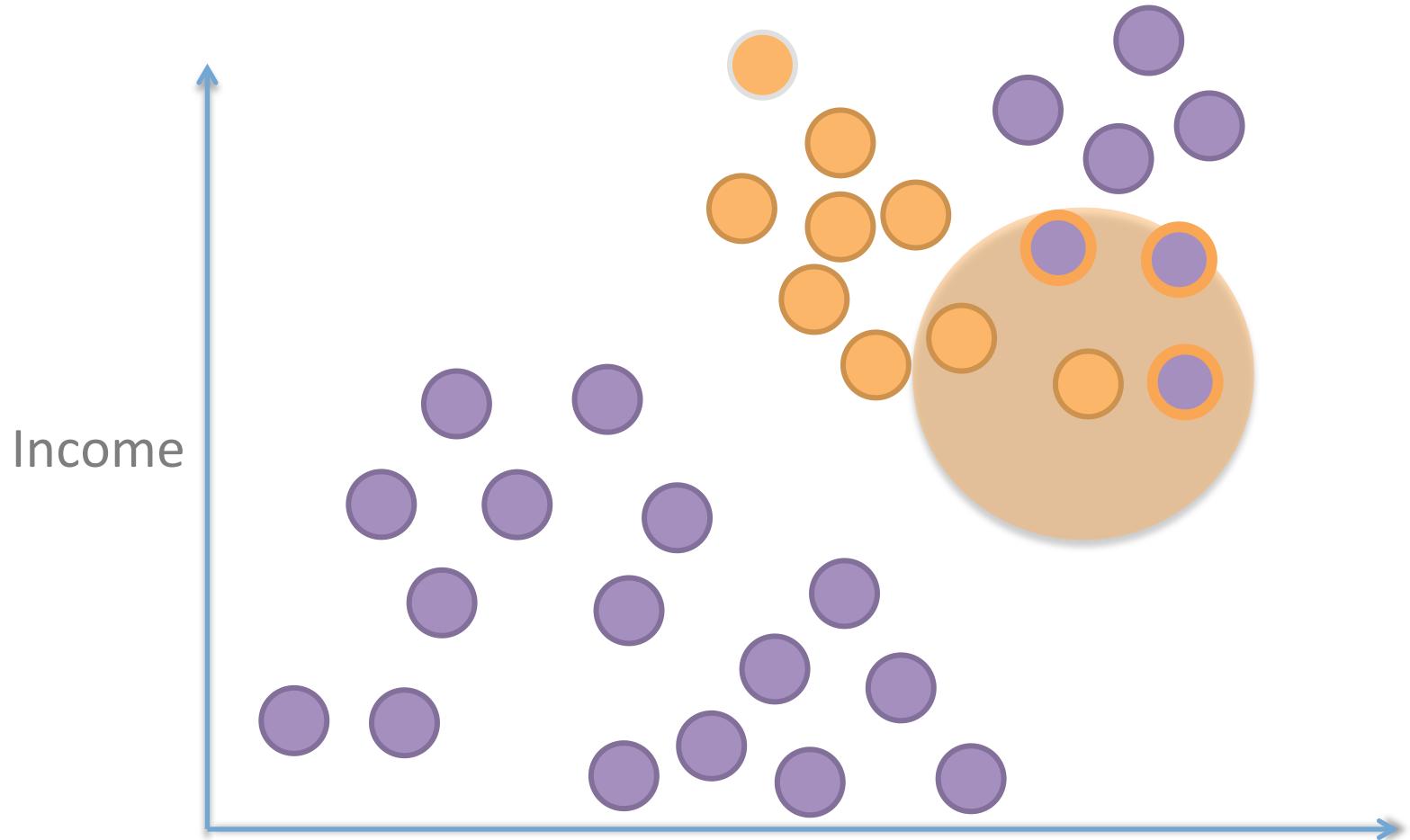
Keep going until you find the entire cluster and no point is left unvisited by this chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

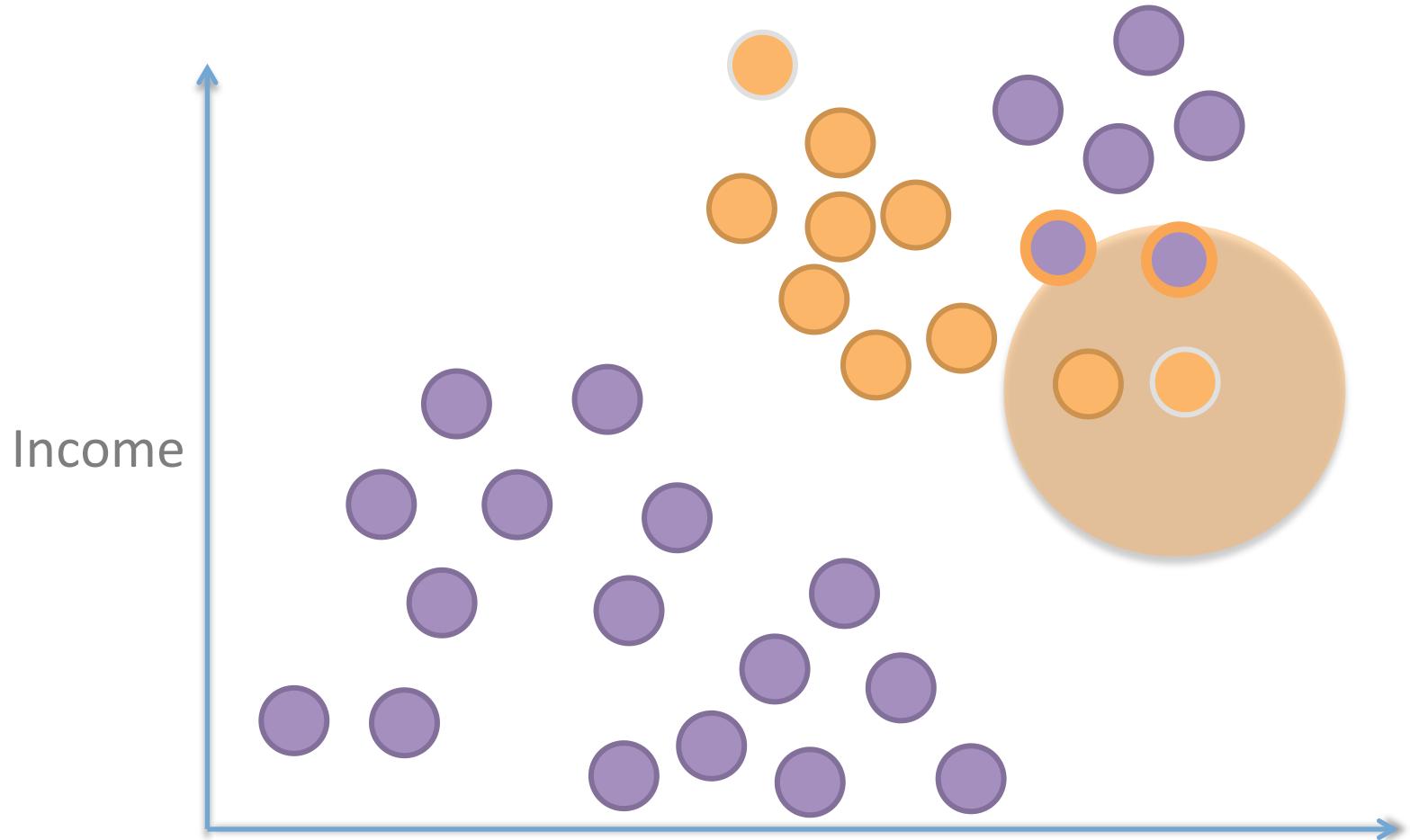
Keep going until you find the entire cluster and no point is left unvisited by this chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

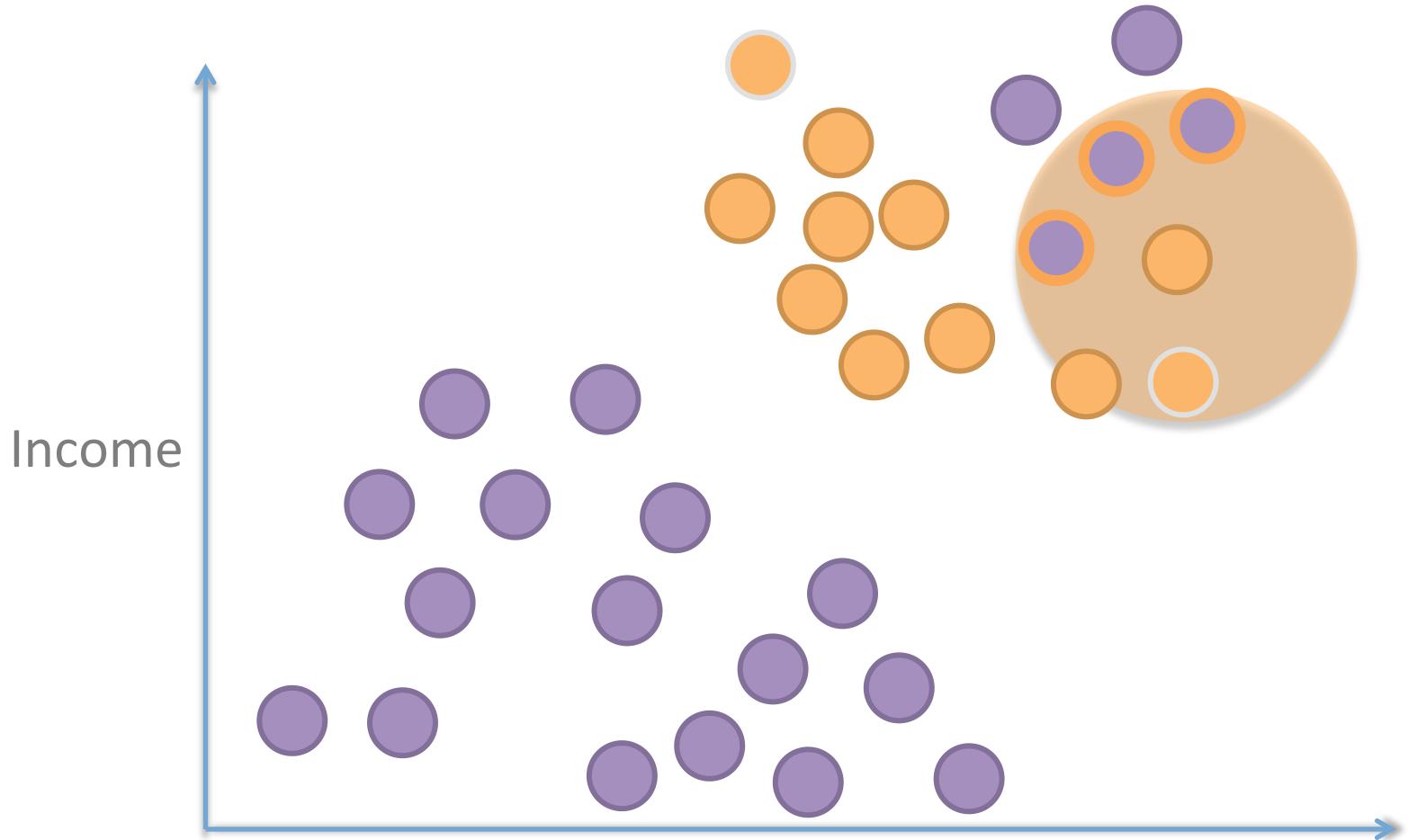
Keep going until you find the entire cluster and no point is left unvisited by this chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

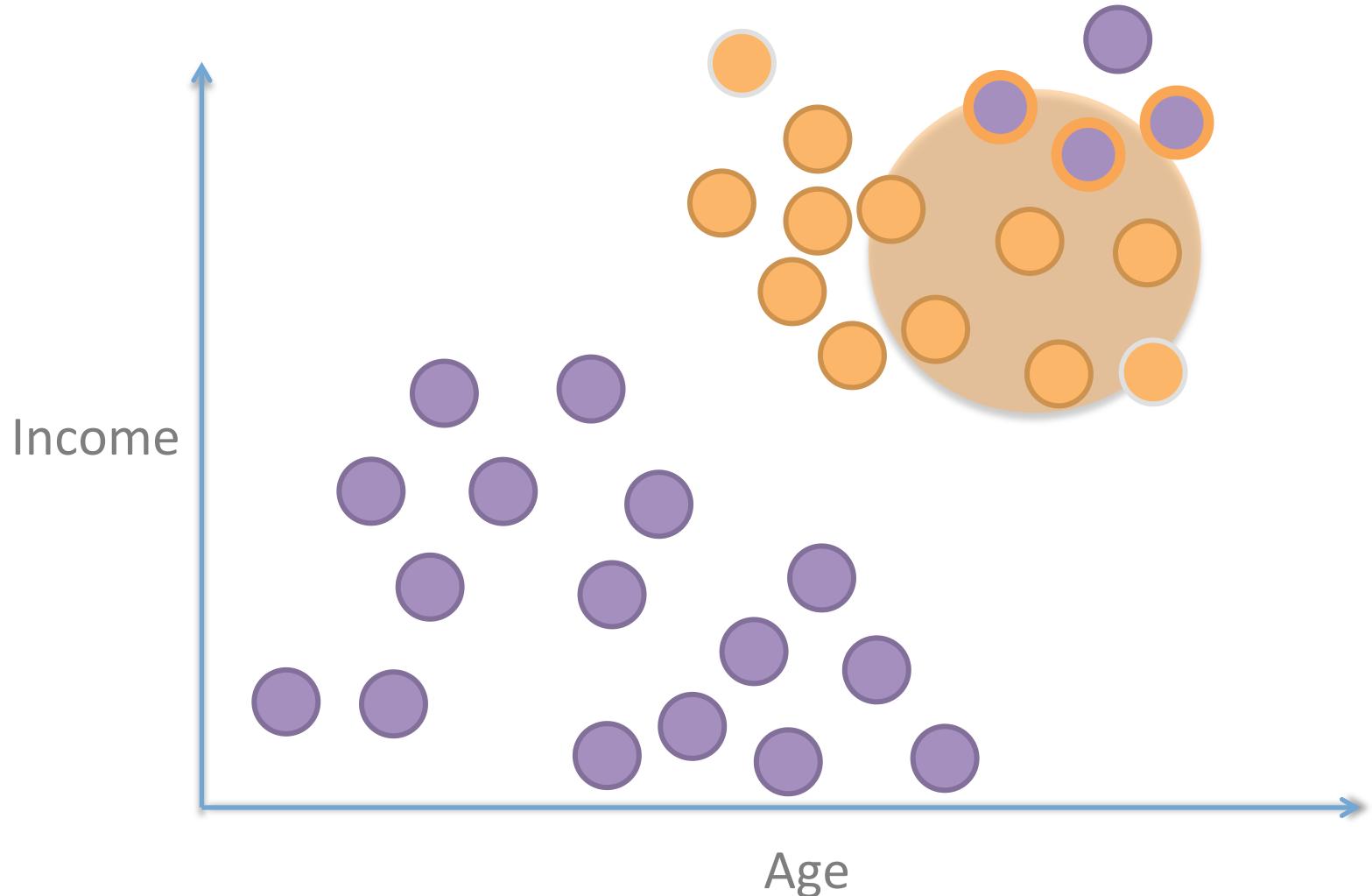
Keep going until you find the entire cluster and no point is left unvisited by this chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

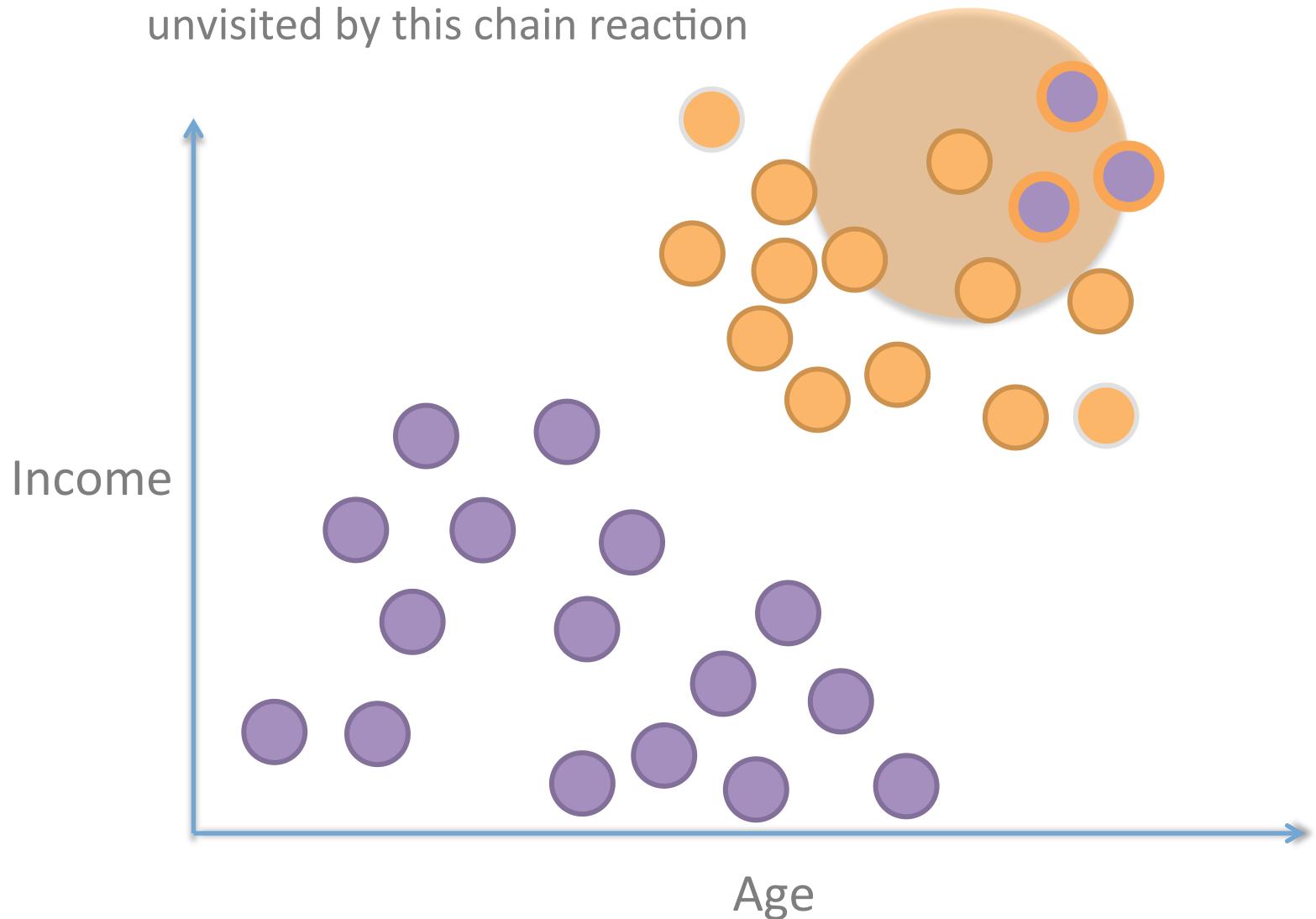
Keep going until you find the entire cluster and no point is left unvisited by this chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

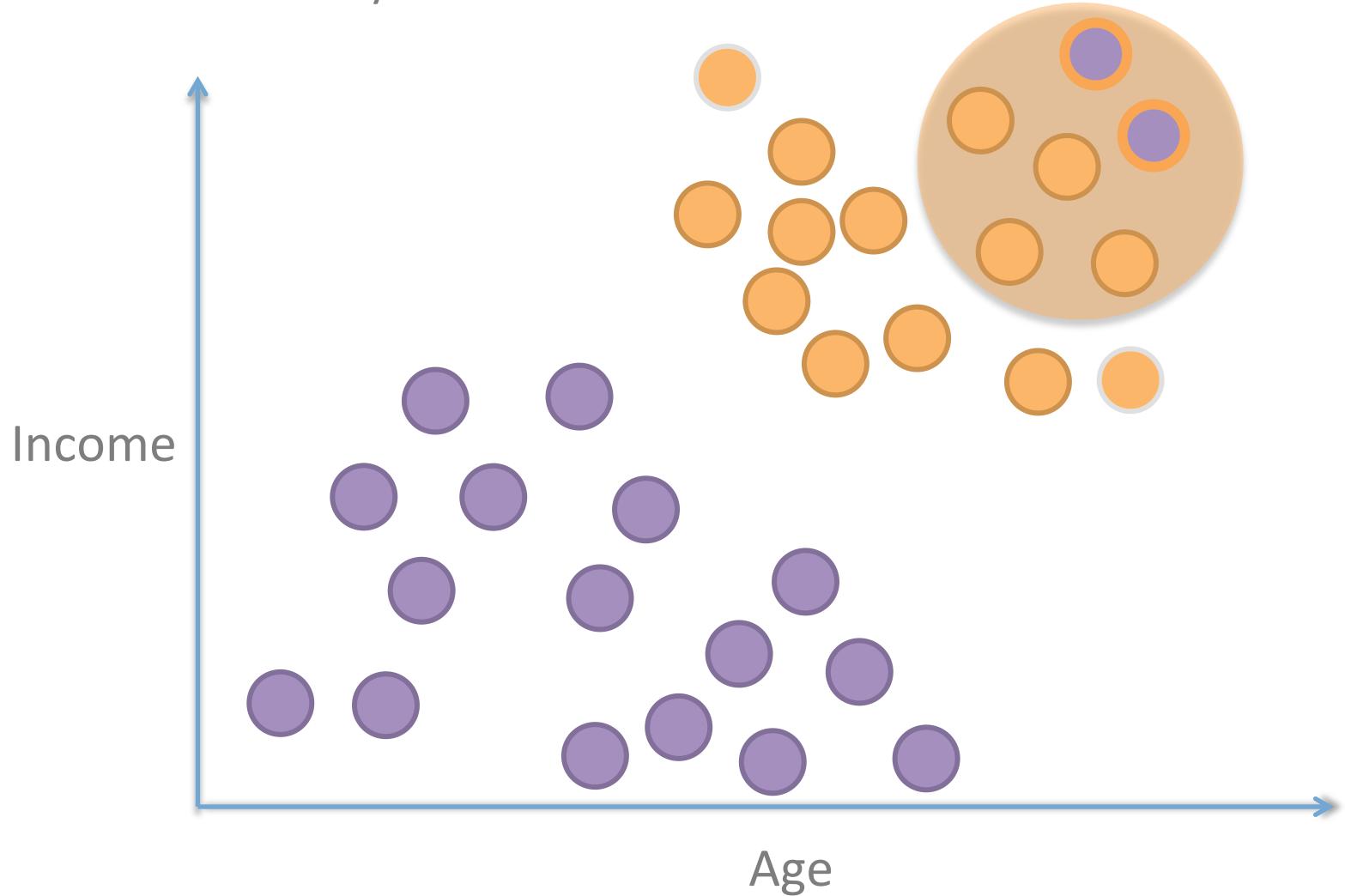
Keep going until you find the entire cluster and no point is left unvisited by this chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

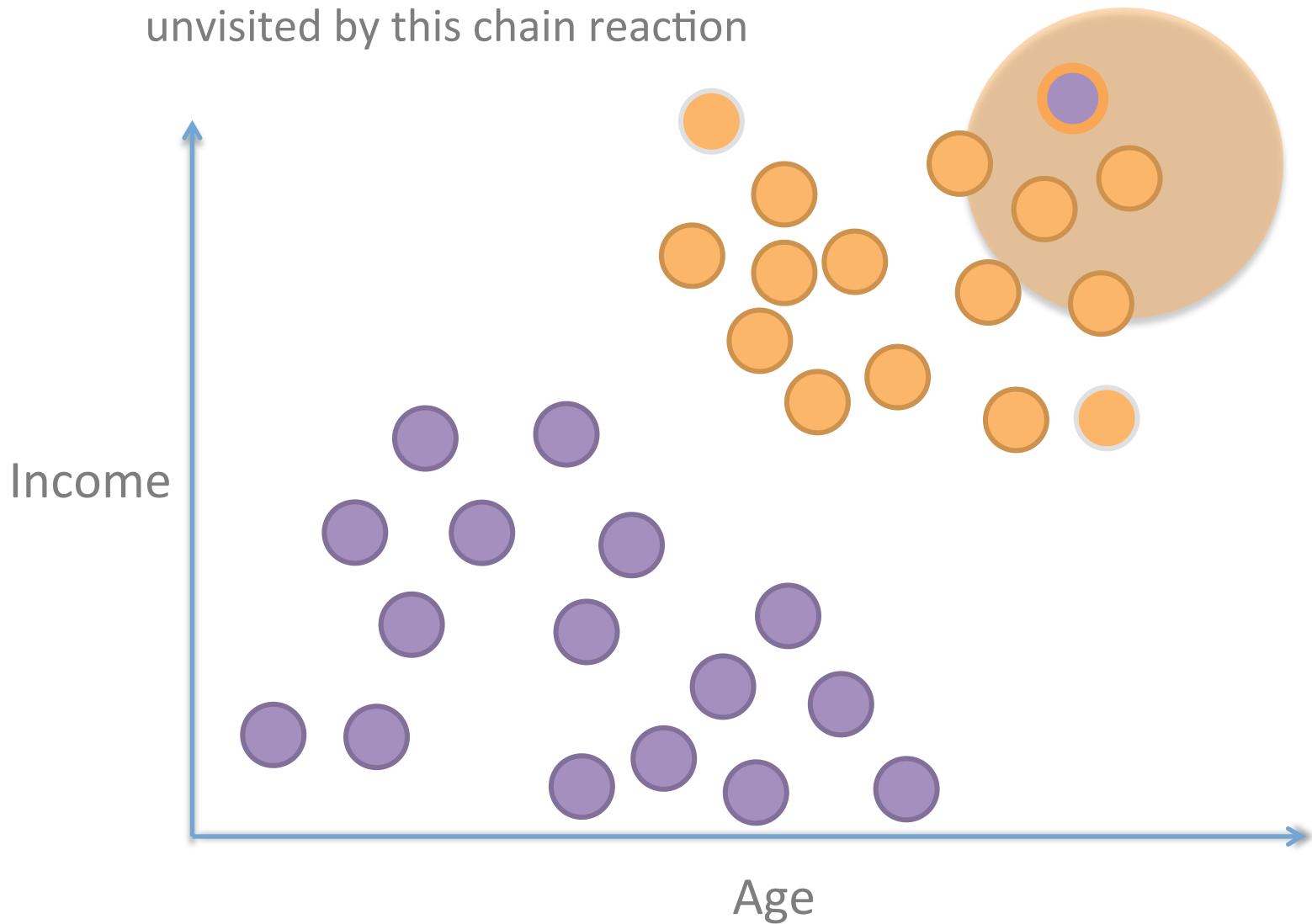
Keep going until you find the entire cluster and no point is left unvisited by this chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

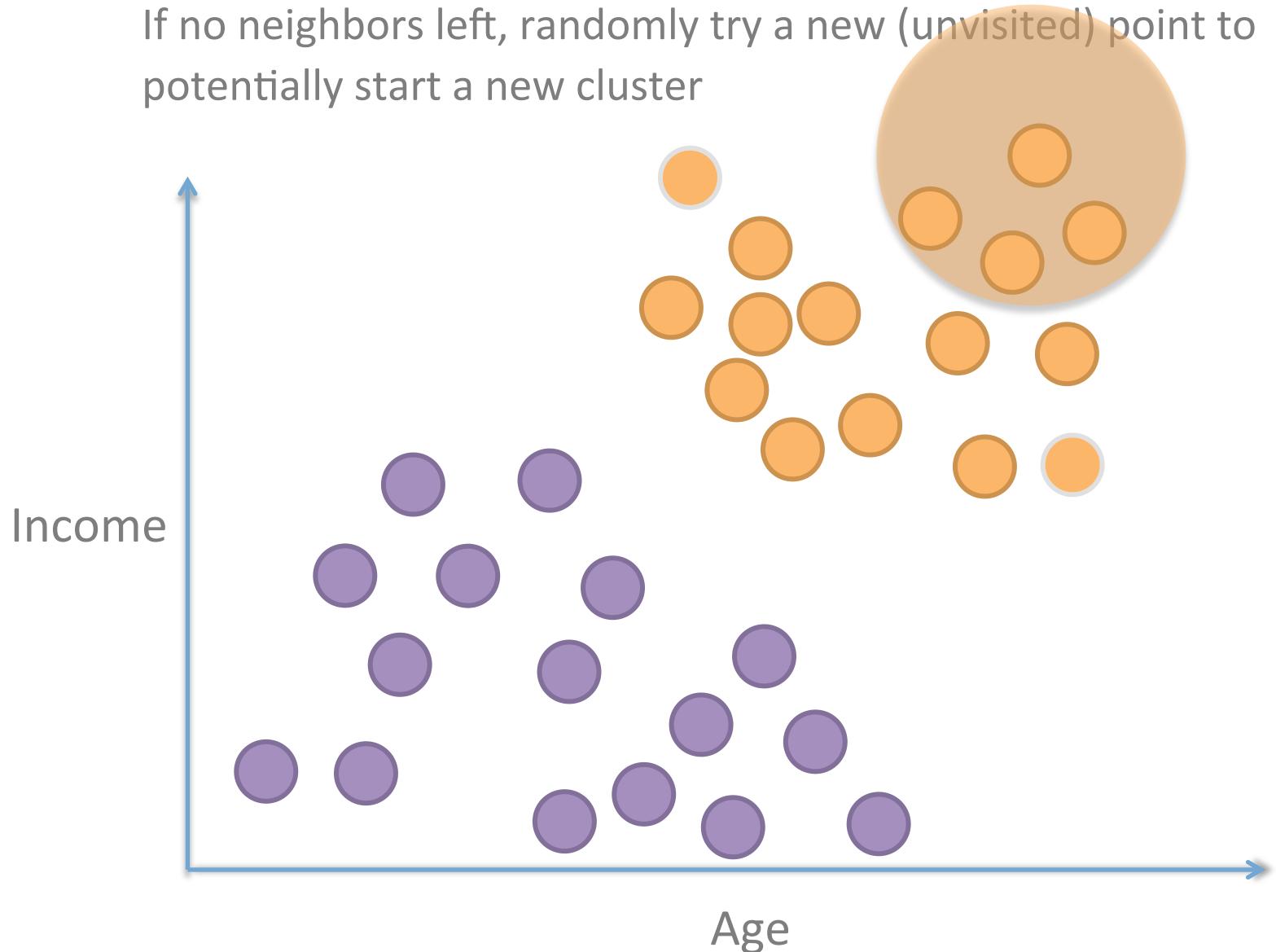
Keep going until you find the entire cluster and no point is left unvisited by this chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

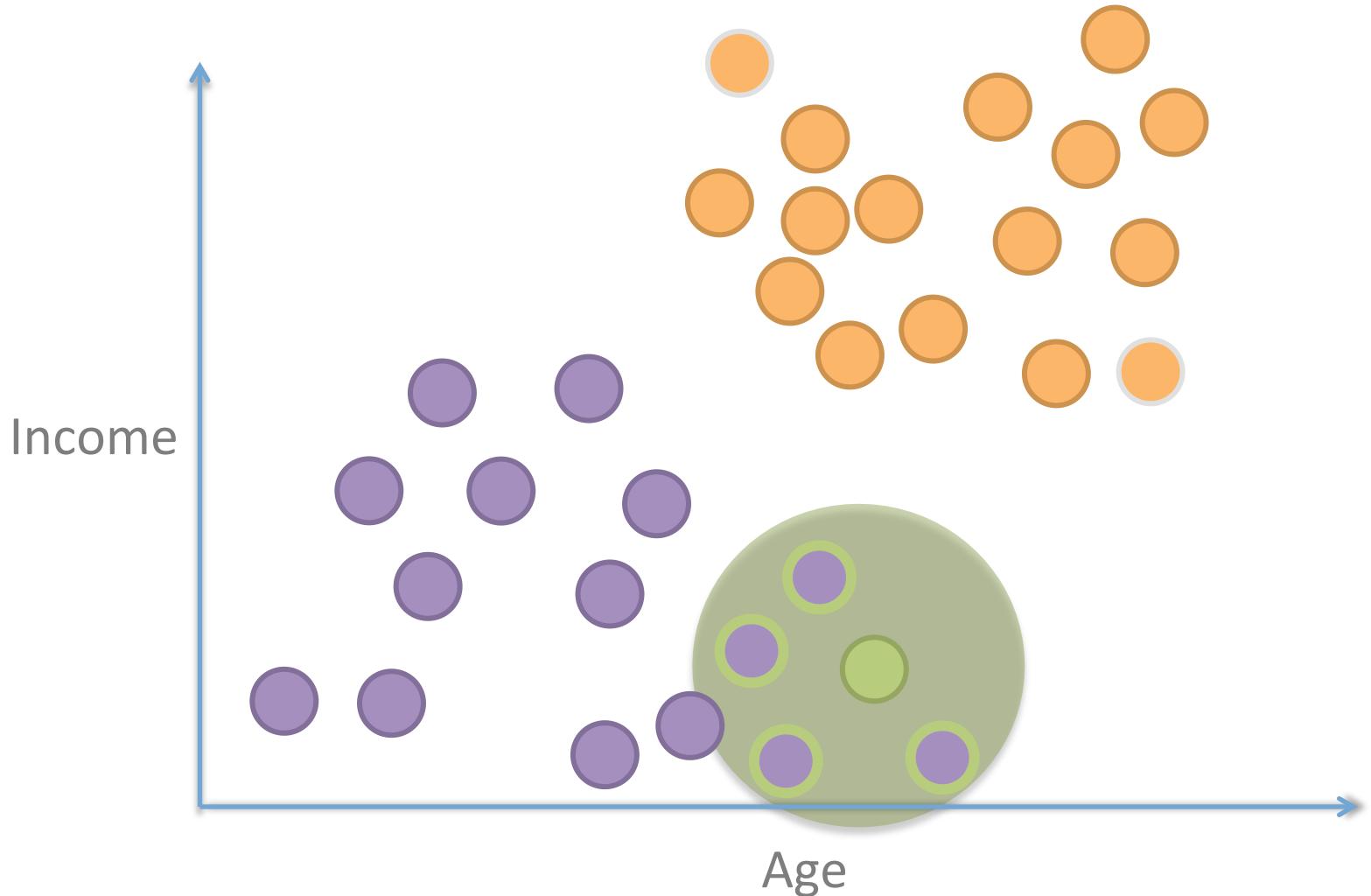
If no neighbors left, randomly try a new (unvisited) point to potentially start a new cluster



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

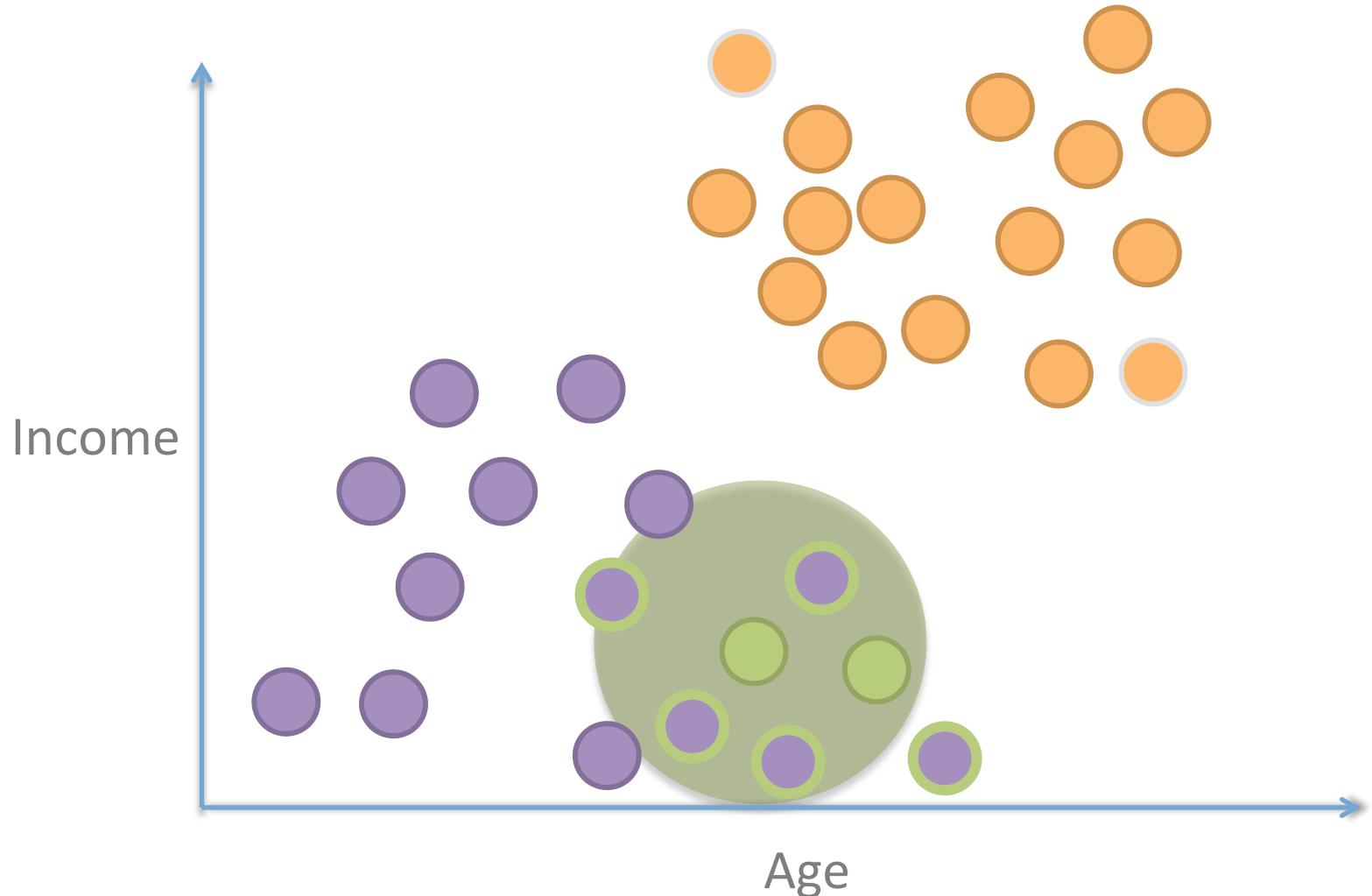
If no neighbors left, randomly try a new (unvisited) point to potentially start a new cluster



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

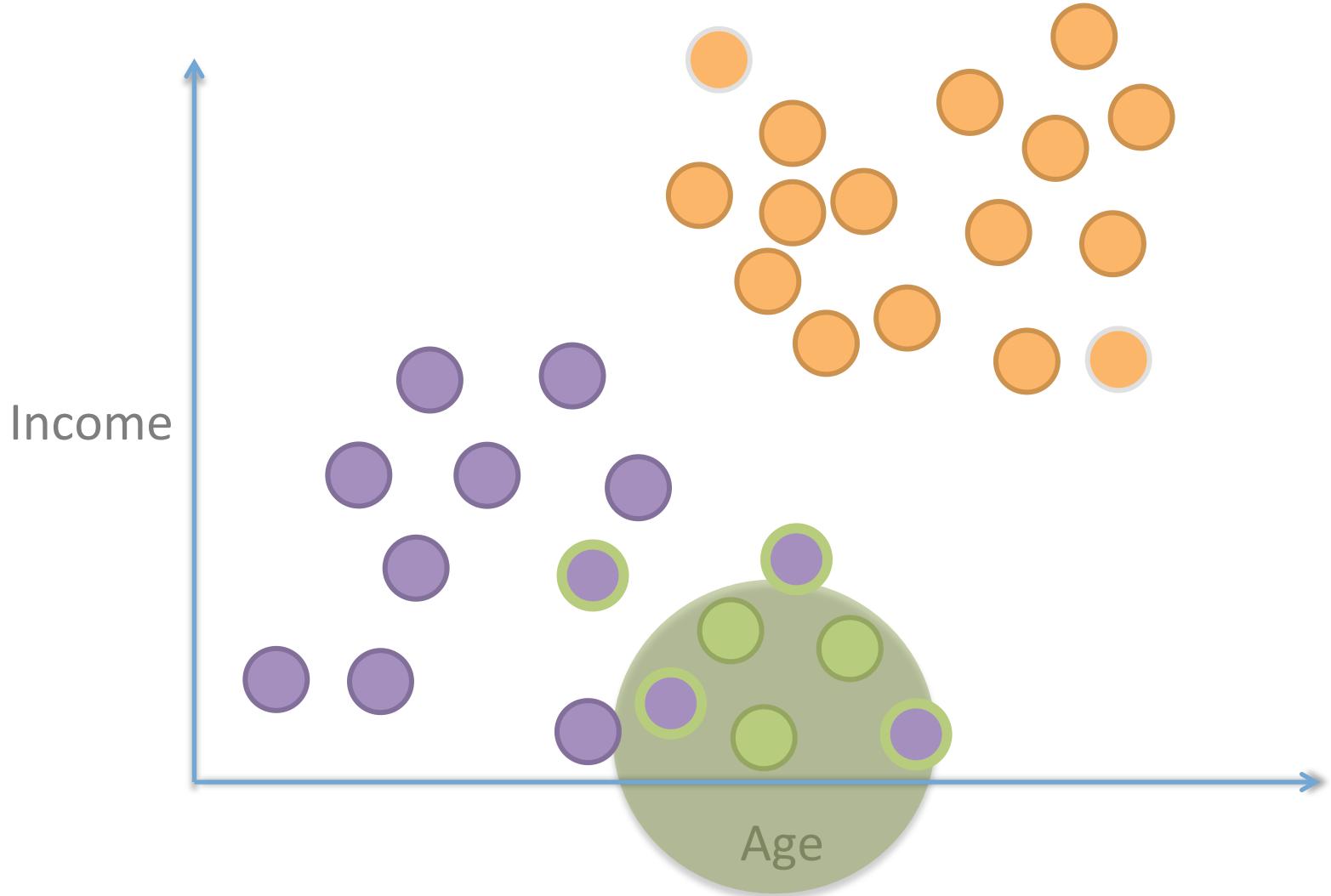
Keep adding neighbors within epsilon as chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

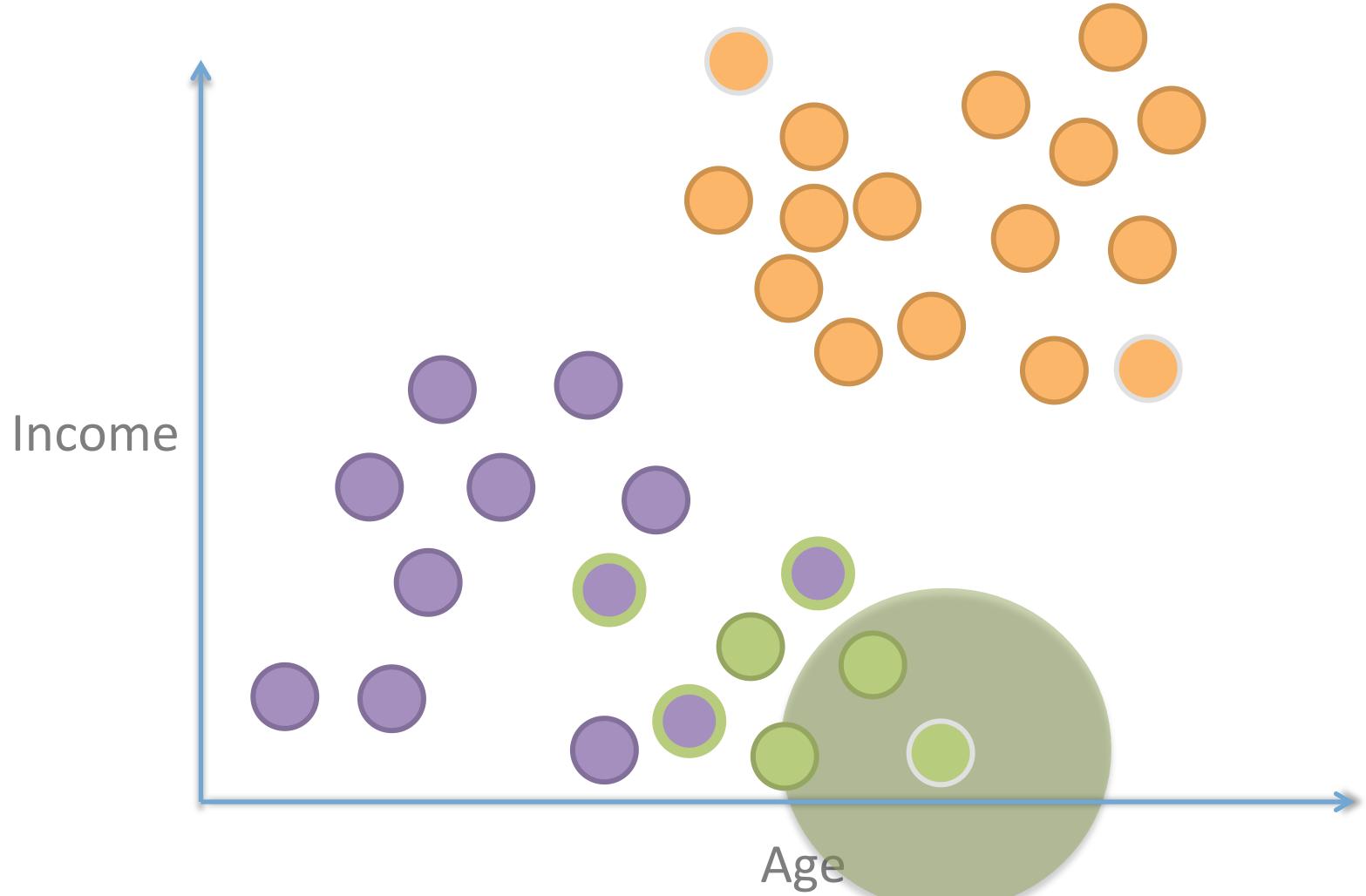
Keep adding neighbors within epsilon as chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

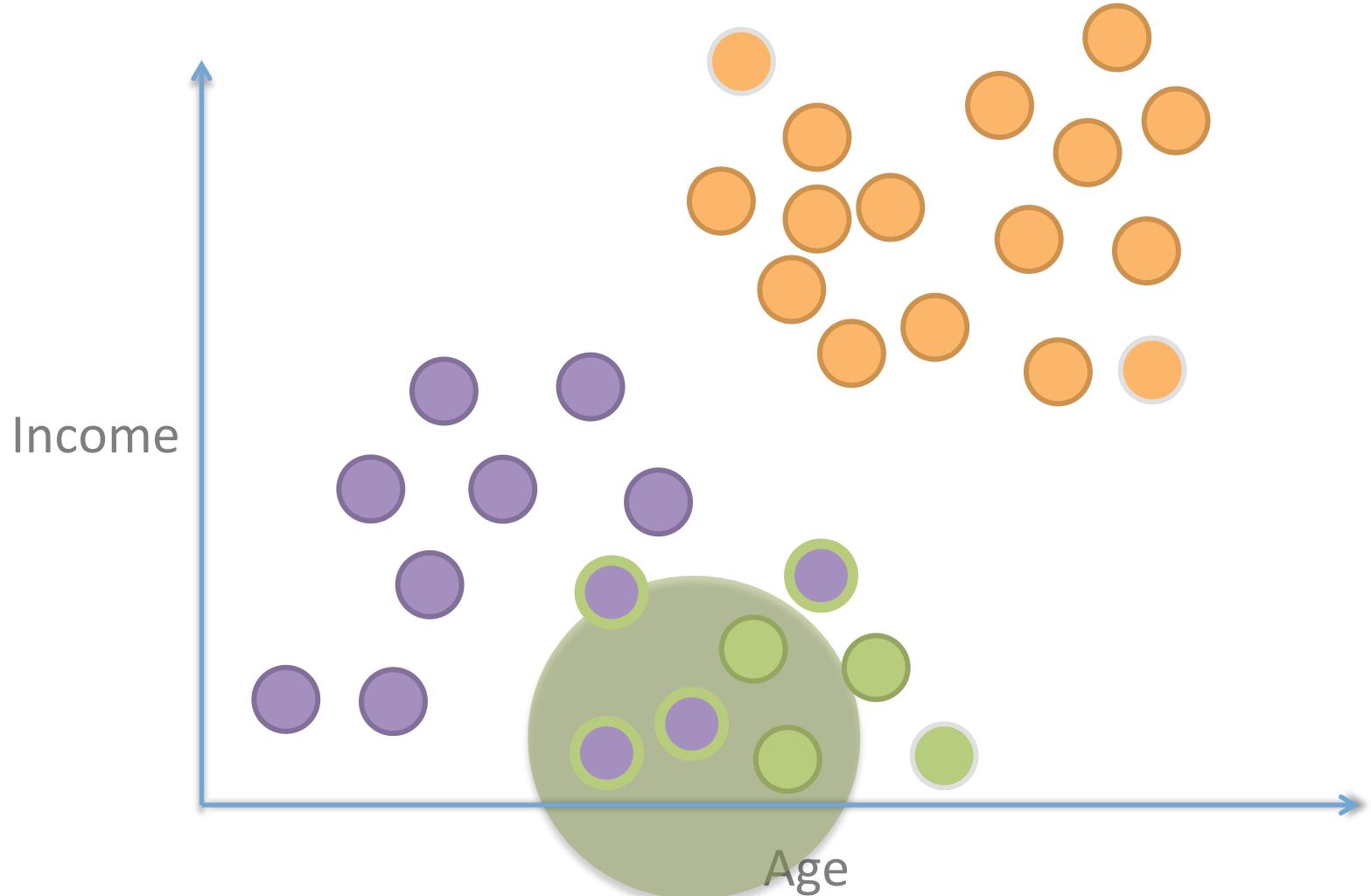
Keep adding neighbors within epsilon as chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

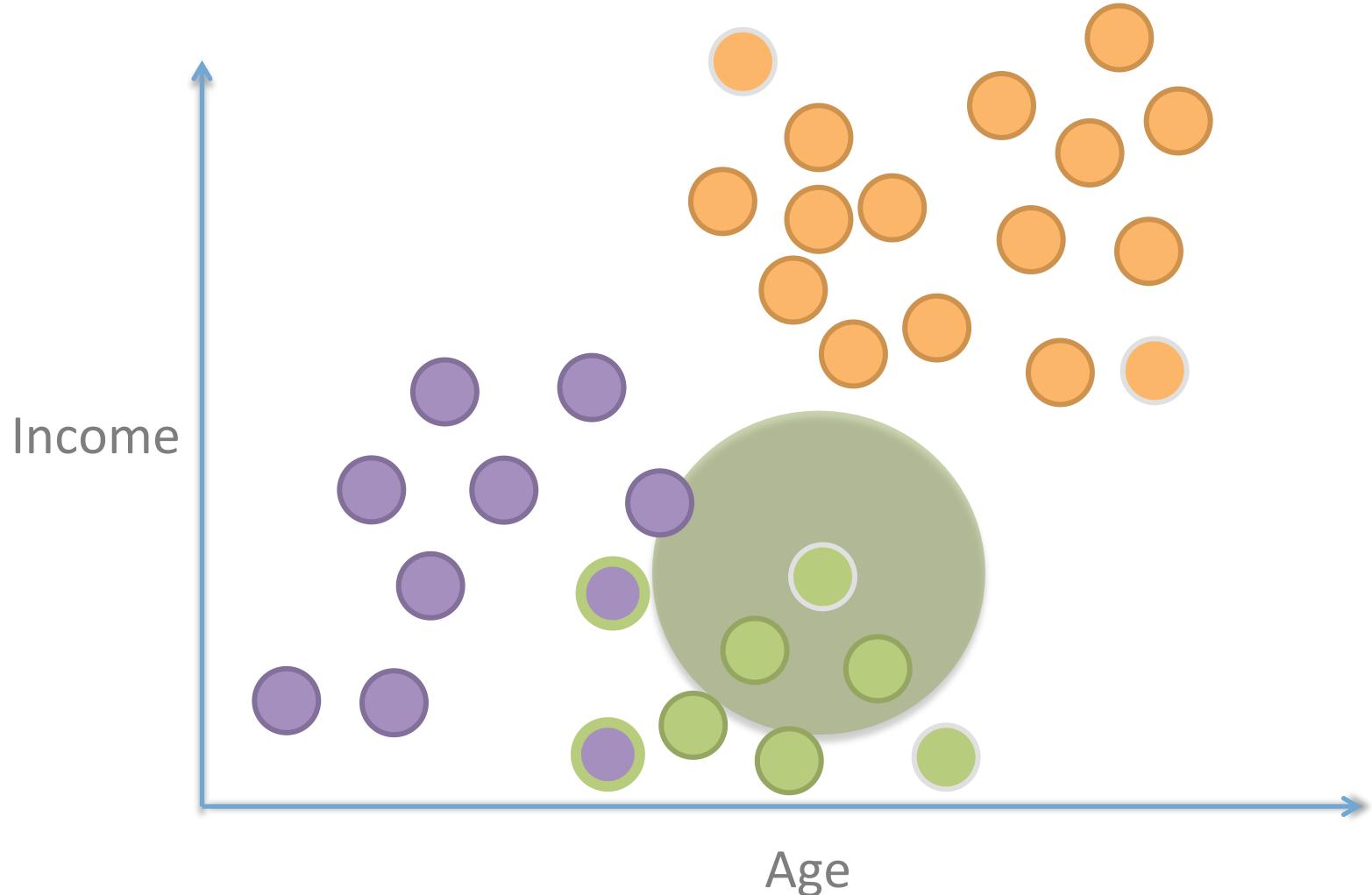
Keep adding neighbors within epsilon as chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

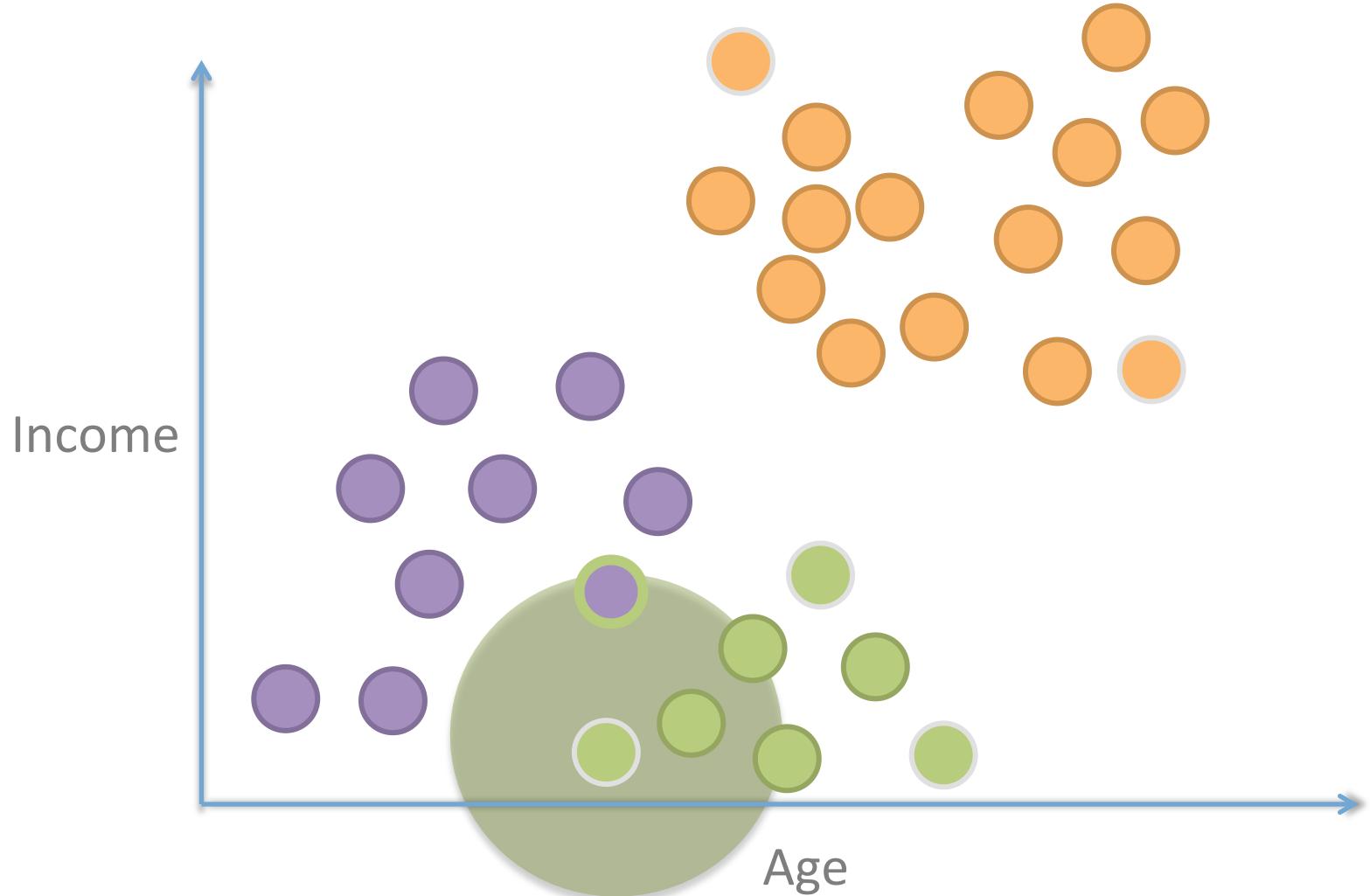
Keep adding neighbors within epsilon as chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

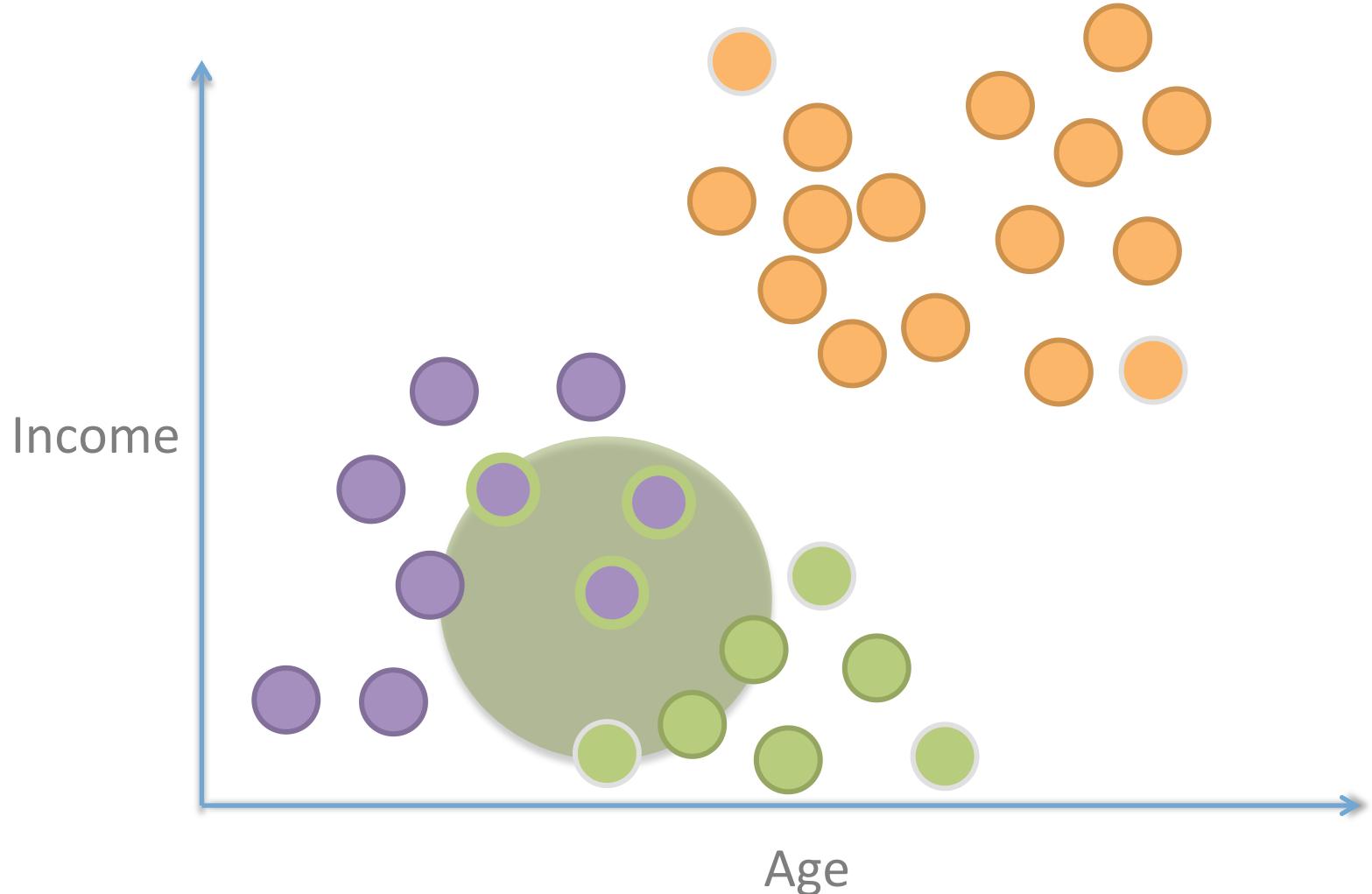
Keep adding neighbors within epsilon as chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

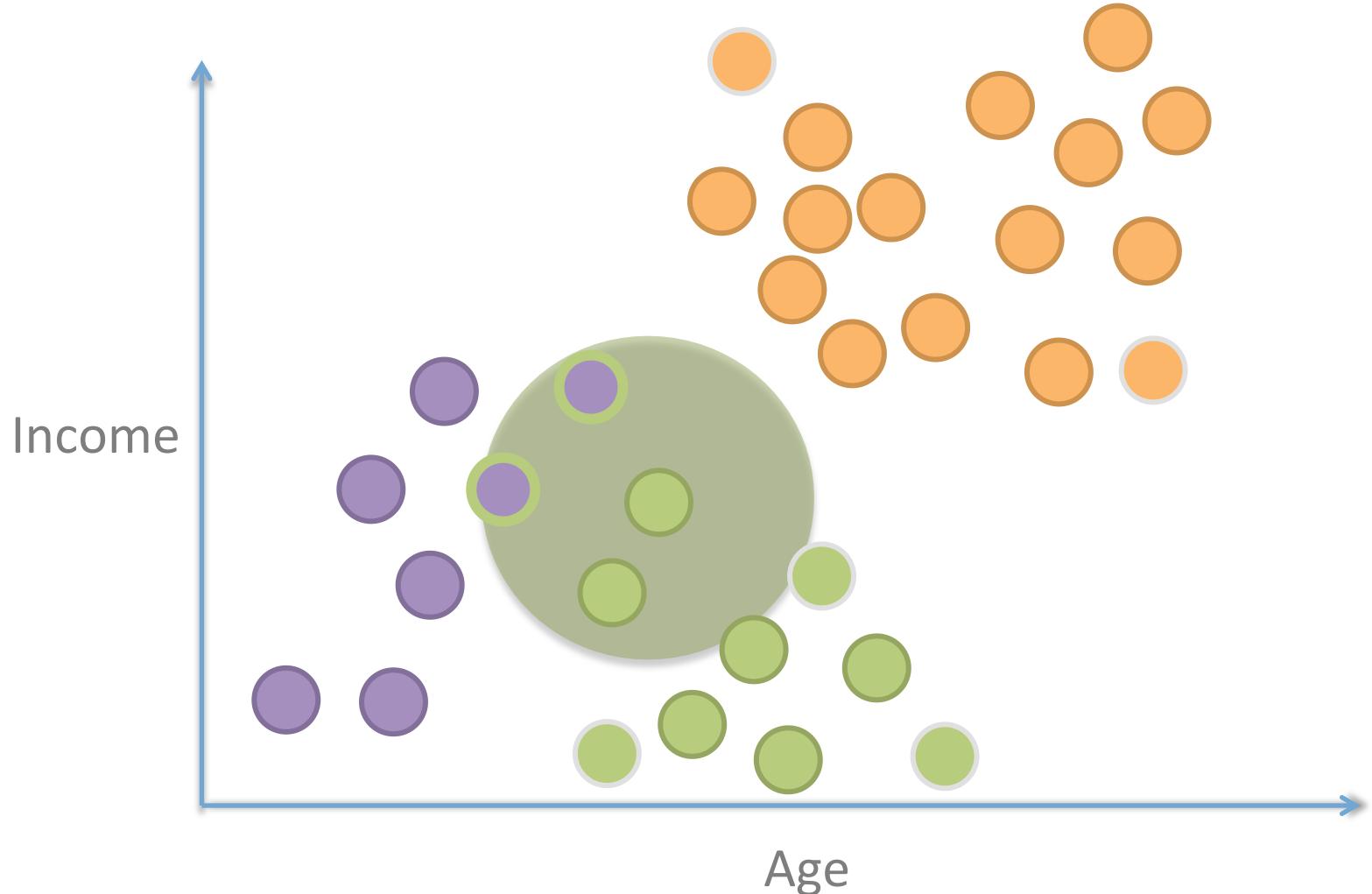
Keep adding neighbors within epsilon as chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

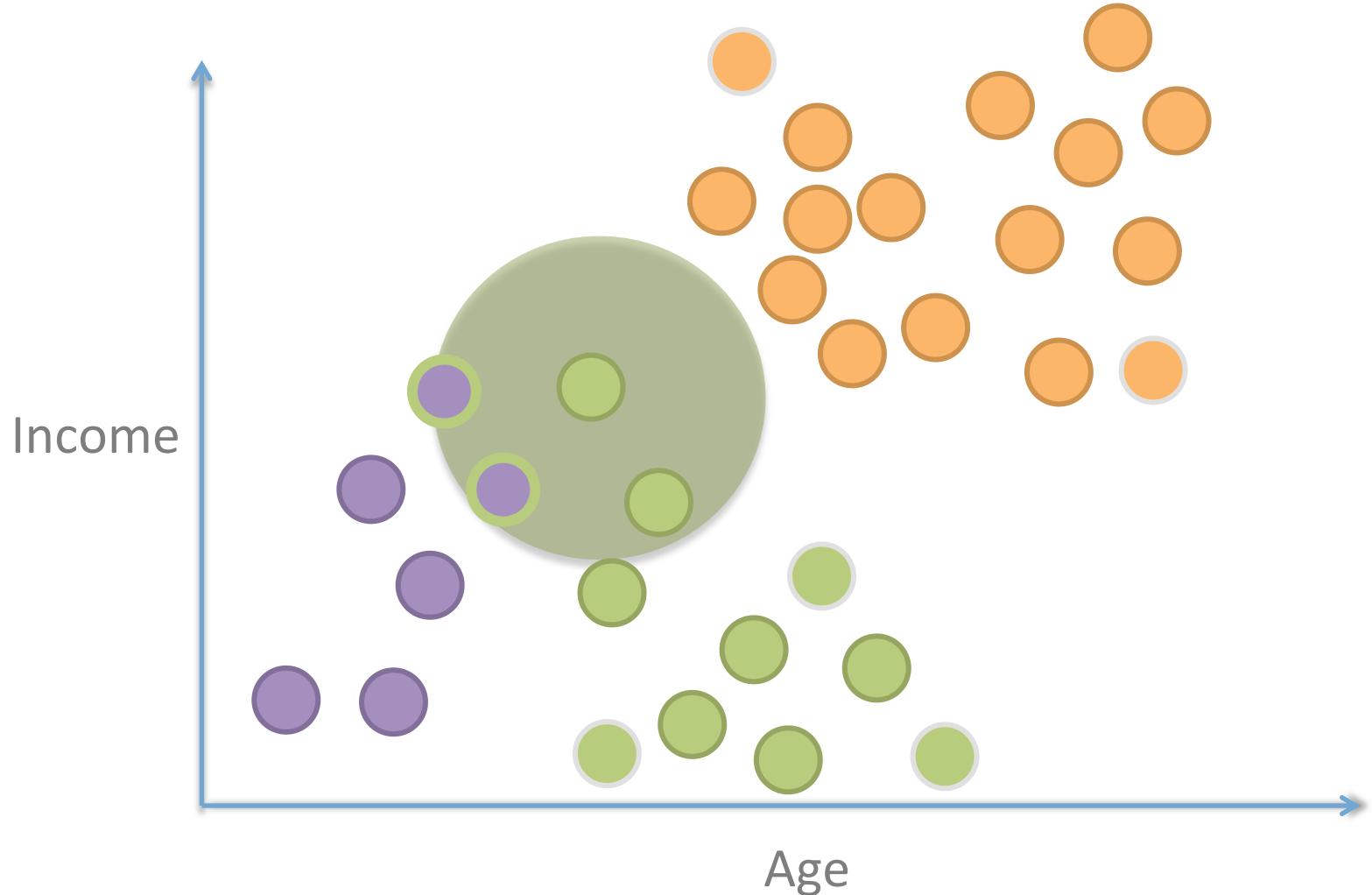
Keep adding neighbors within epsilon as chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

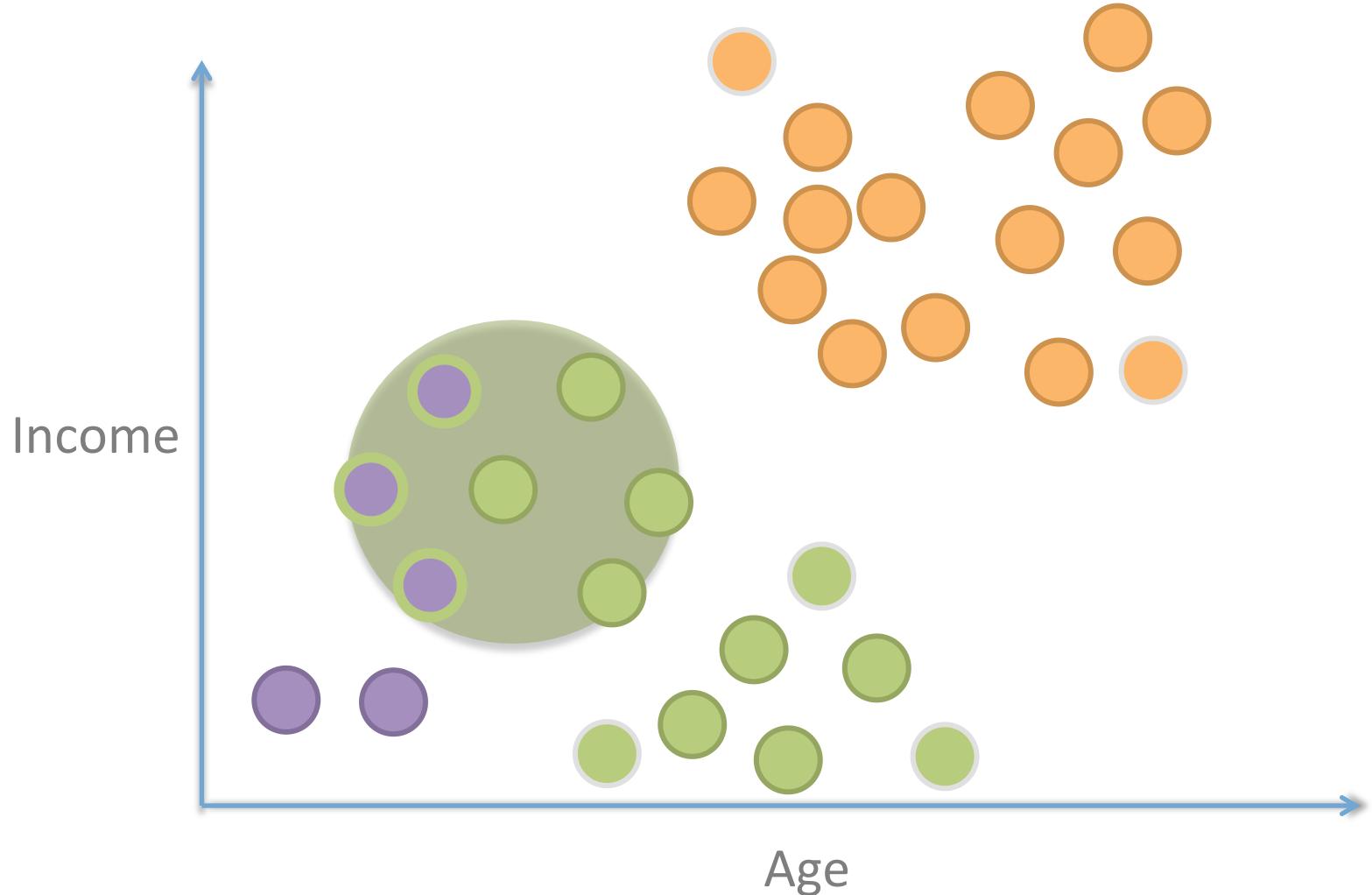
Keep adding neighbors within epsilon as chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

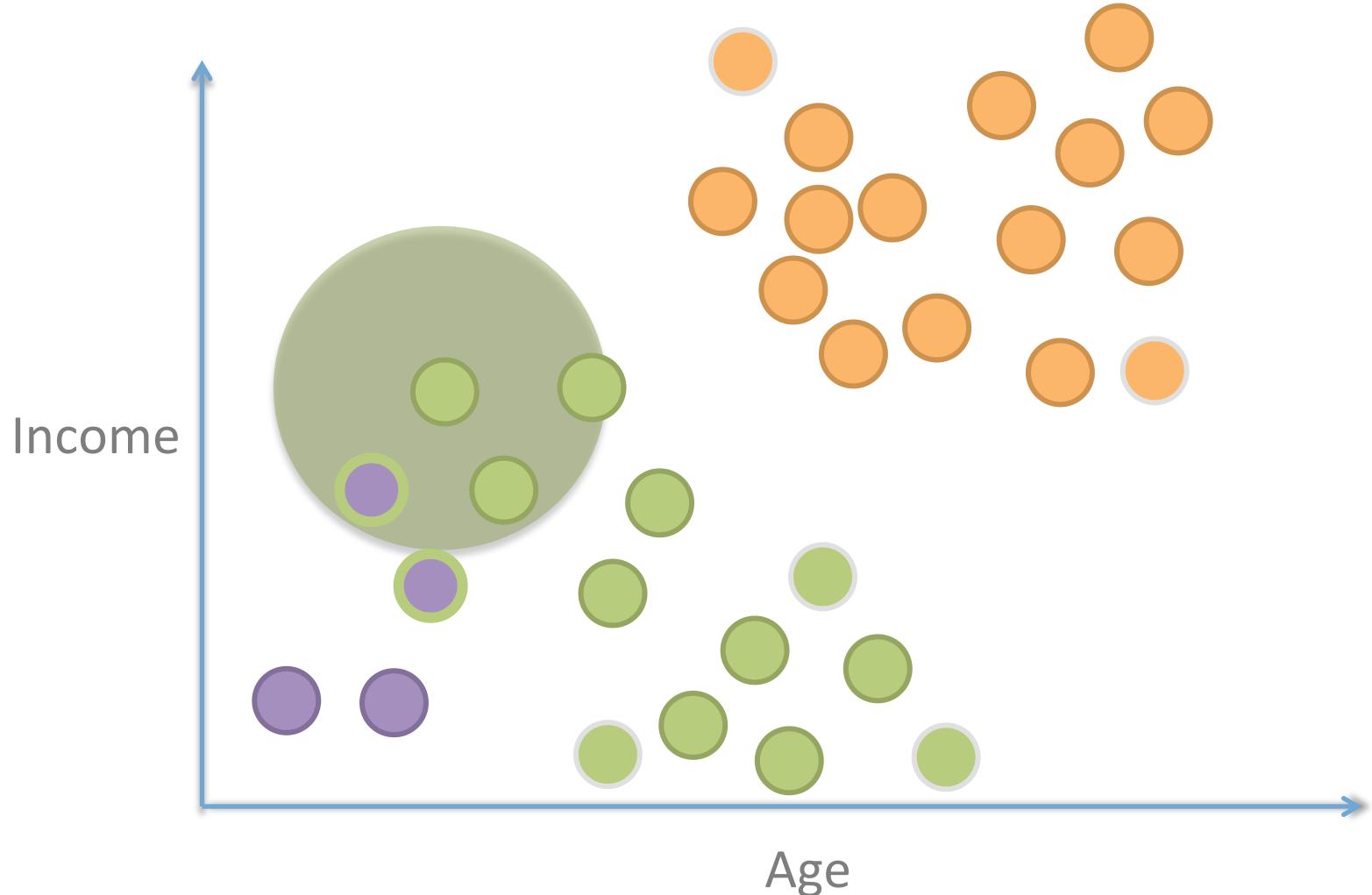
Keep adding neighbors within epsilon as chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

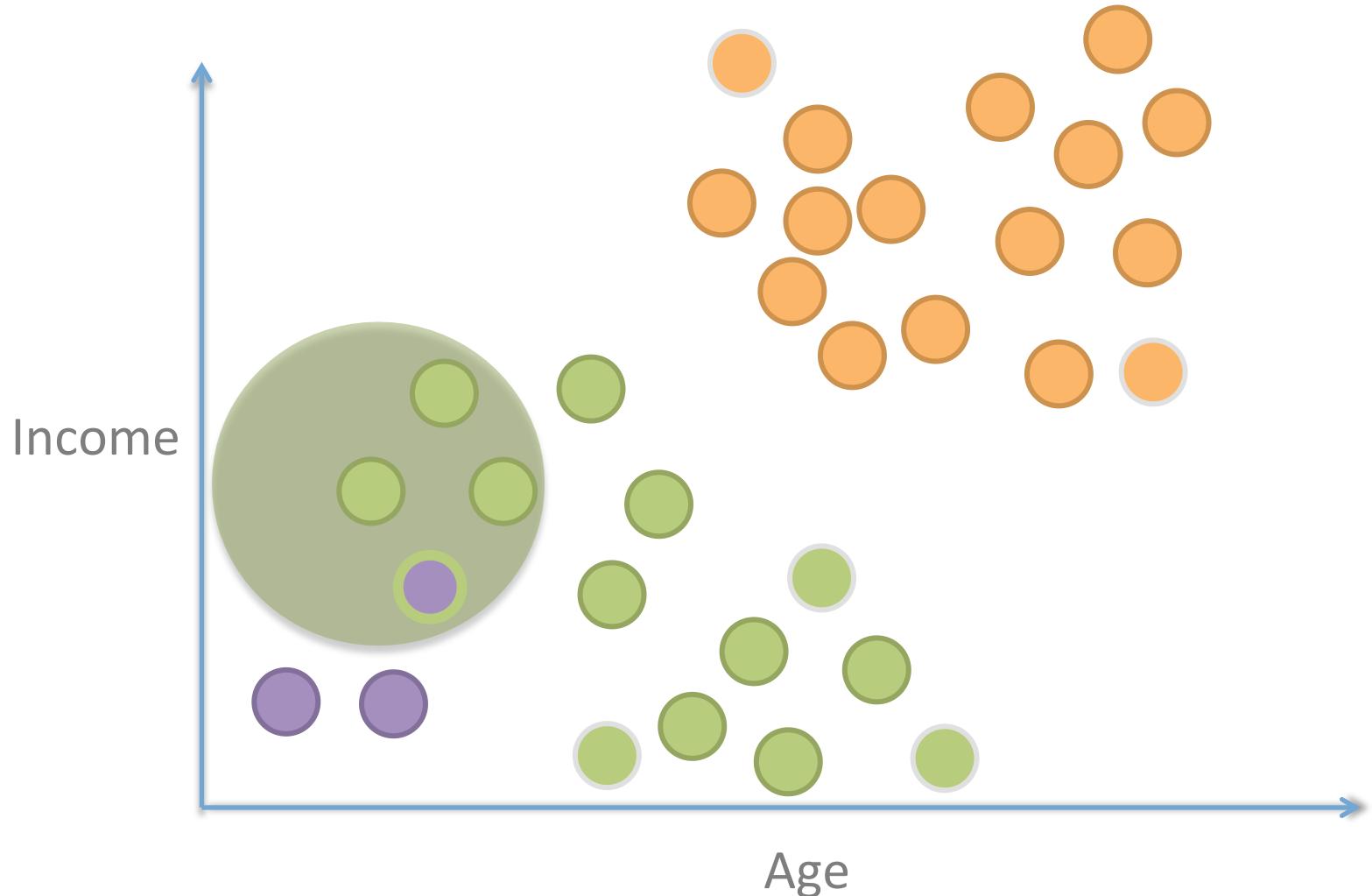
Keep adding neighbors within epsilon as chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

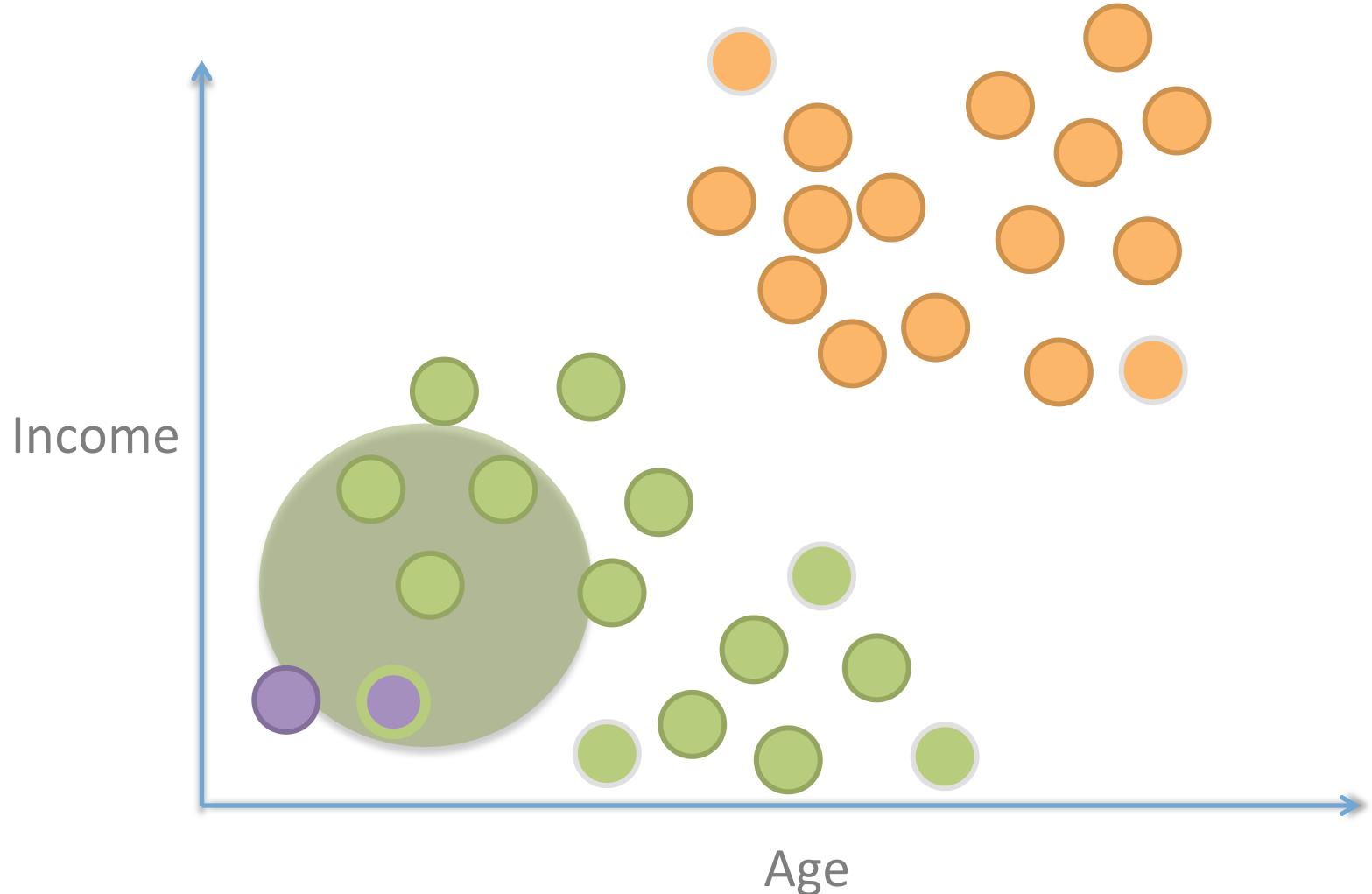
Keep adding neighbors within epsilon as chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

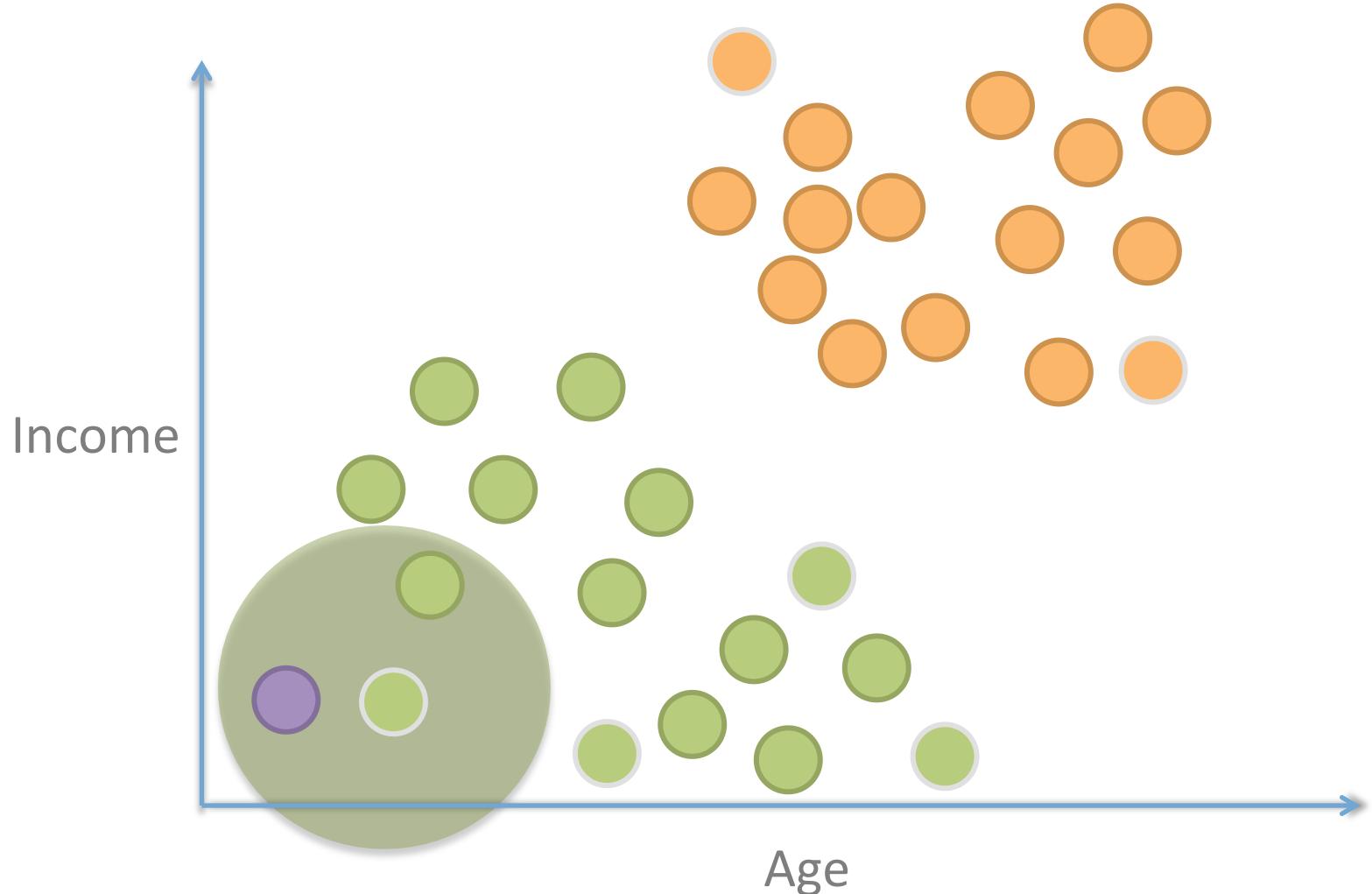
Keep adding neighbors within epsilon as chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

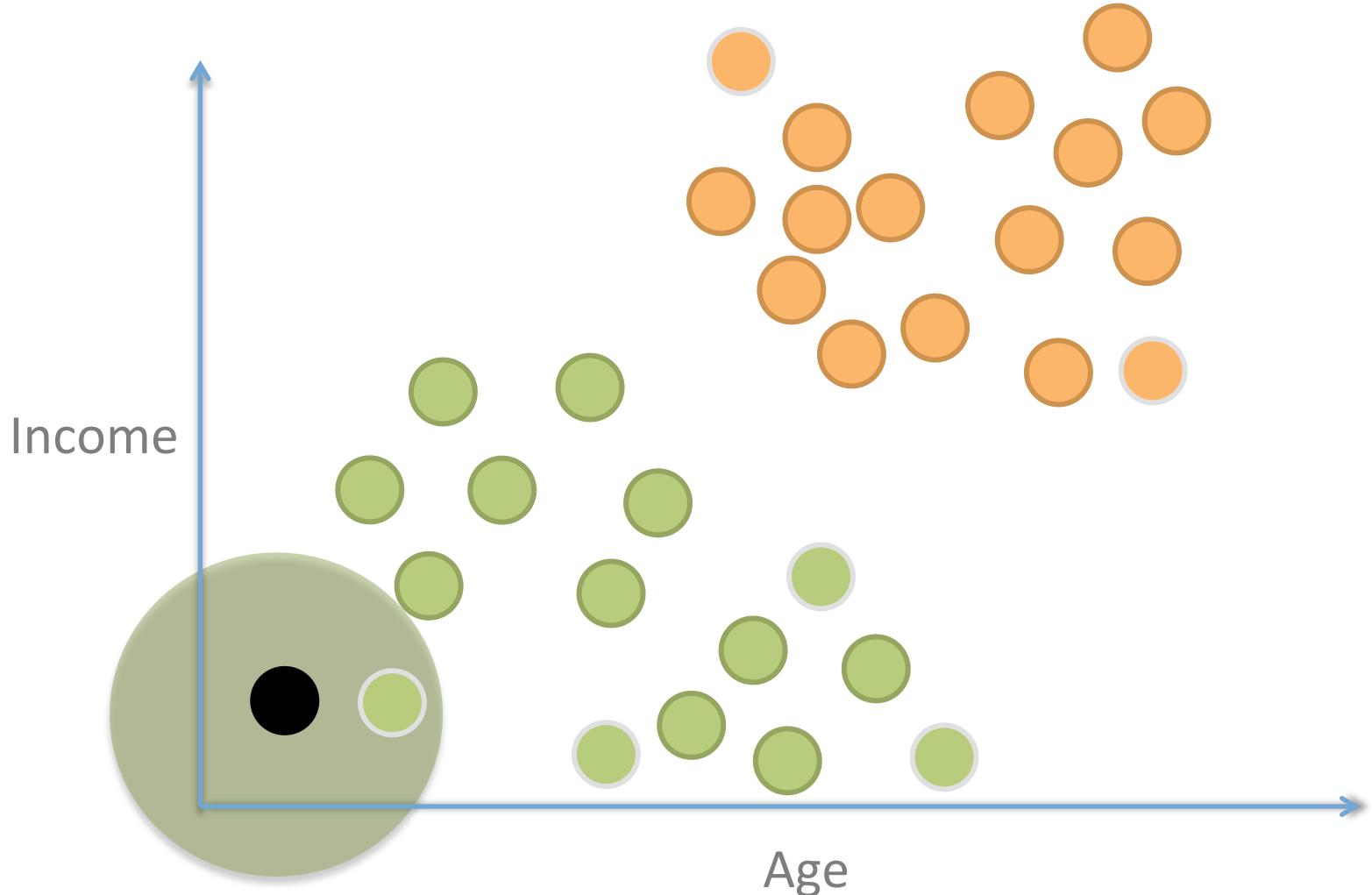
Keep adding neighbors within epsilon as chain reaction



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

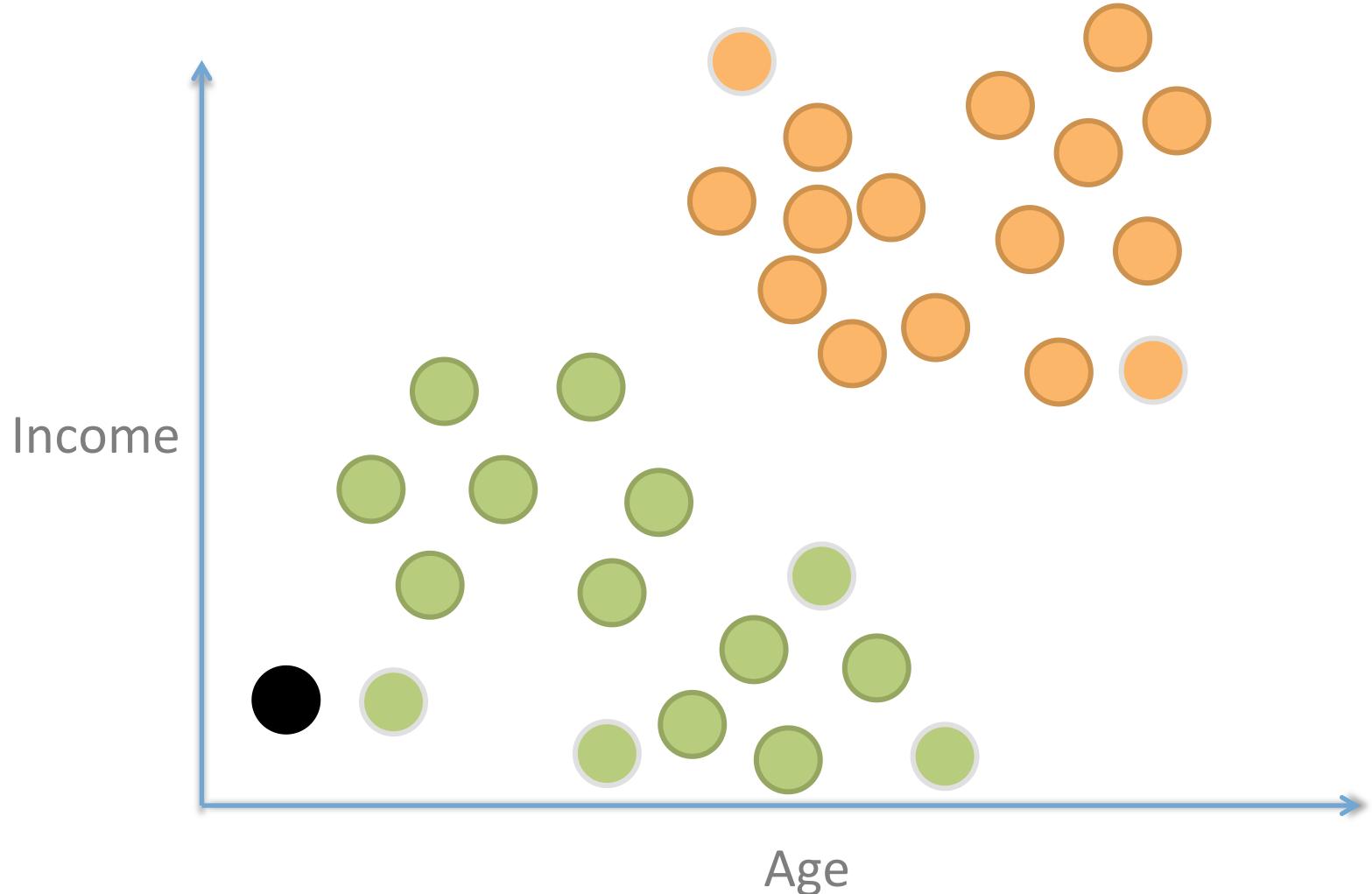
If there is a point that does not have n_{clu} neighbors and is not reached from a core point: It is a **noise** point.



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

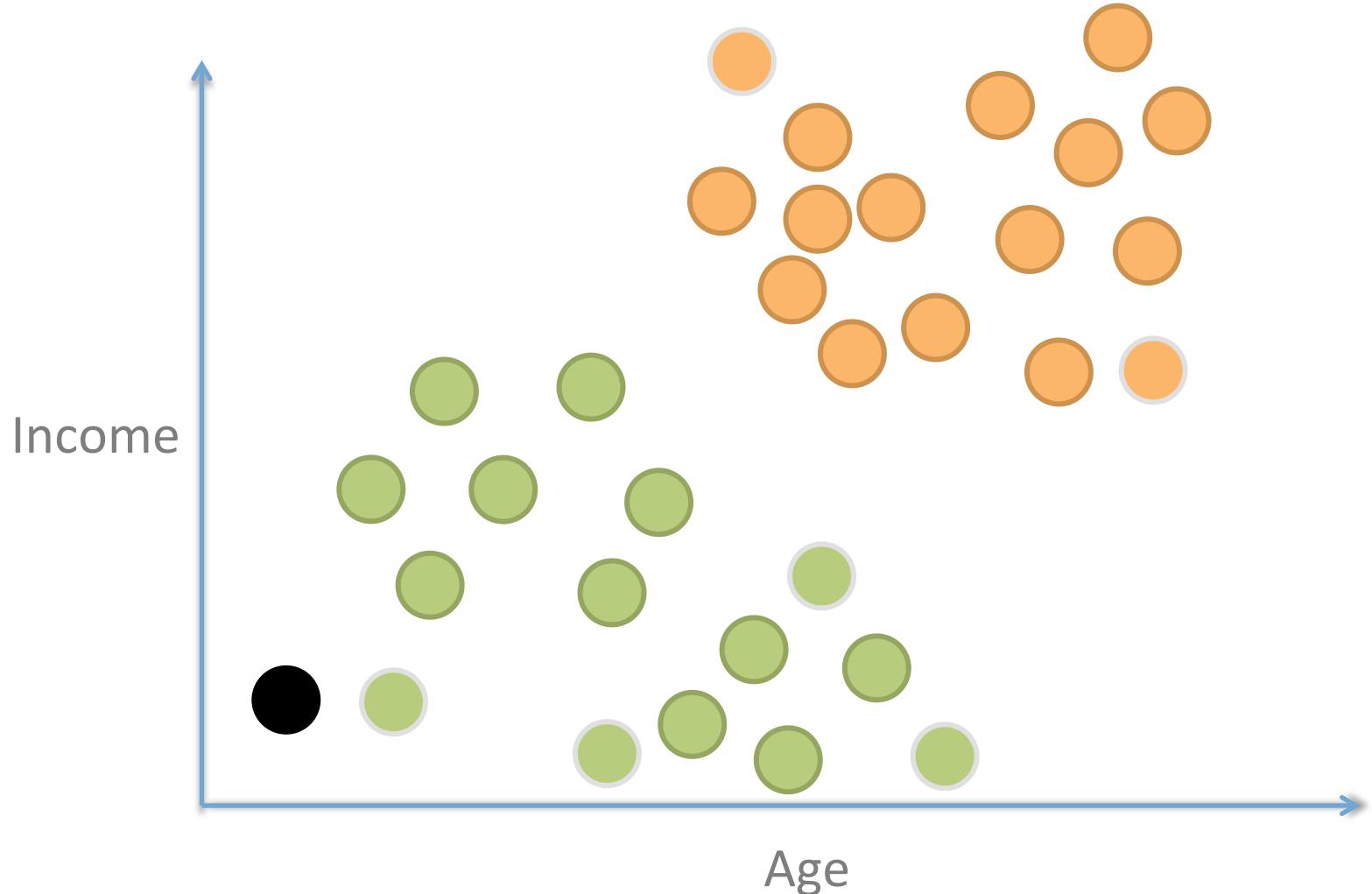
Ta--daa! Notice the core points, density-reachable points (at the borders) and the **noise** point.



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 3$

DBSCAN

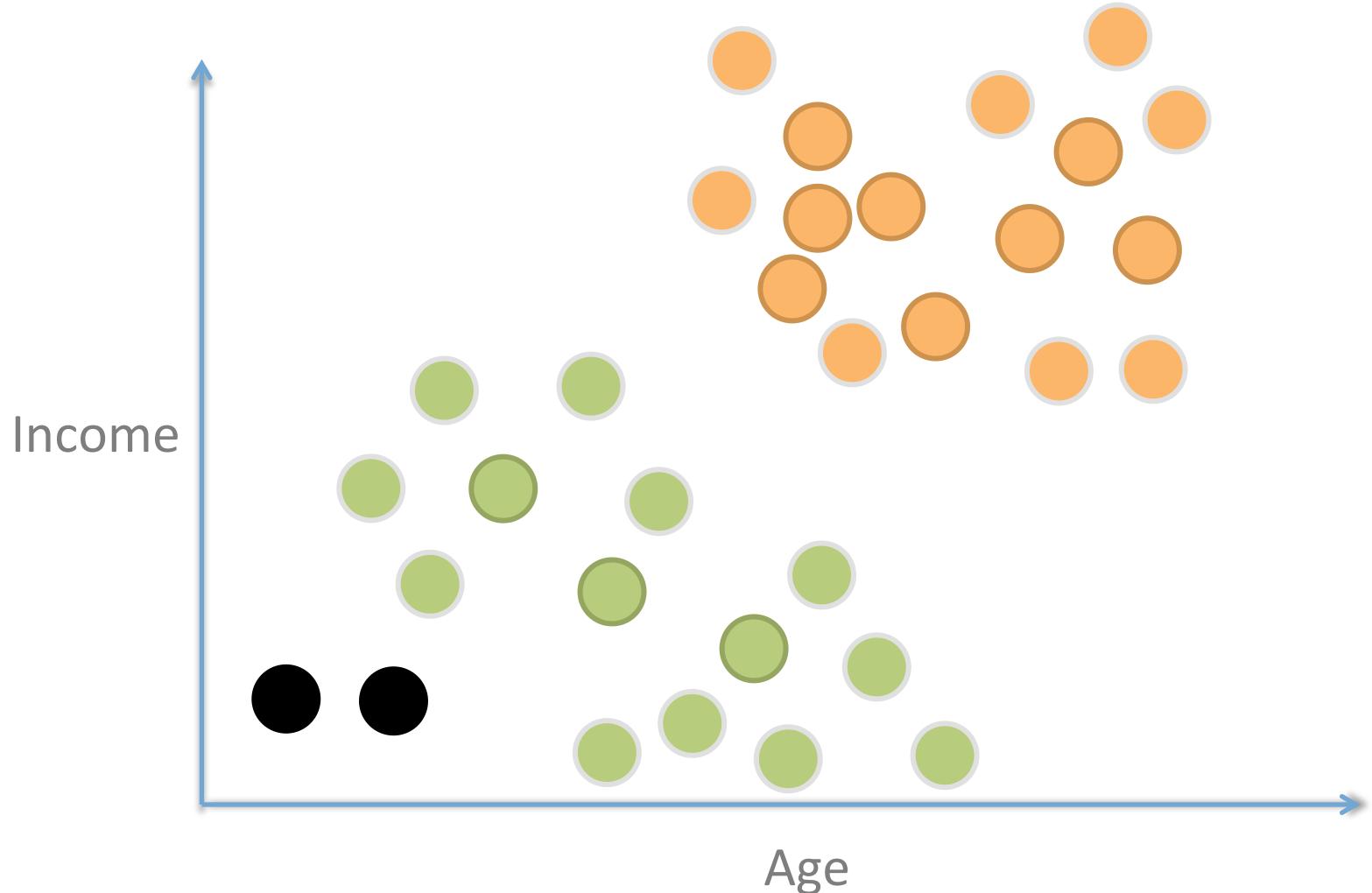
The effect of the parameters



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 5$

DBSCAN

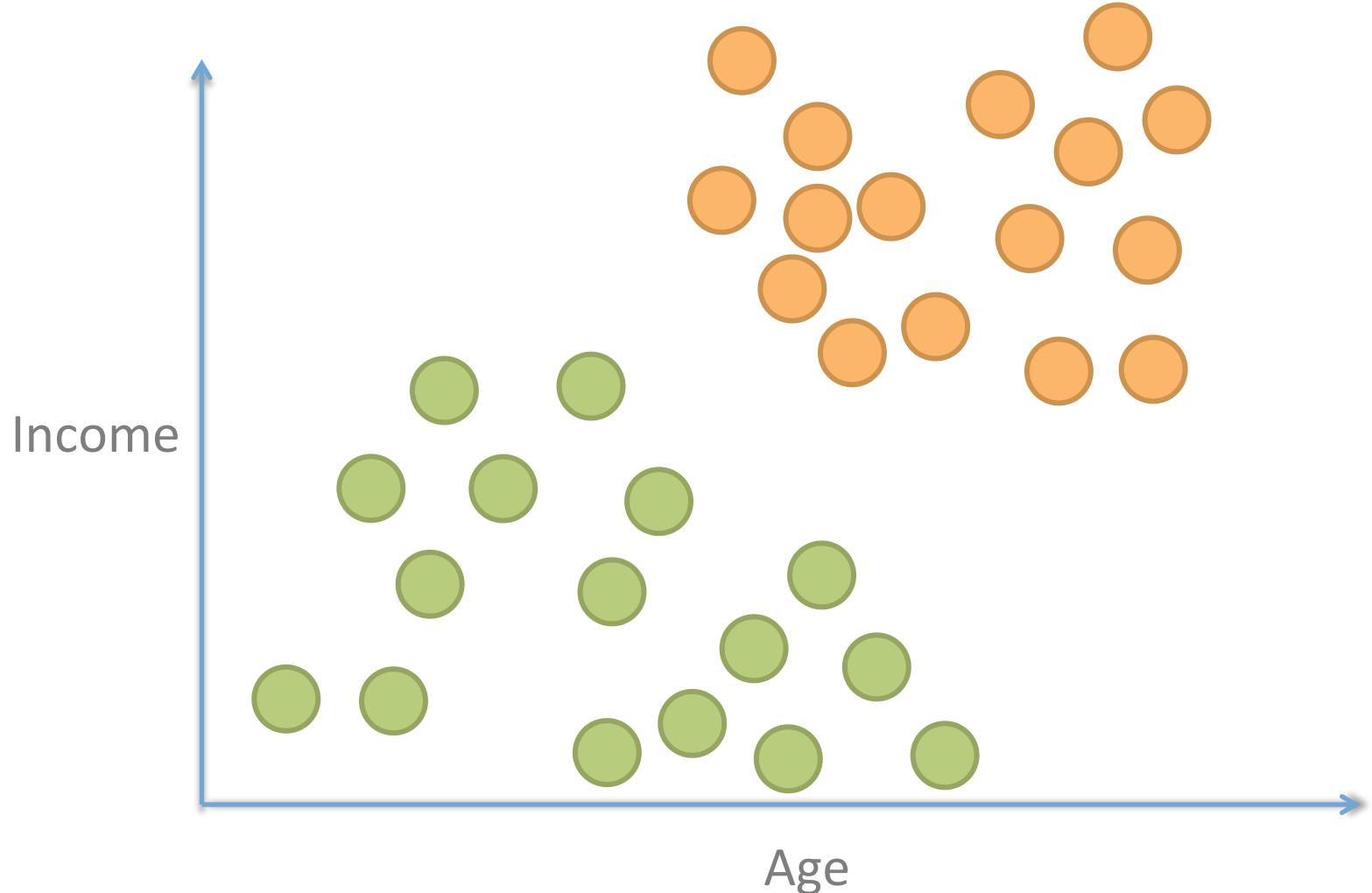
The effect of the parameters



$\text{epsilon} = 1.75$
 $n_{\text{clu}} = 1$

DBSCAN

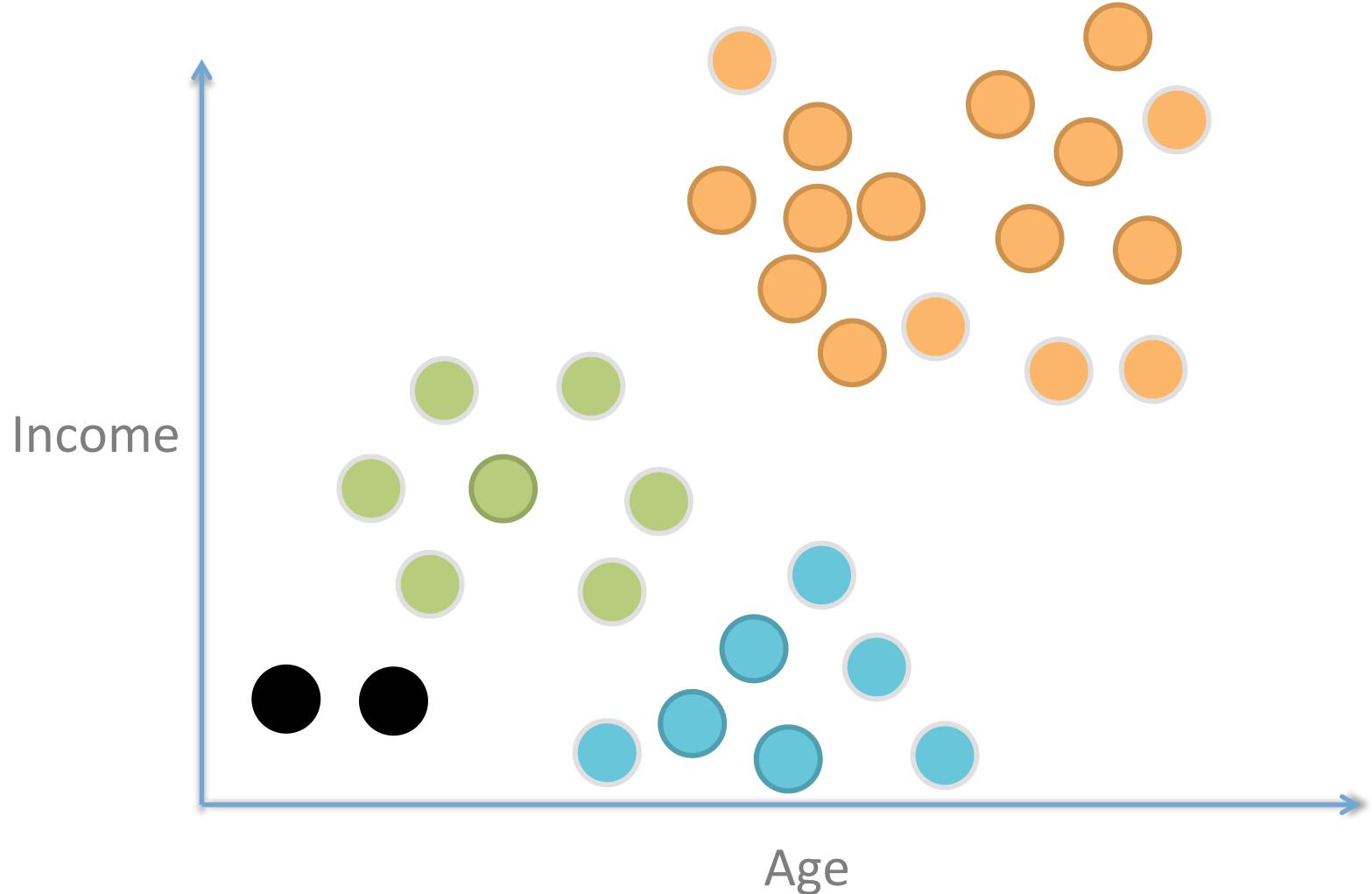
The effect of the parameters



epsilon = 1.25
n_clu = 3

DBSCAN

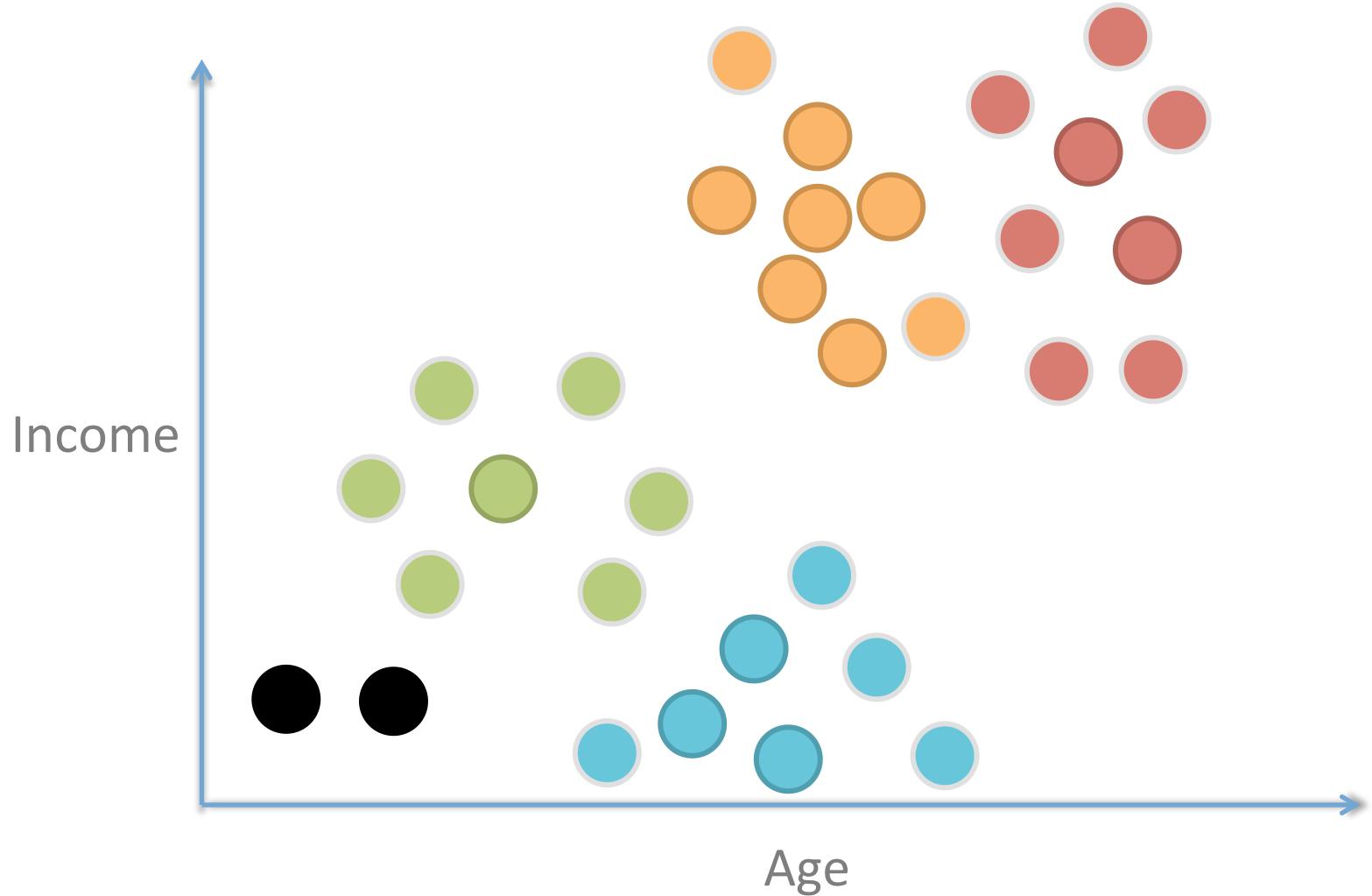
The effect of the parameters



epsilon = 1.00
n_clu = 3

DBSCAN

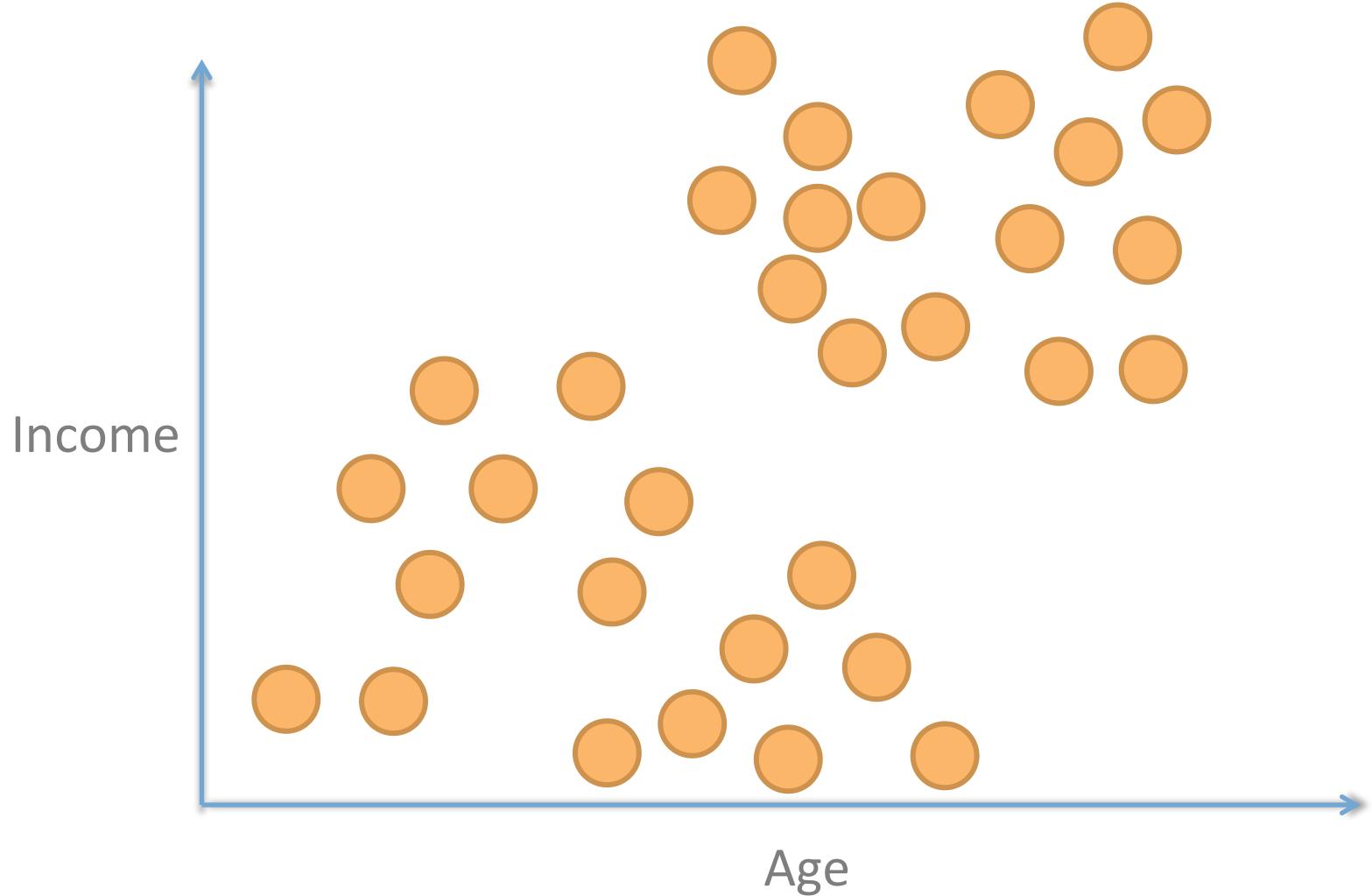
The effect of the parameters



epsilon = 3.50
n_clu = 3

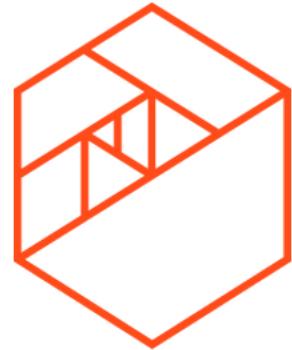
DBSCAN

The effect of the parameters

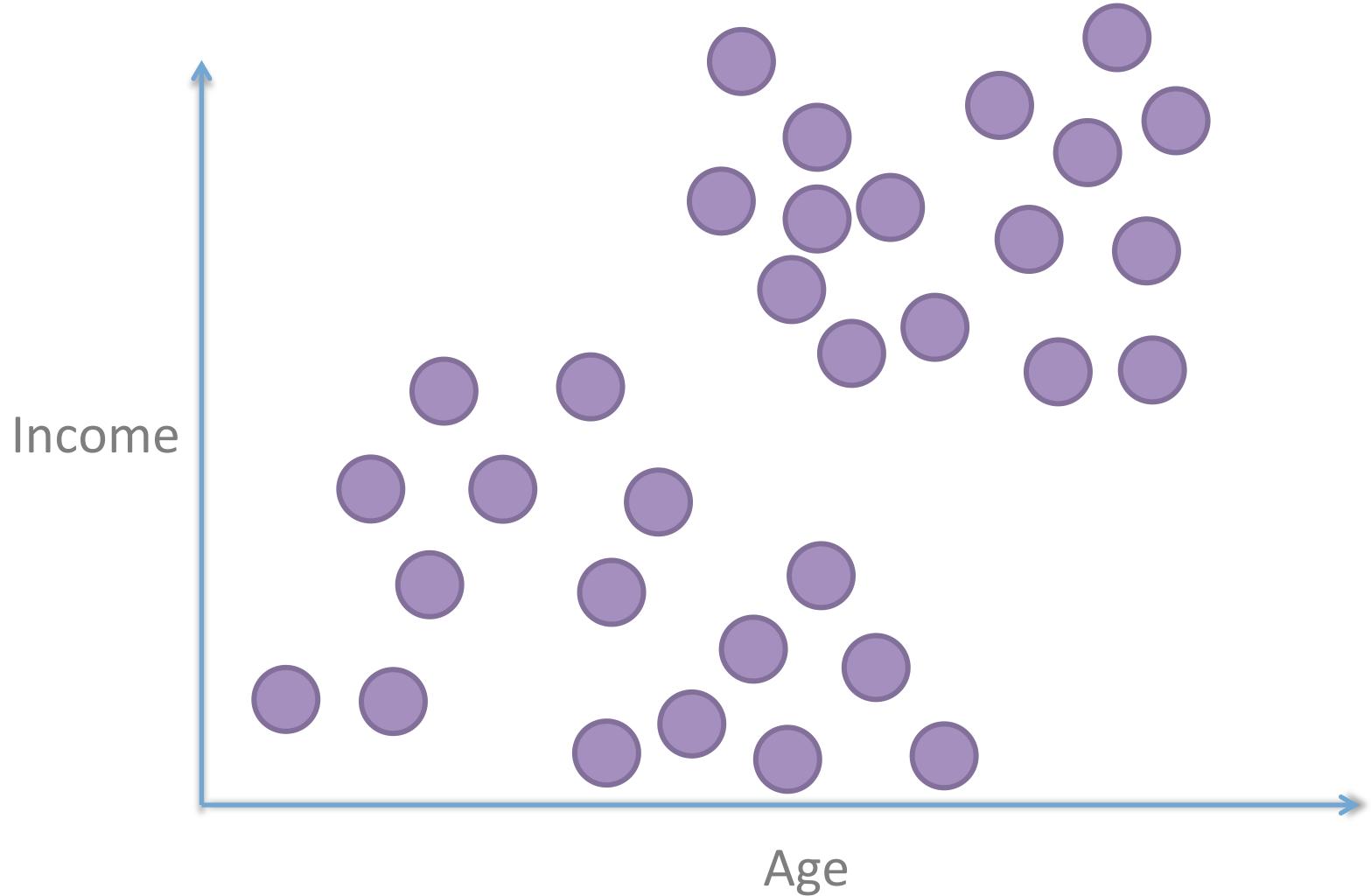


```
from sklearn.cluster import DBSCAN  
  
DBSCAN( eps = 0.5, min_samples = 5, metric = "euclidean" )
```

Mean Shift

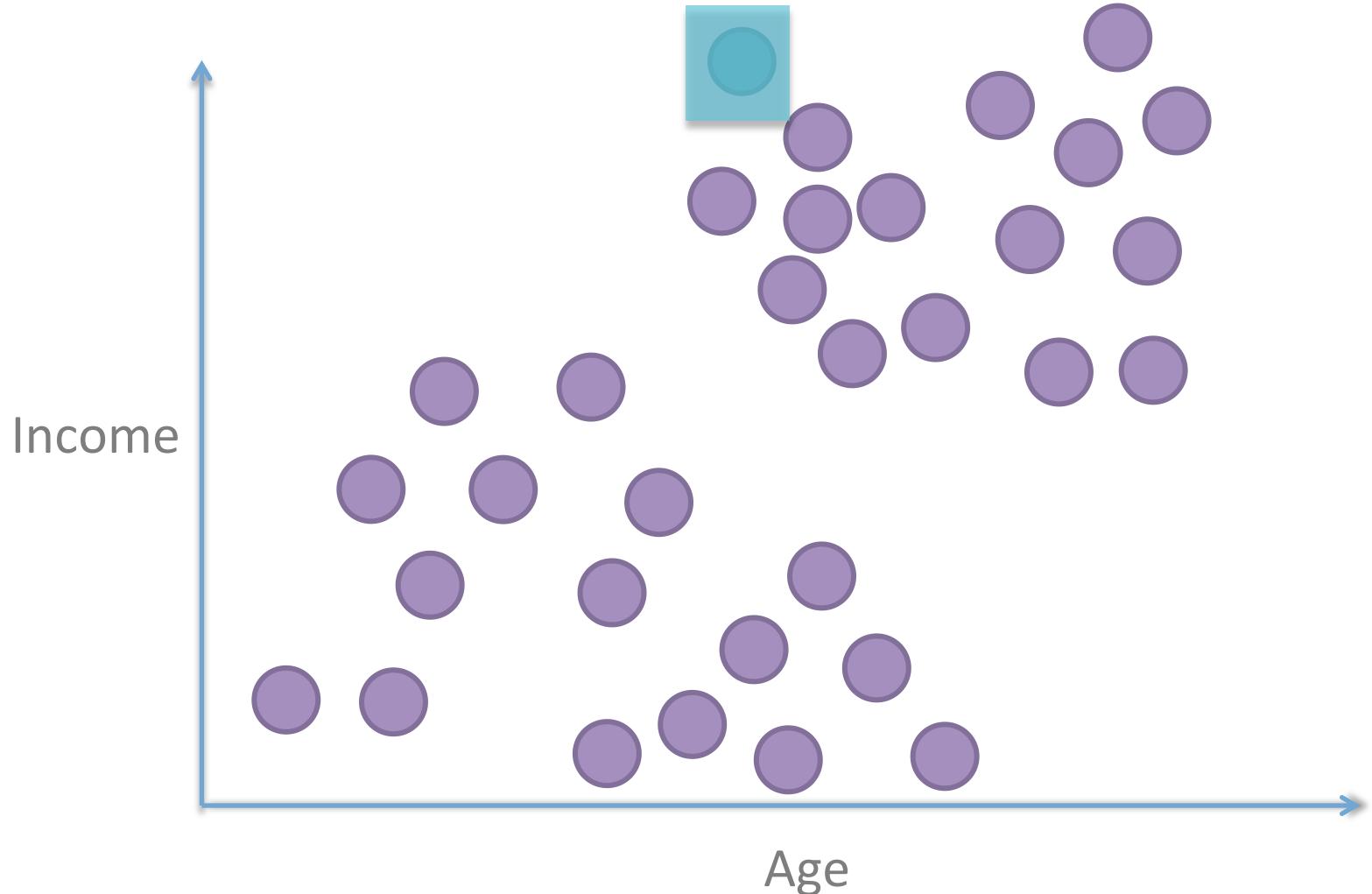


Mean Shift



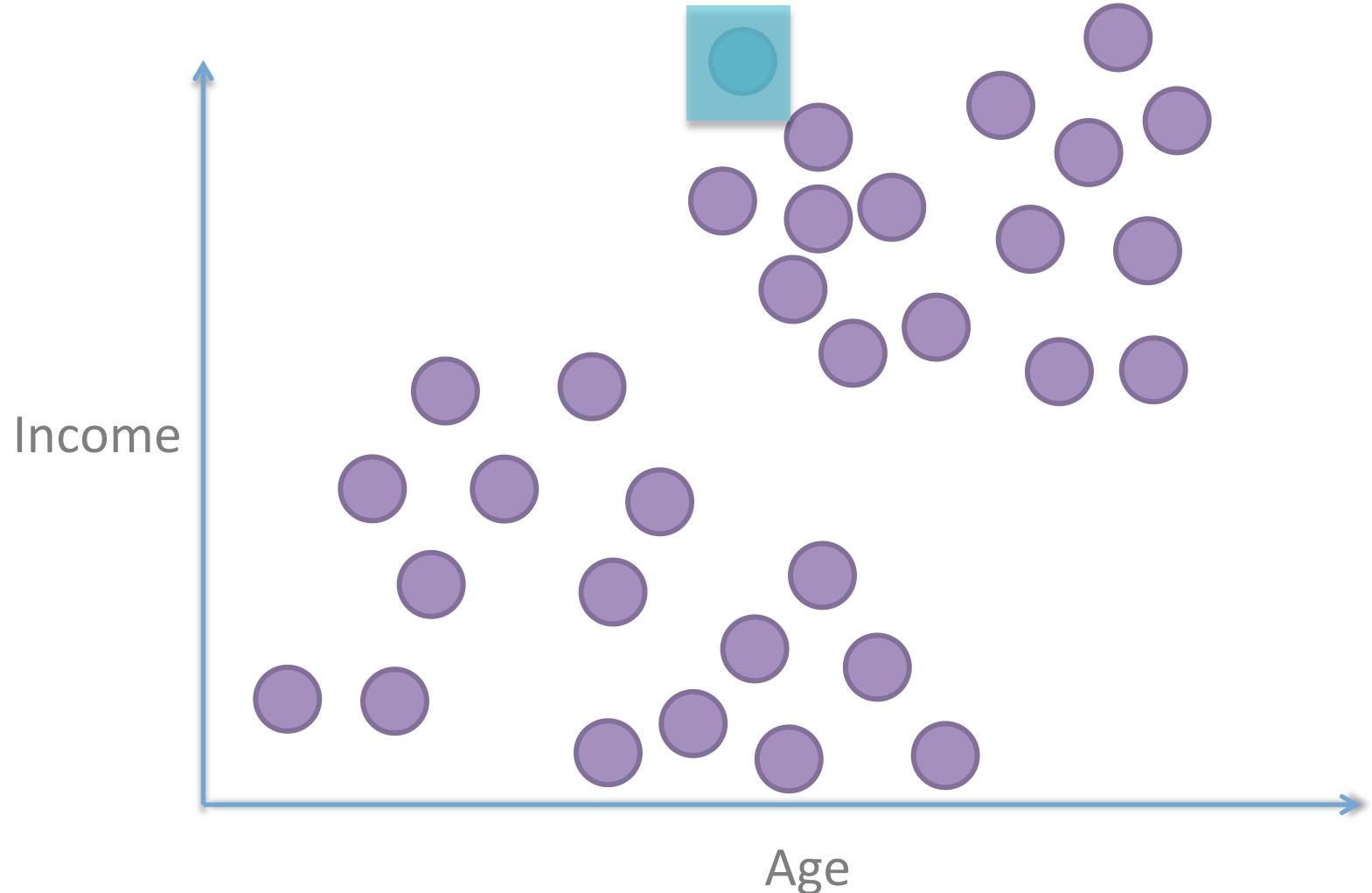
Mean Shift

Start with a centroid at a point



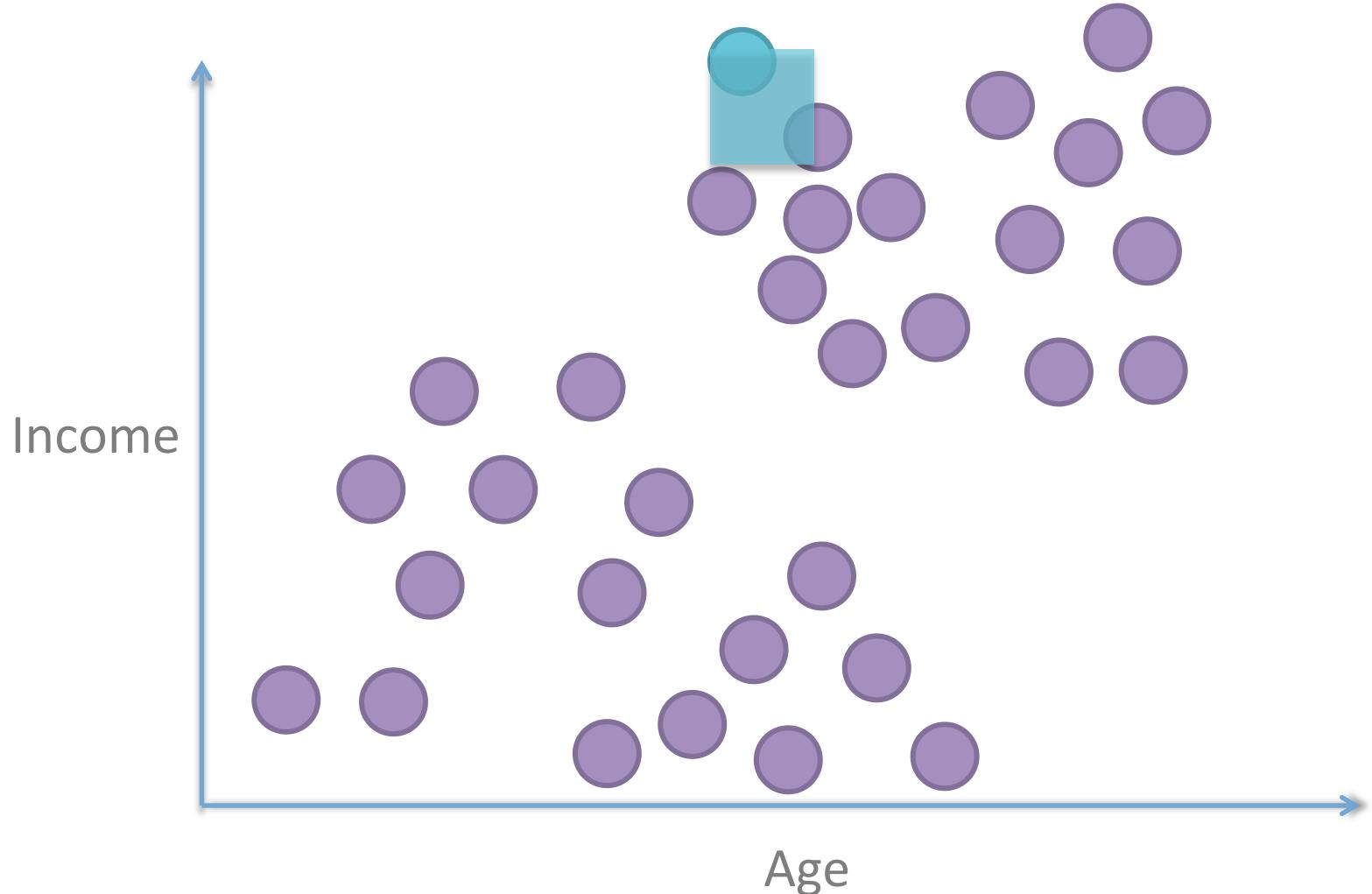
Mean Shift

Sample local density, follow gradient towards denser direction



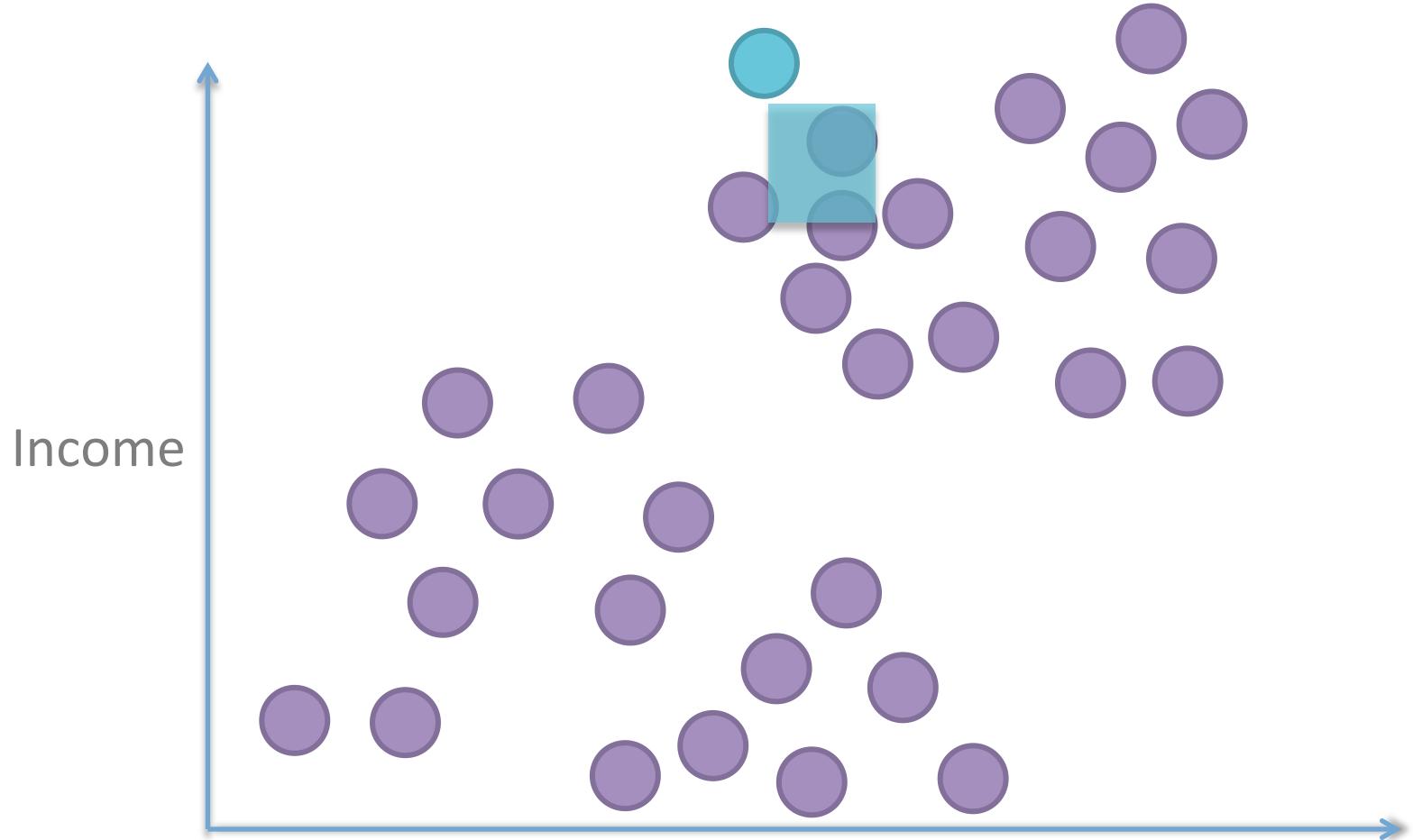
Mean Shift

Sample local density, follow gradient towards denser direction



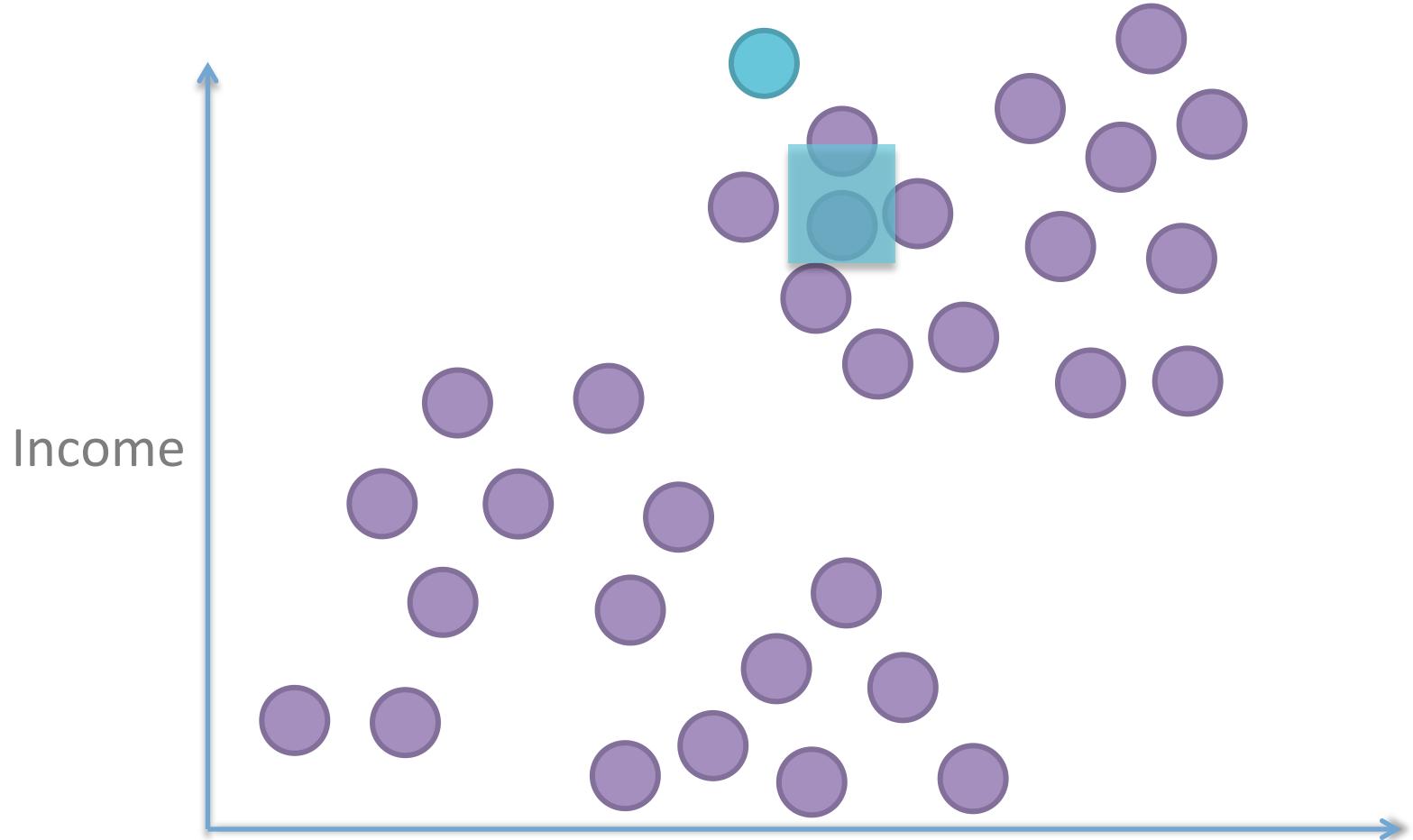
Mean Shift

Sample local density, follow gradient towards denser direction



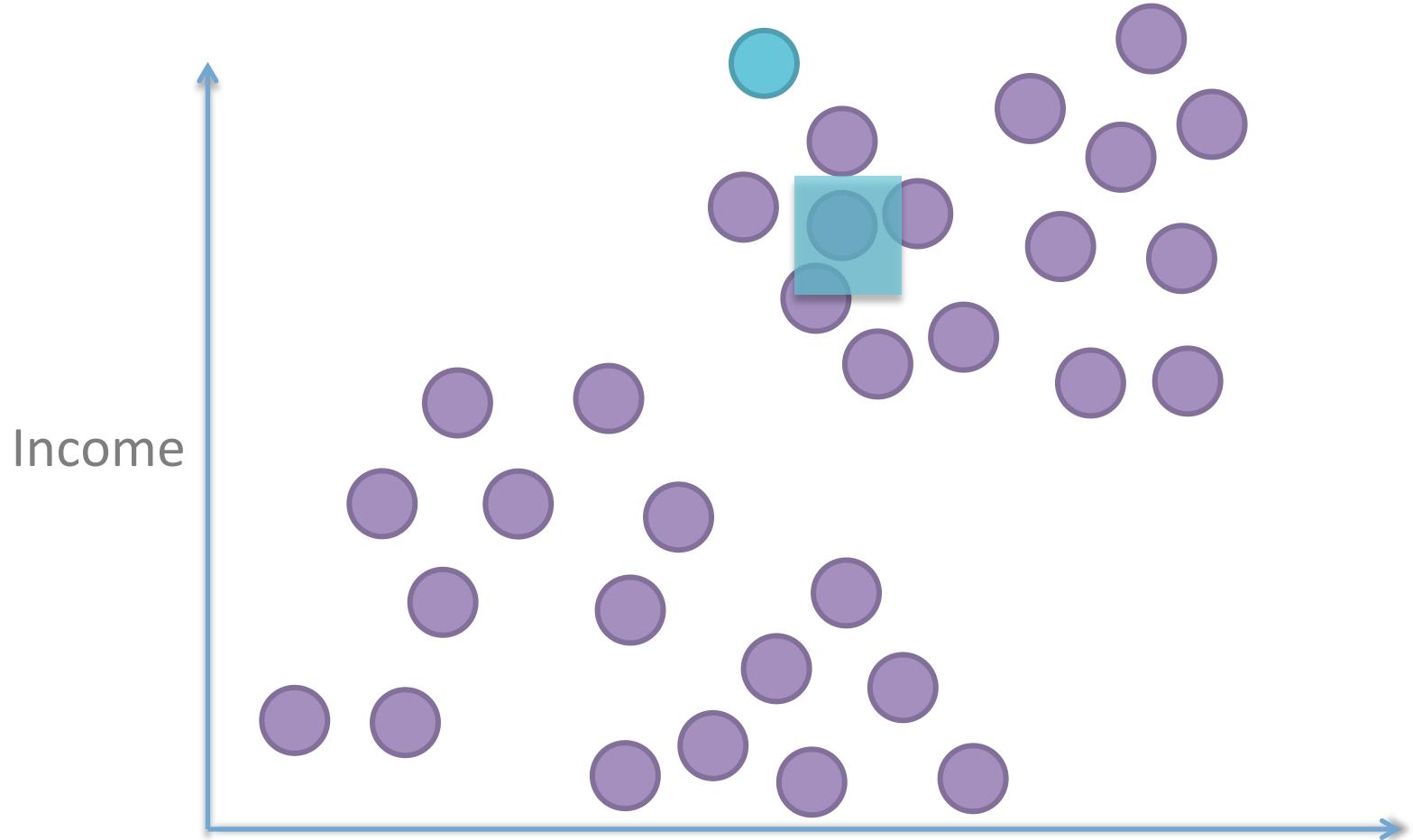
Mean Shift

Sample local density, follow gradient towards denser direction



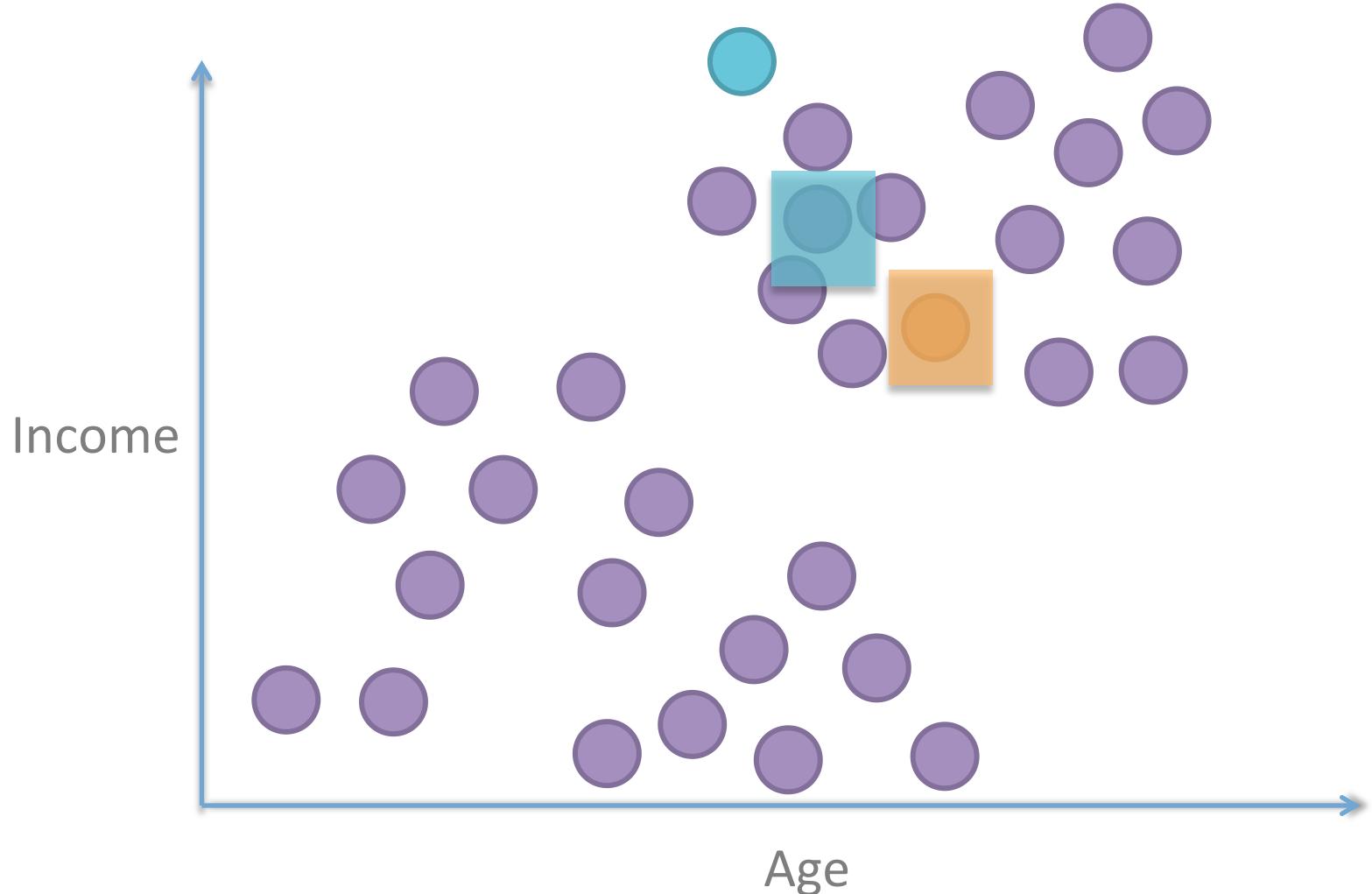
Mean Shift

Found local density maximum! Stop.



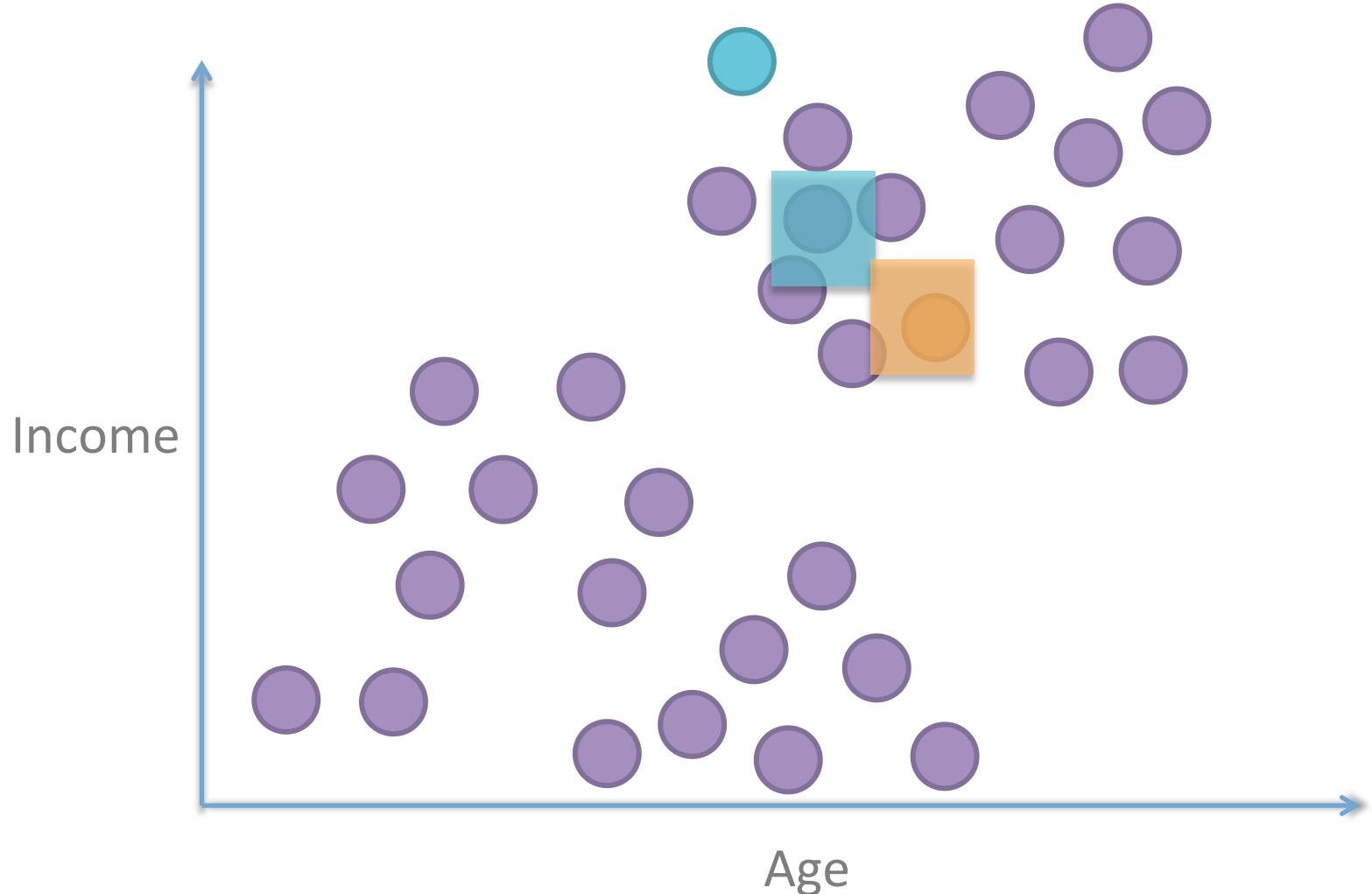
Mean Shift

Start at another point.



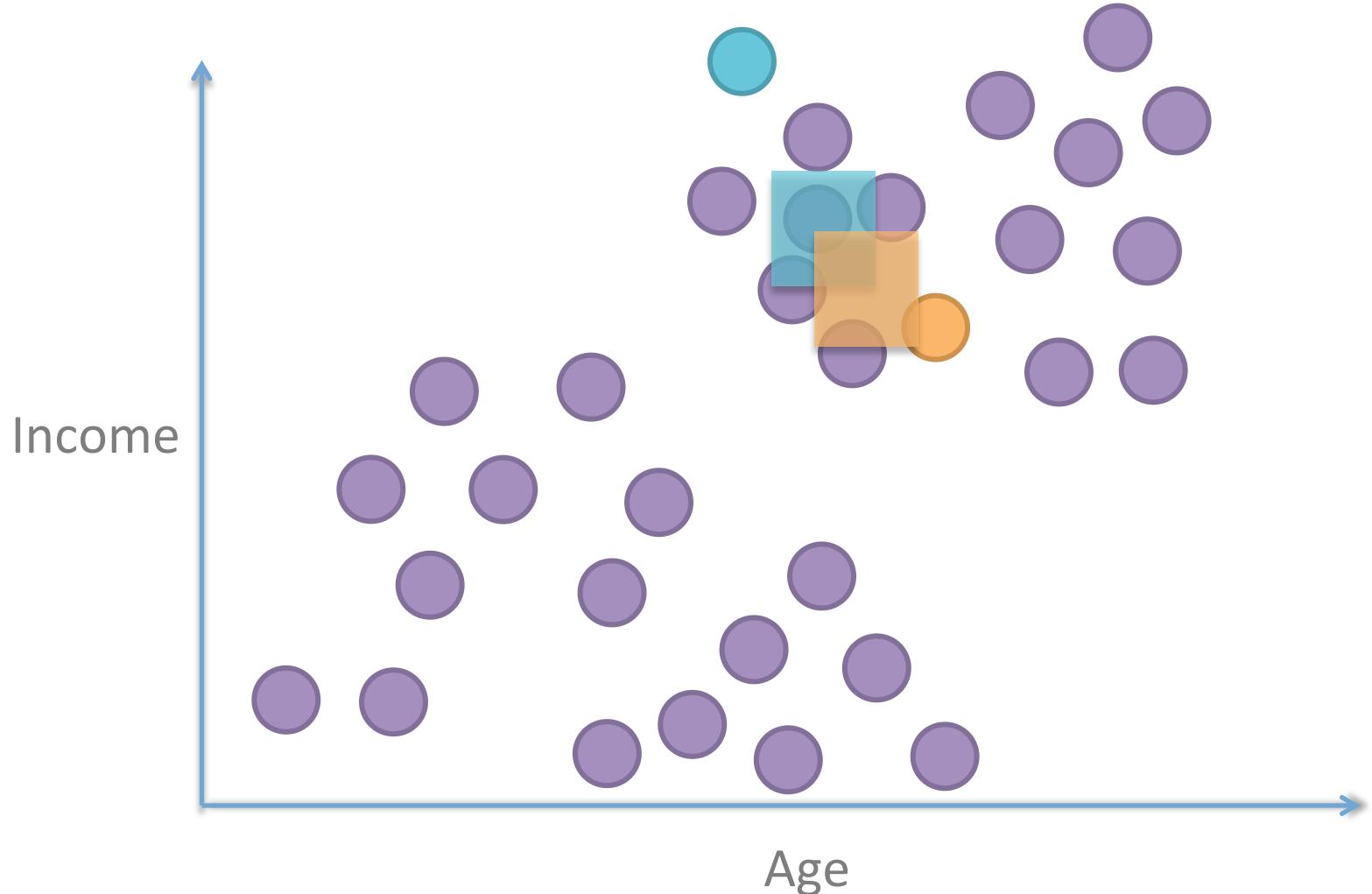
Mean Shift

Sample local density, follow gradient towards denser direction



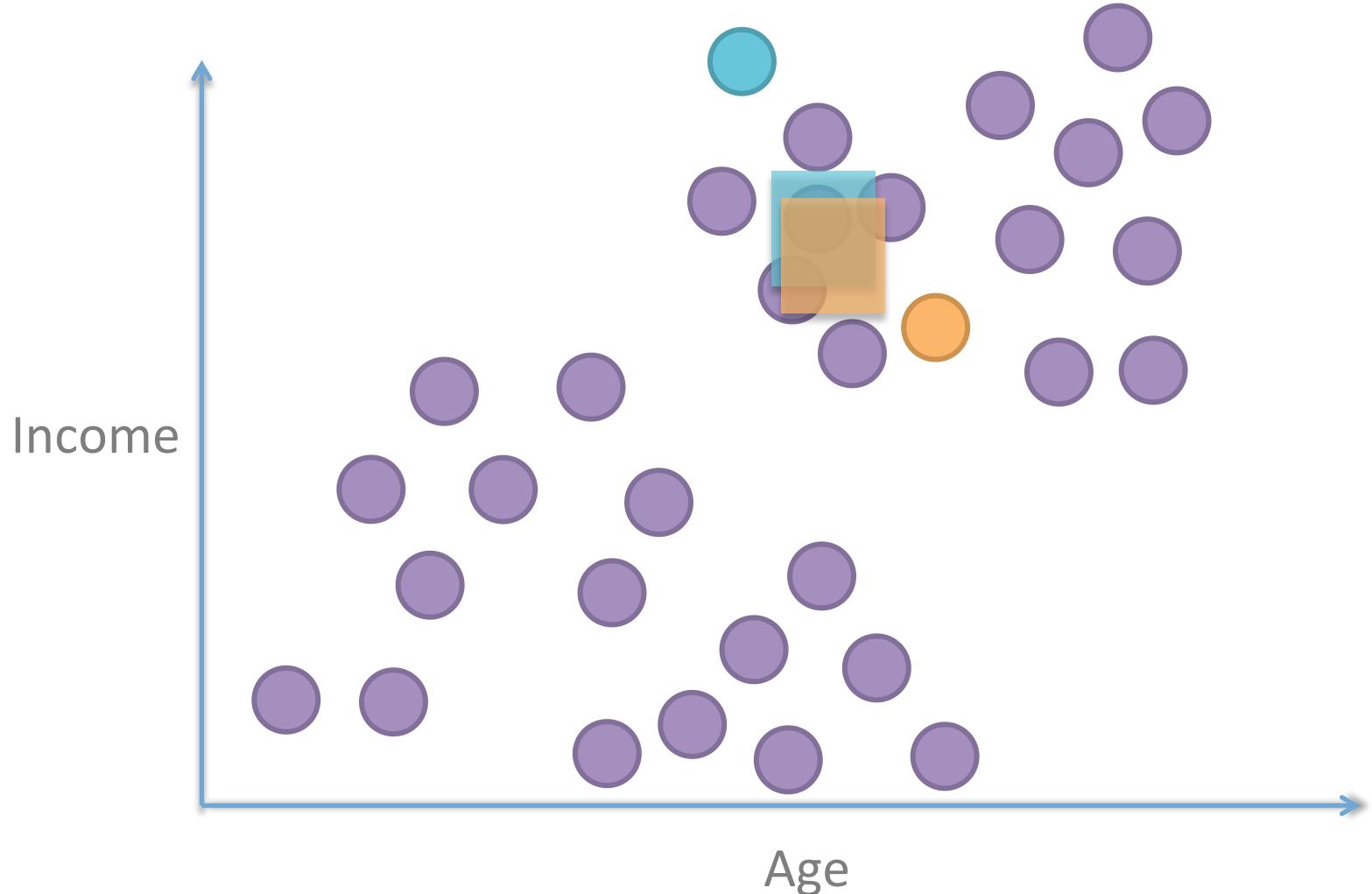
Mean Shift

Sample local density, follow gradient towards denser direction



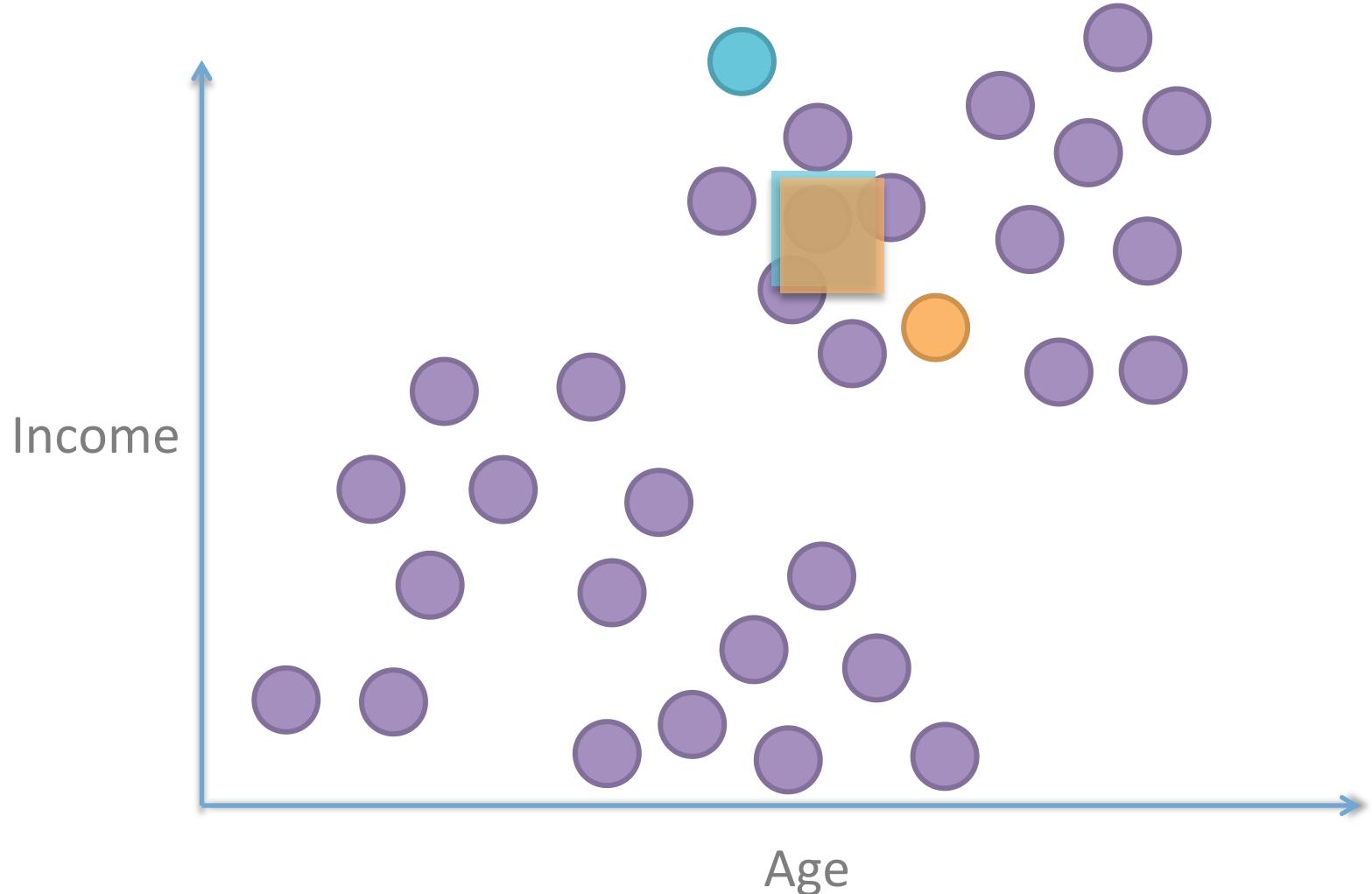
Mean Shift

Sample local density, follow gradient towards denser direction



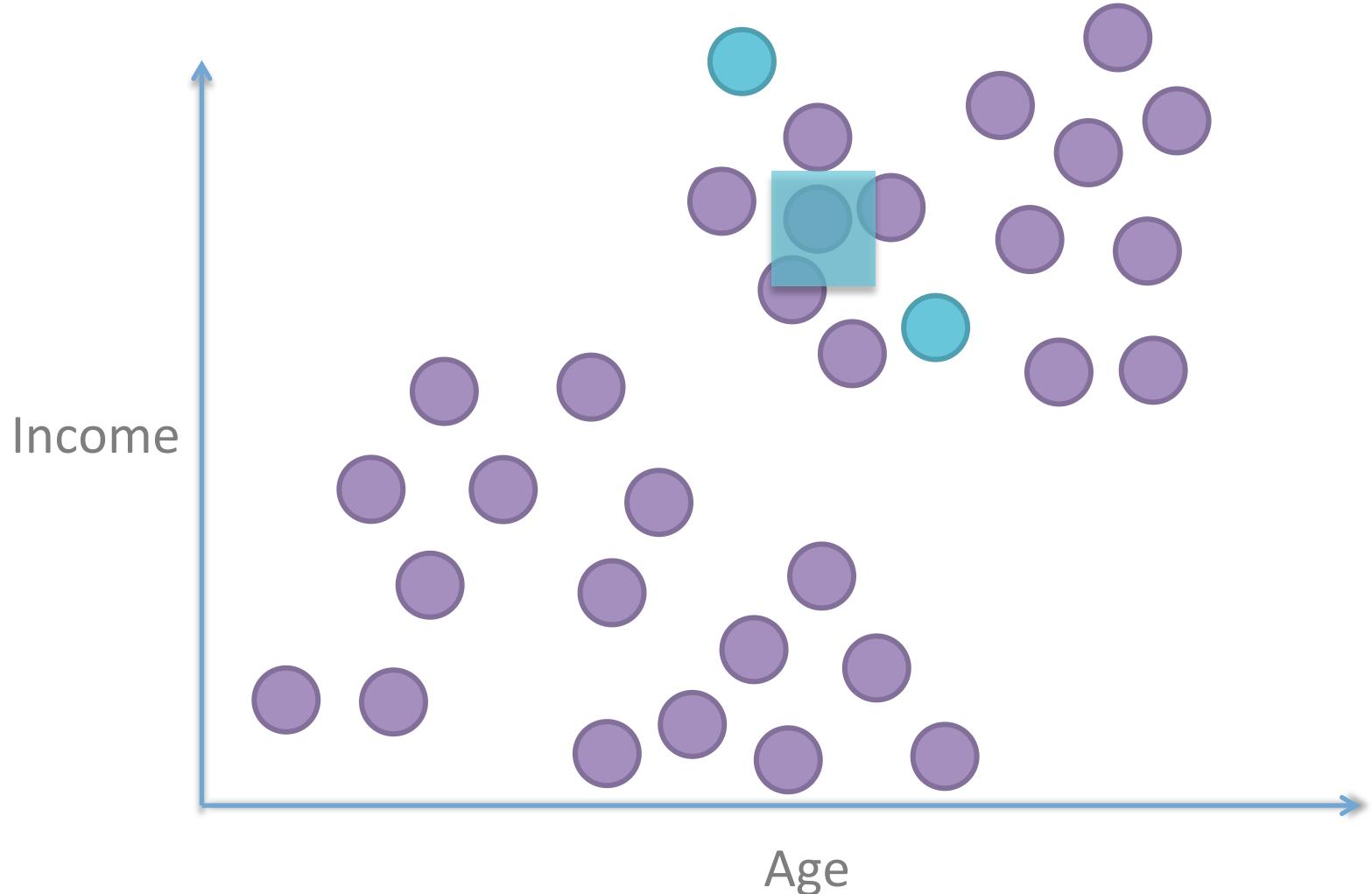
Mean Shift

Sample local density, follow gradient towards denser direction



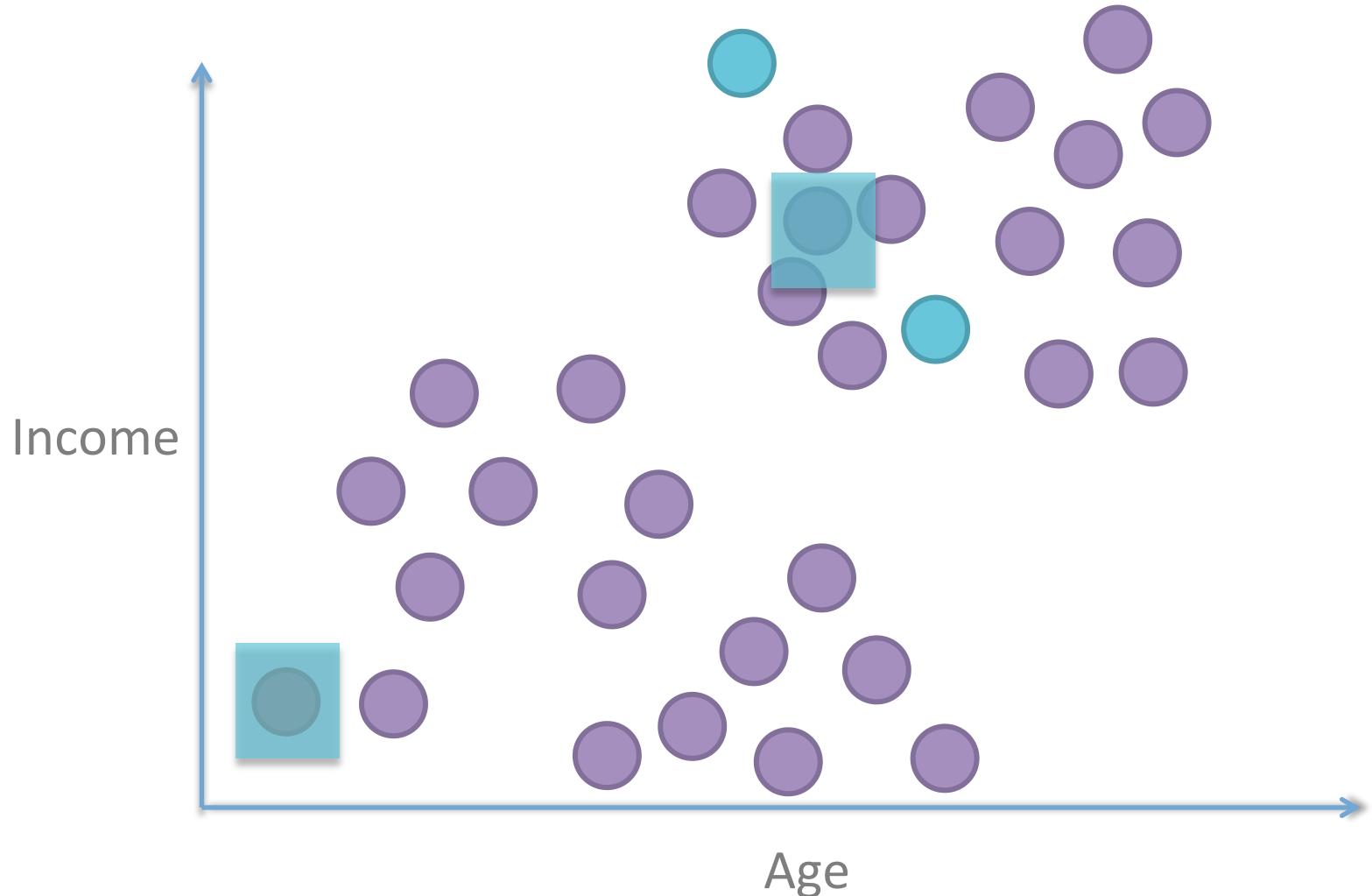
Mean Shift

Found the same (close) local maximum, same cluster



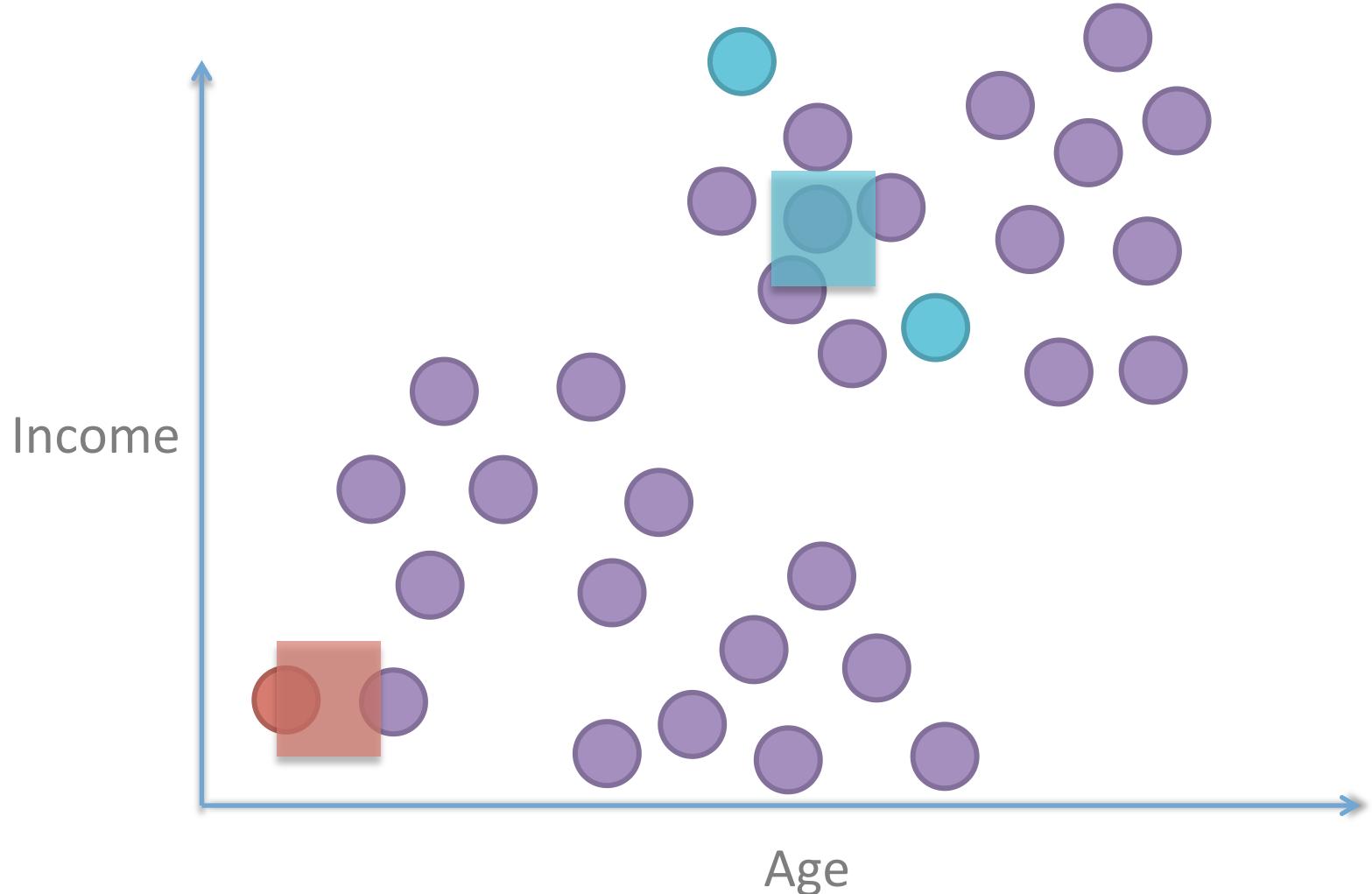
Mean Shift

Start at another point



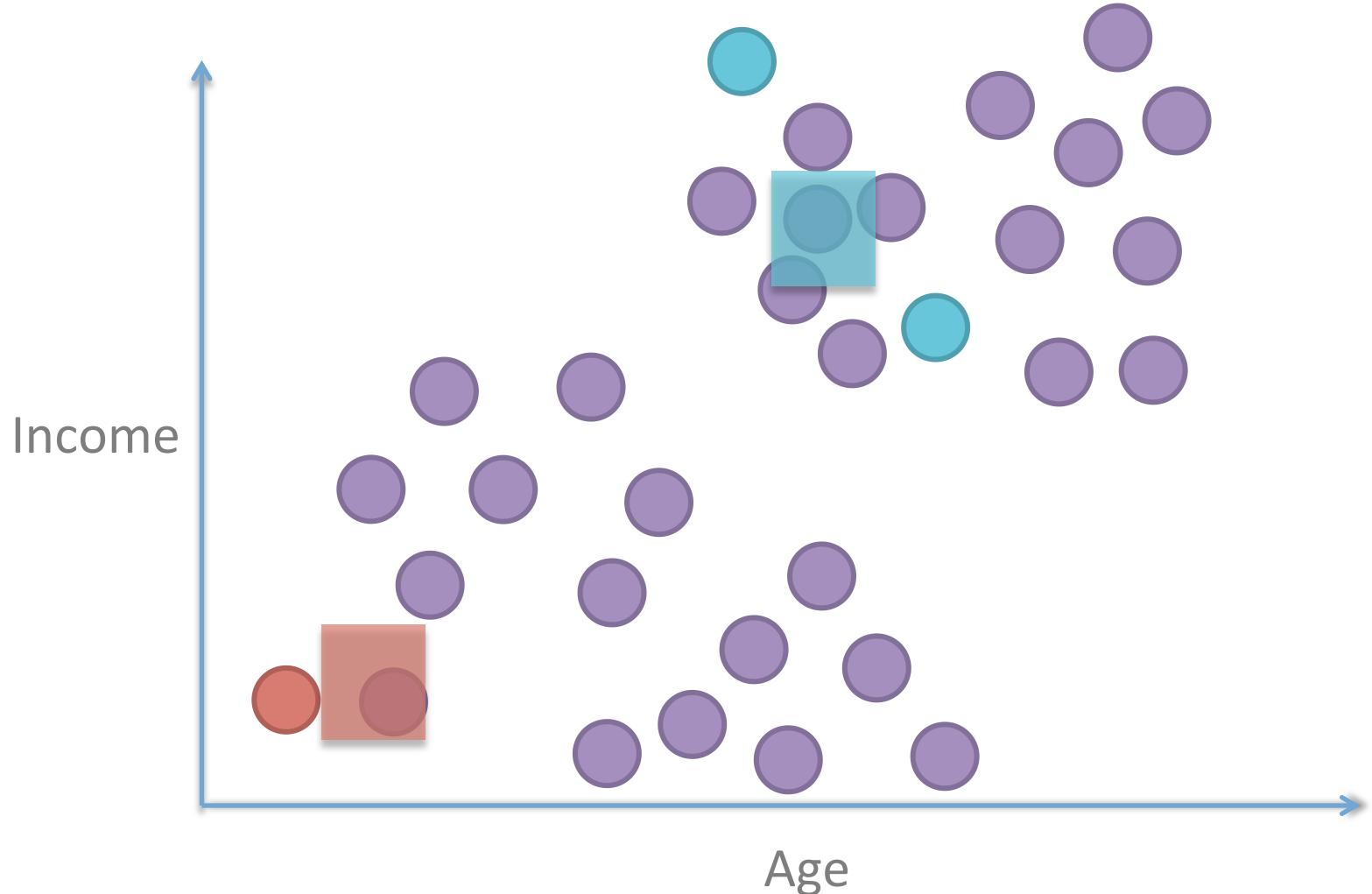
Mean Shift

Sample local density, follow gradient towards denser direction



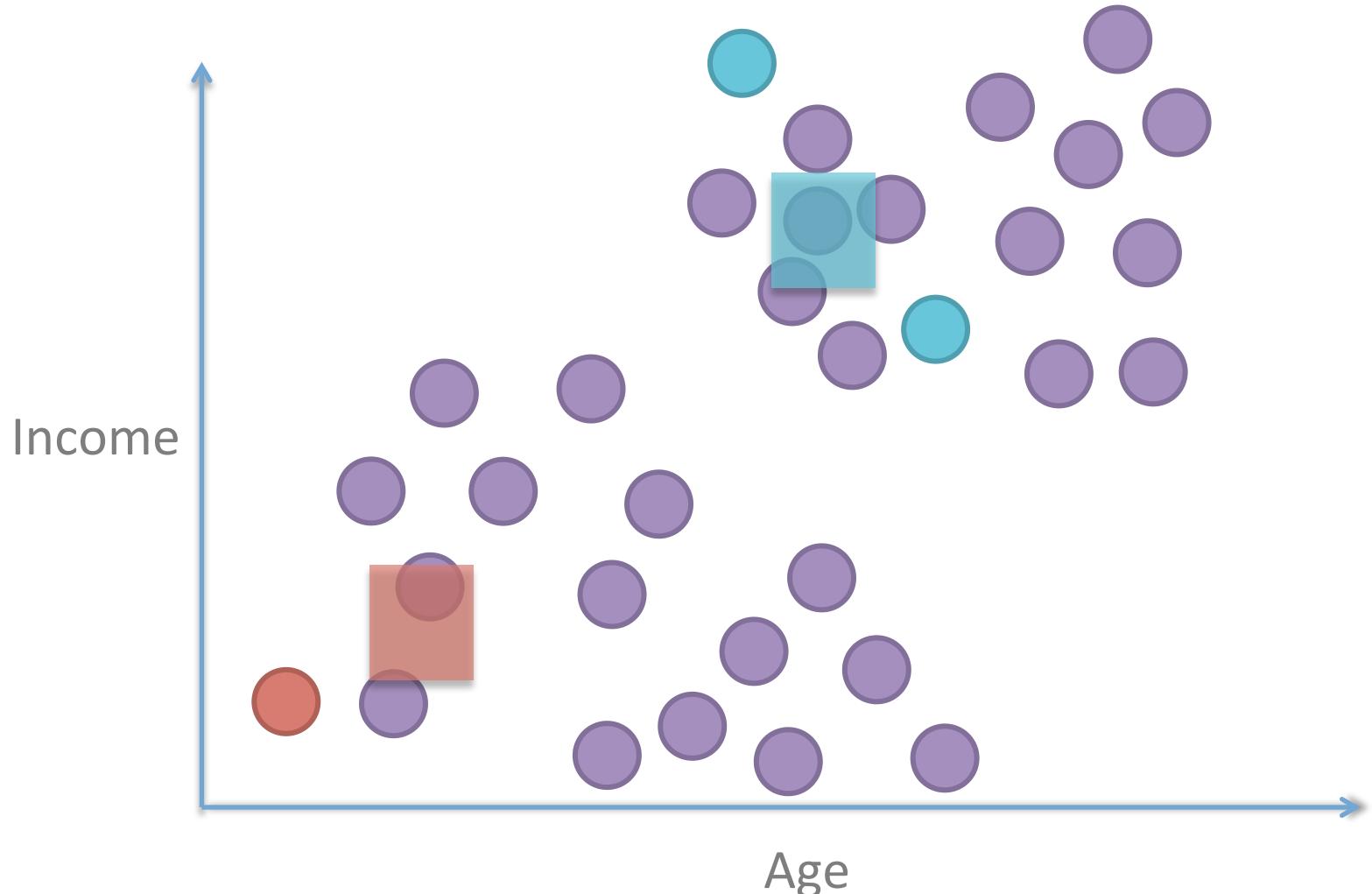
Mean Shift

Sample local density, follow gradient towards denser direction



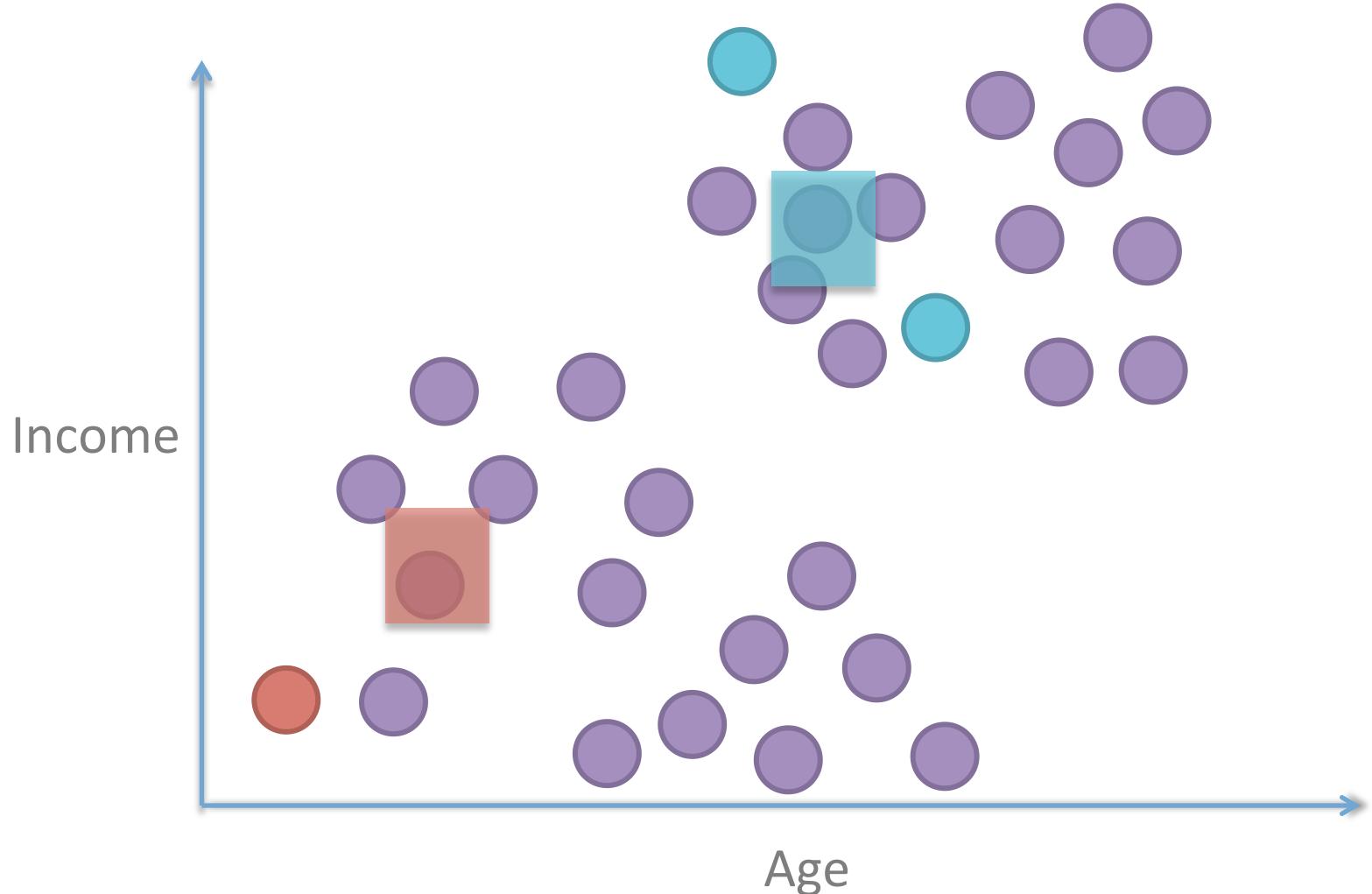
Mean Shift

Sample local density, follow gradient towards denser direction



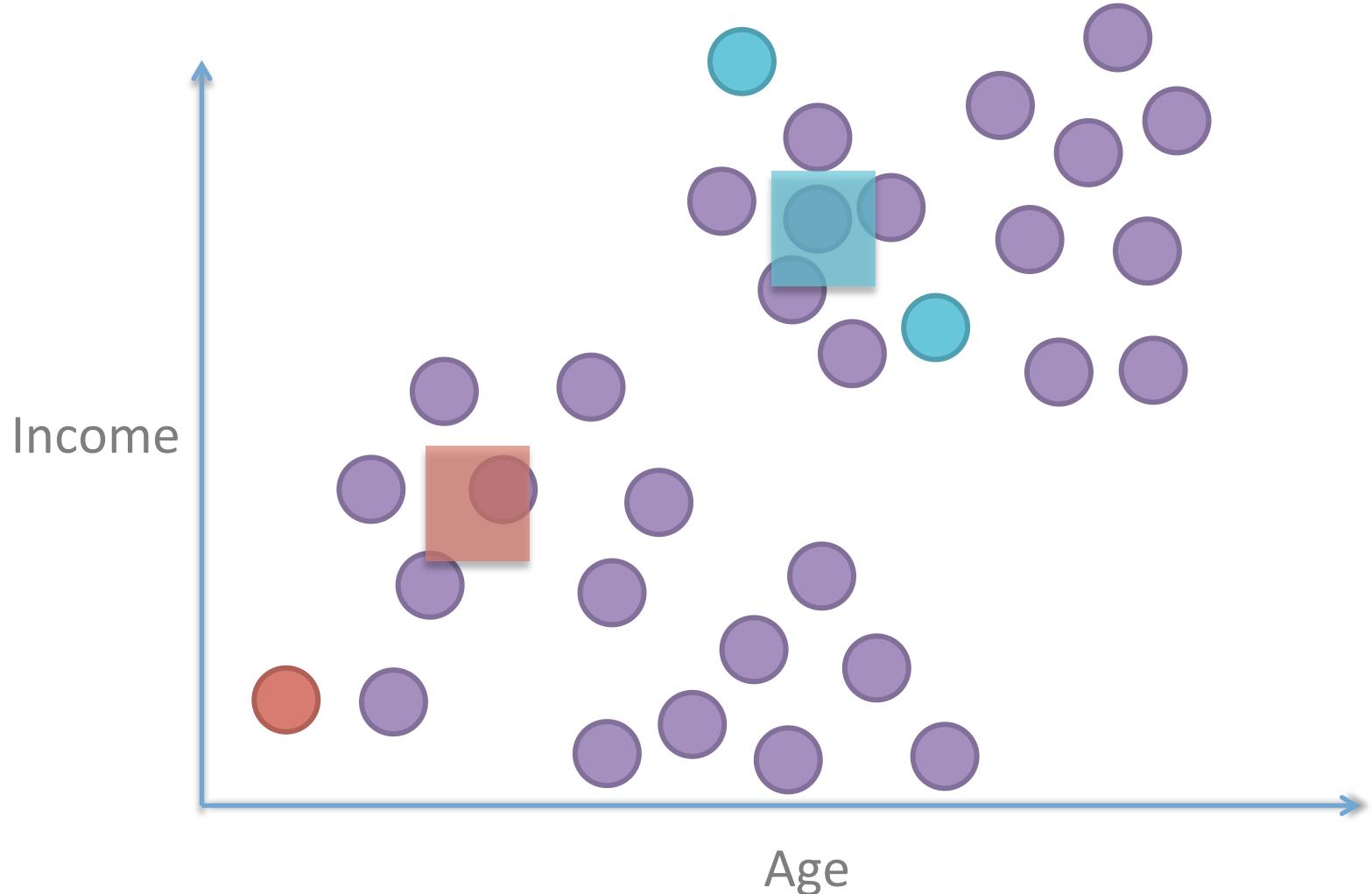
Mean Shift

Sample local density, follow gradient towards denser direction



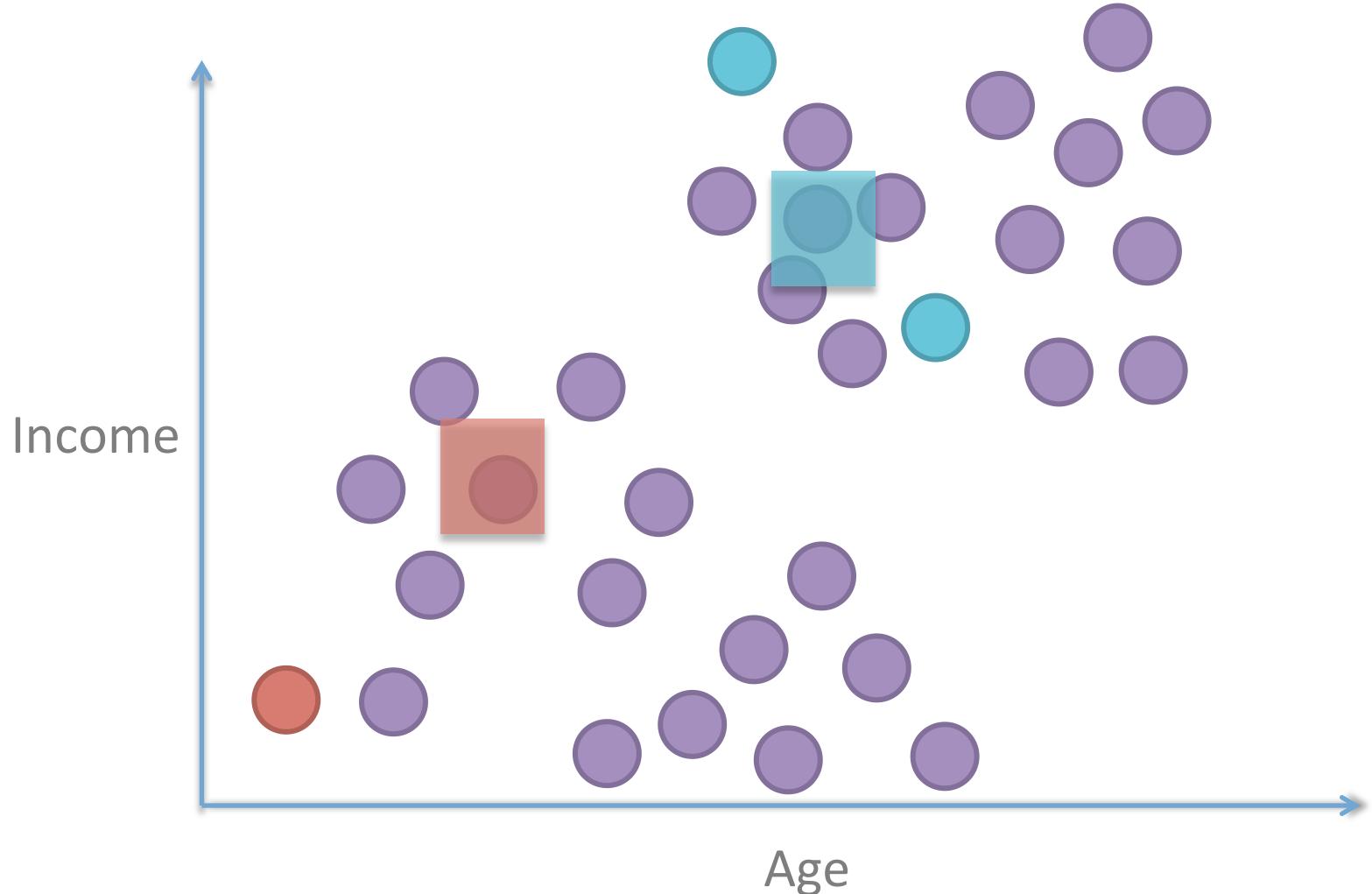
Mean Shift

Sample local density, follow gradient towards denser direction



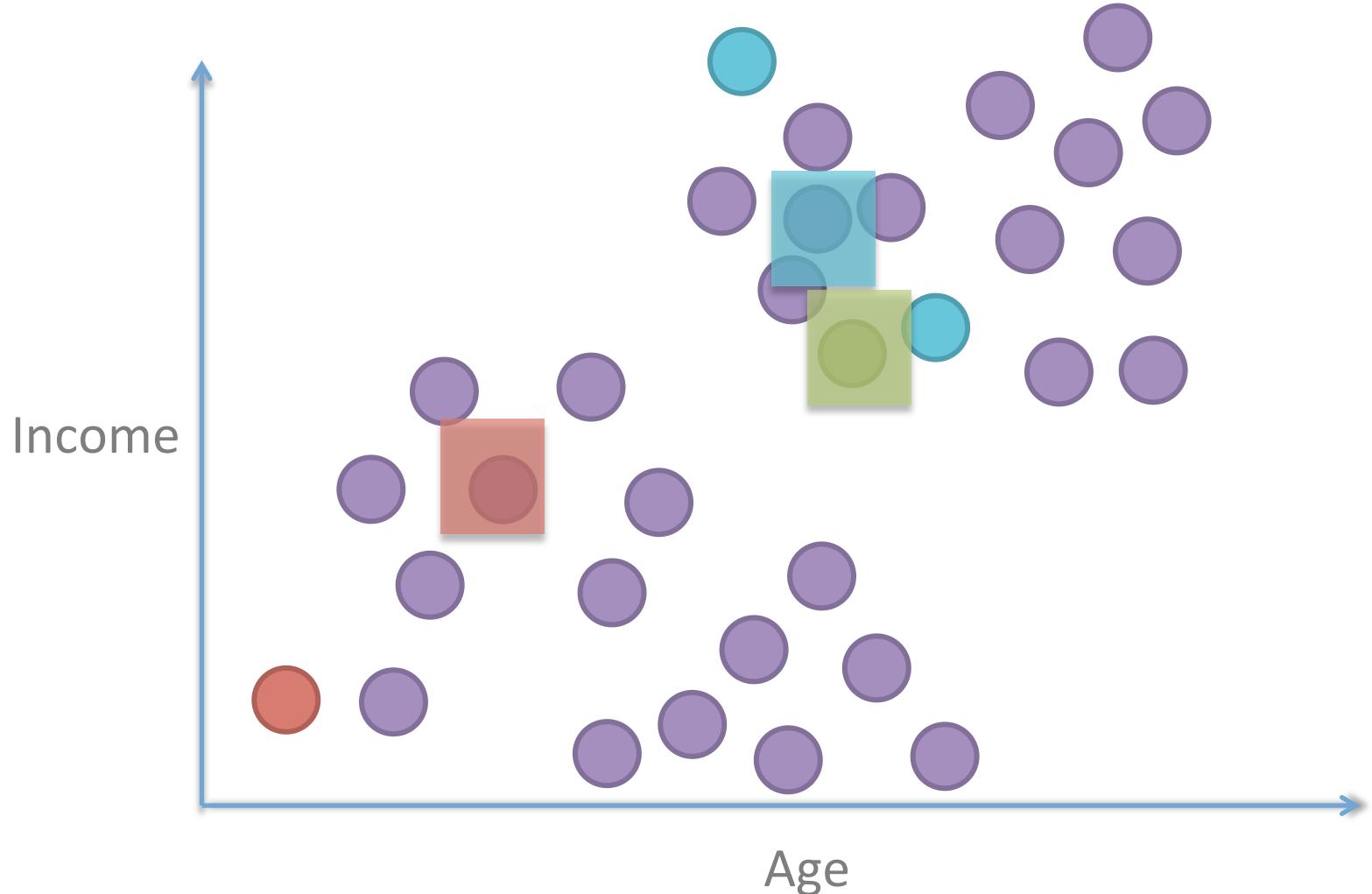
Mean Shift

Found local maximum. Stop.



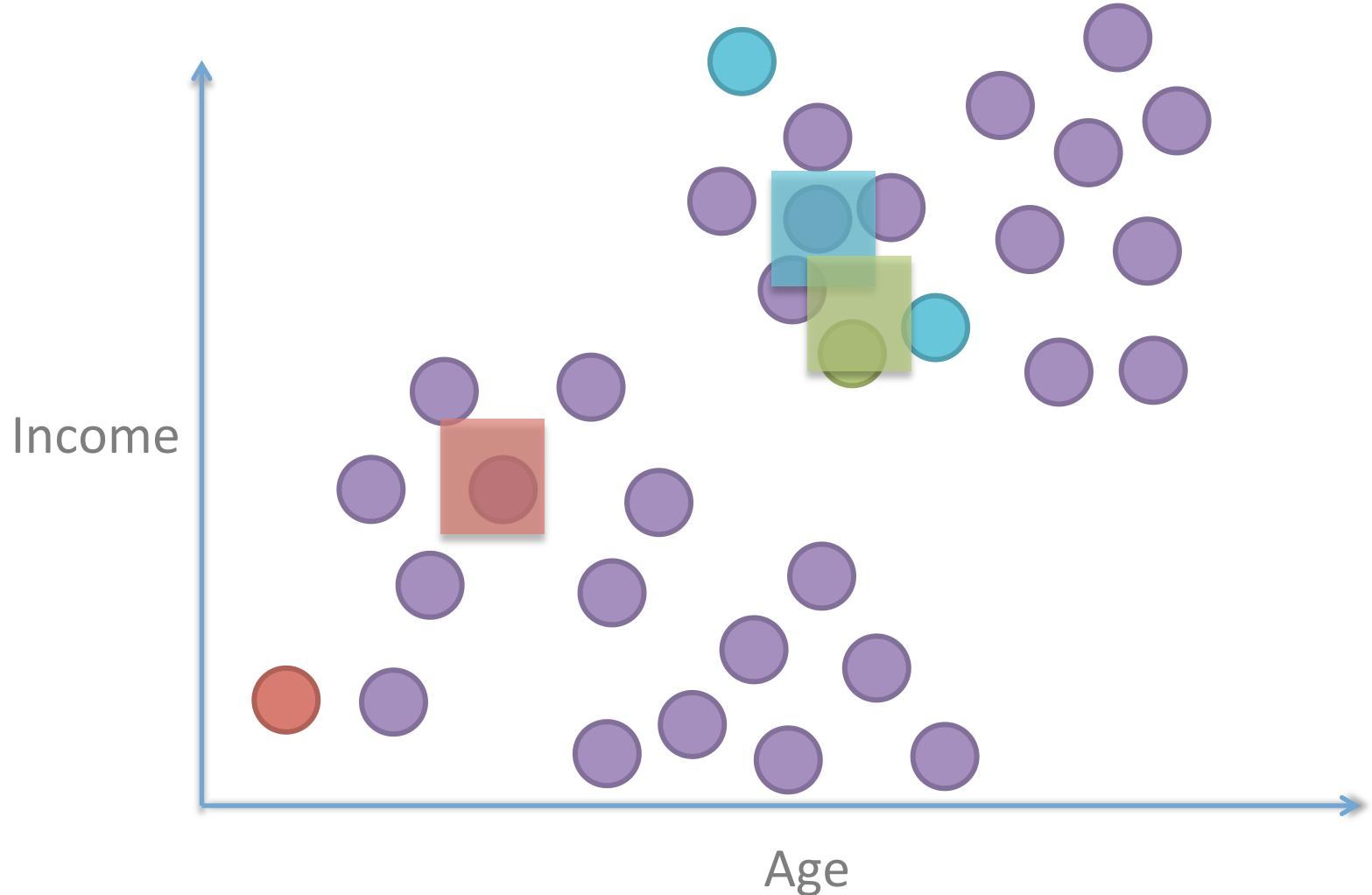
Mean Shift

Start at another point



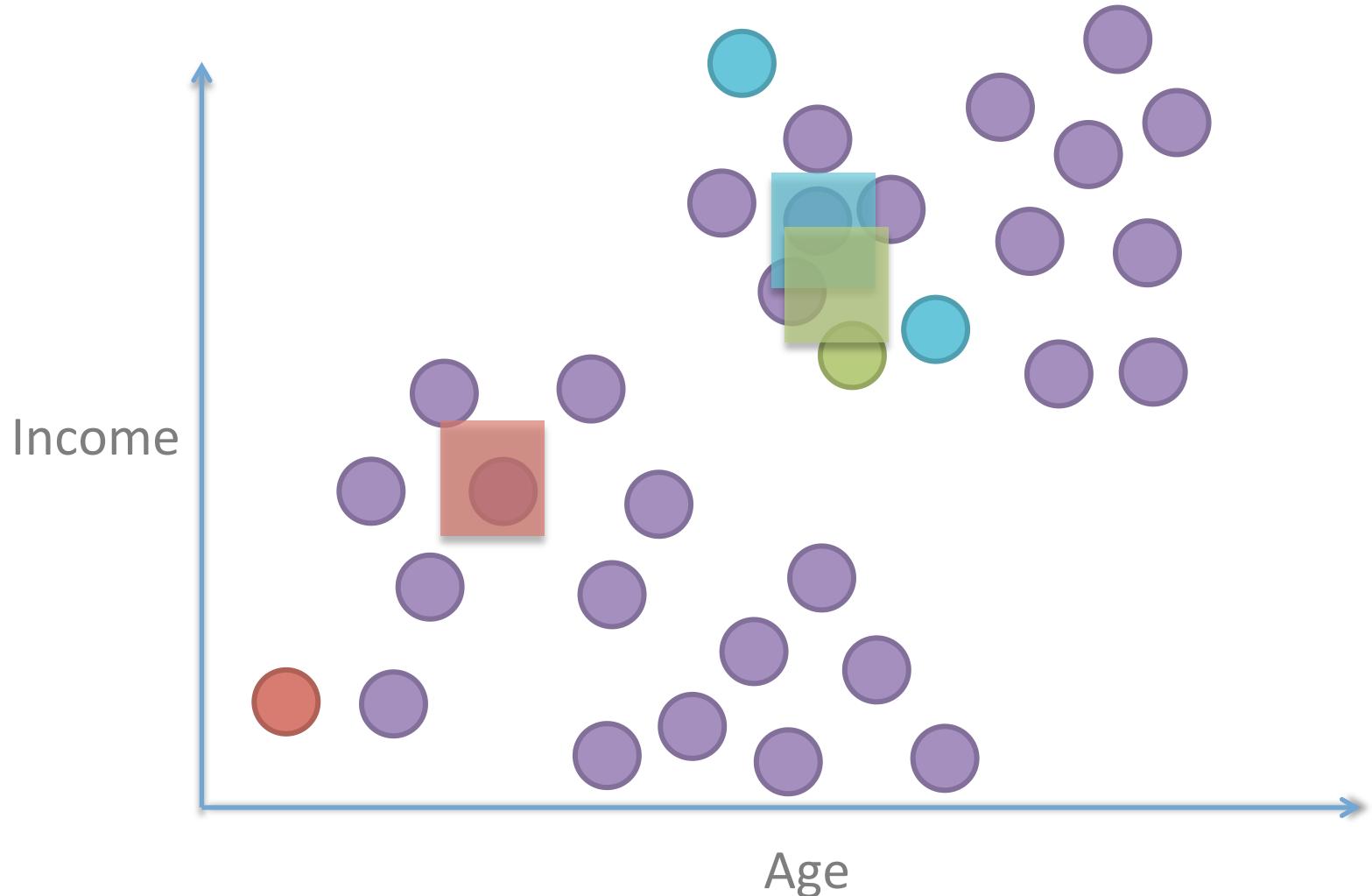
Mean Shift

Search for local density maximum



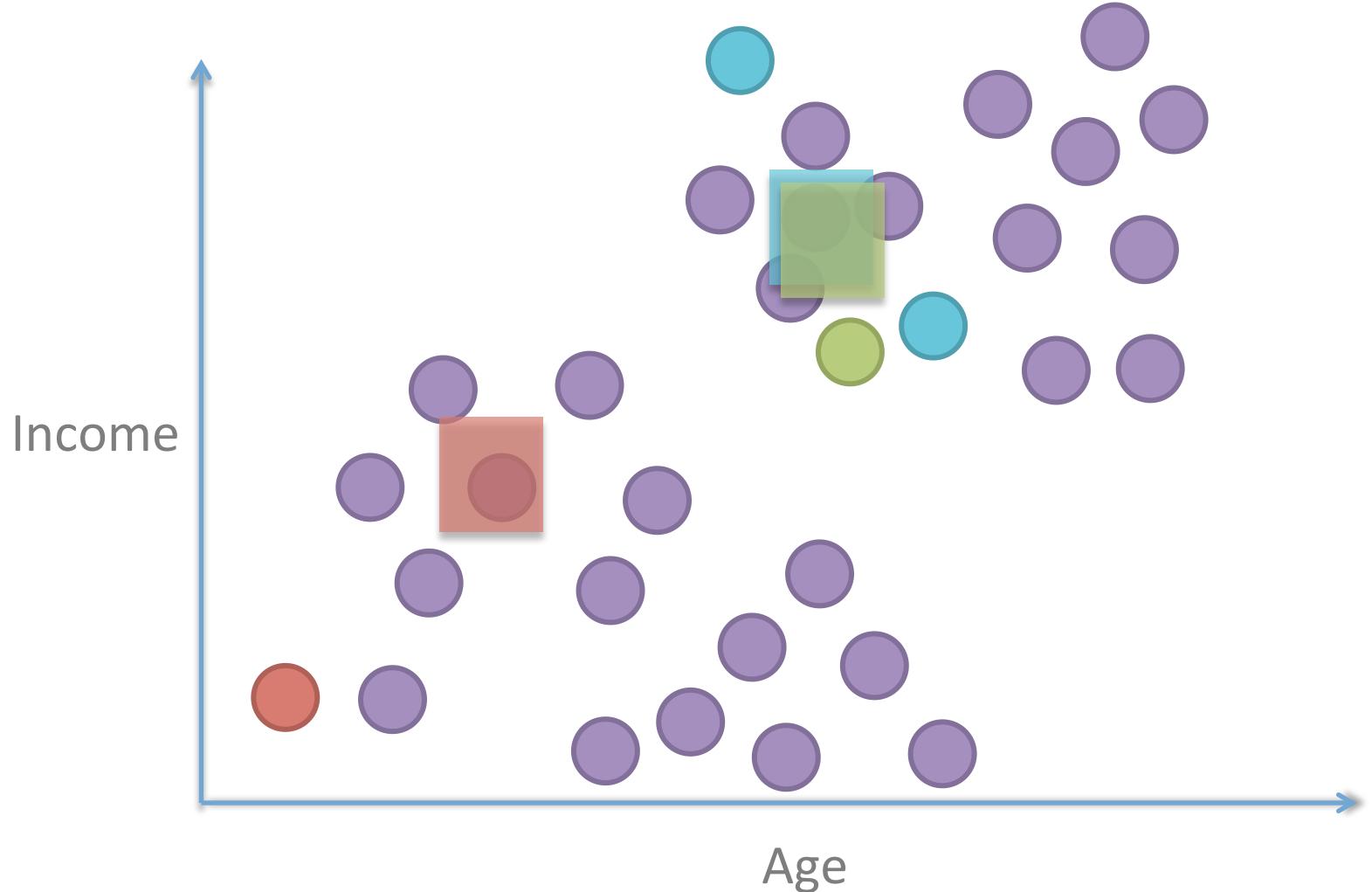
Mean Shift

Search for local density maximum



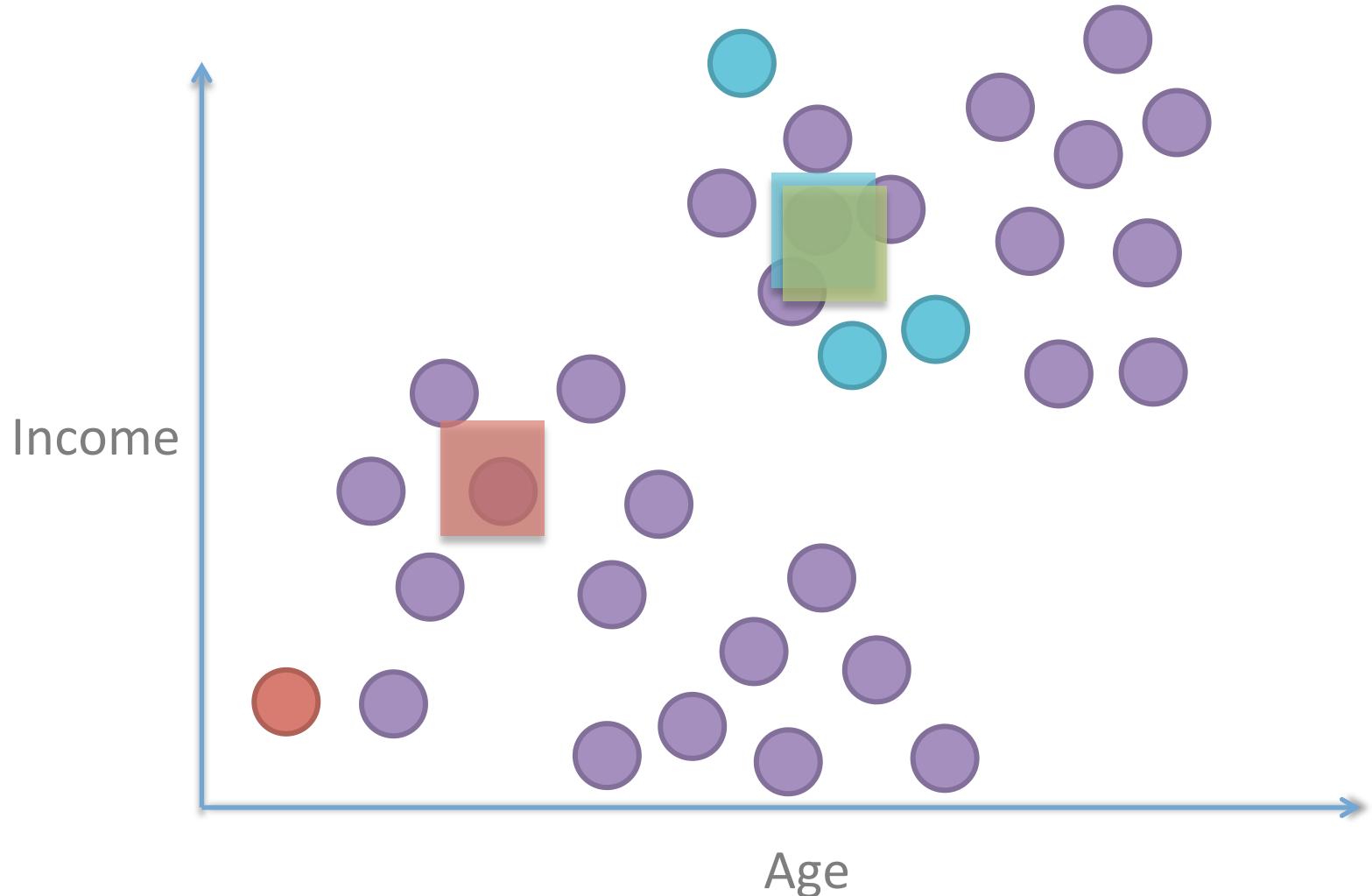
Mean Shift

Found same maximum, same cluster



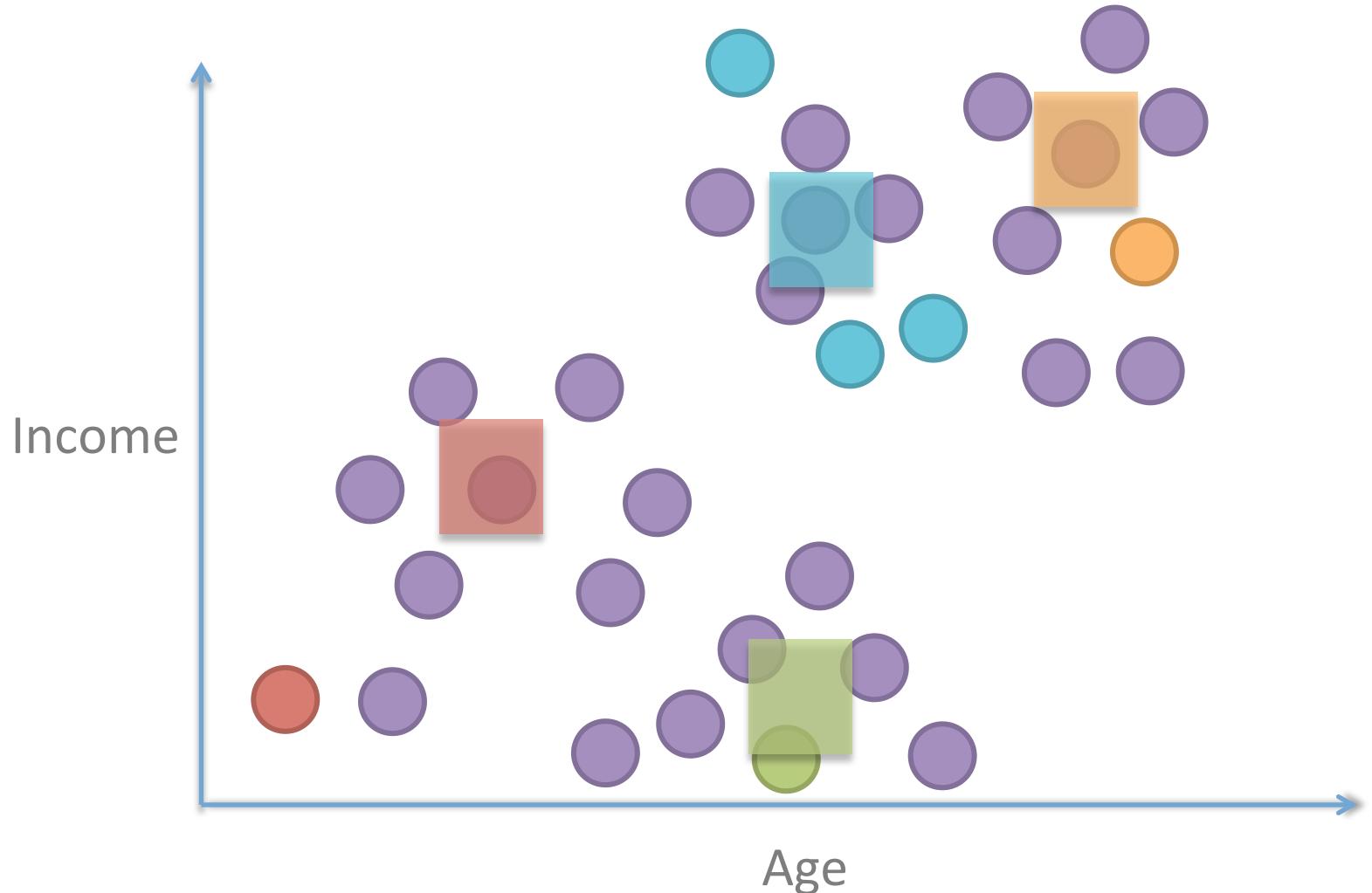
Mean Shift

Found same maximum, same cluster



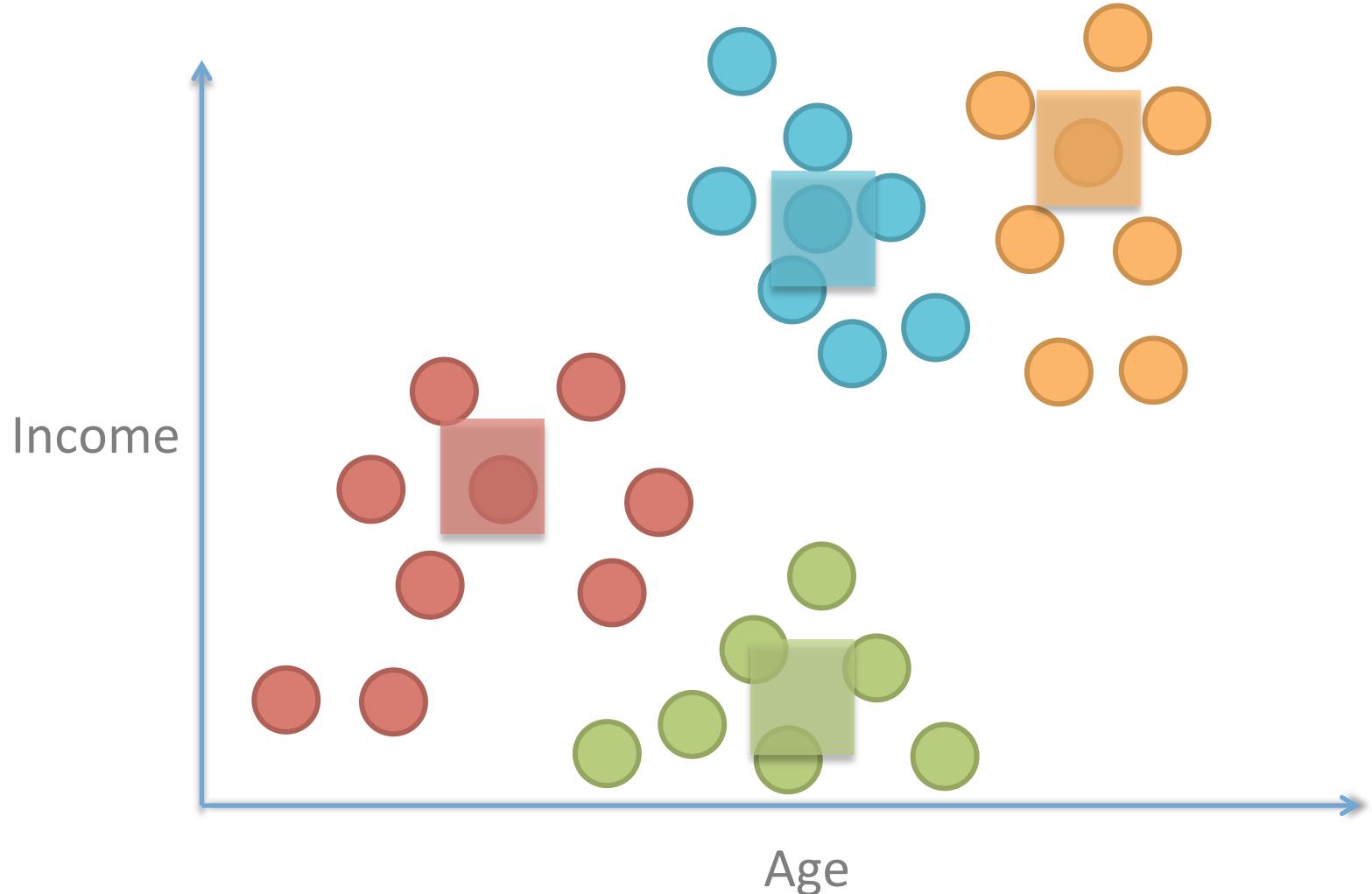
Mean Shift

Keep going like this, eventually it finds four unique local maxima



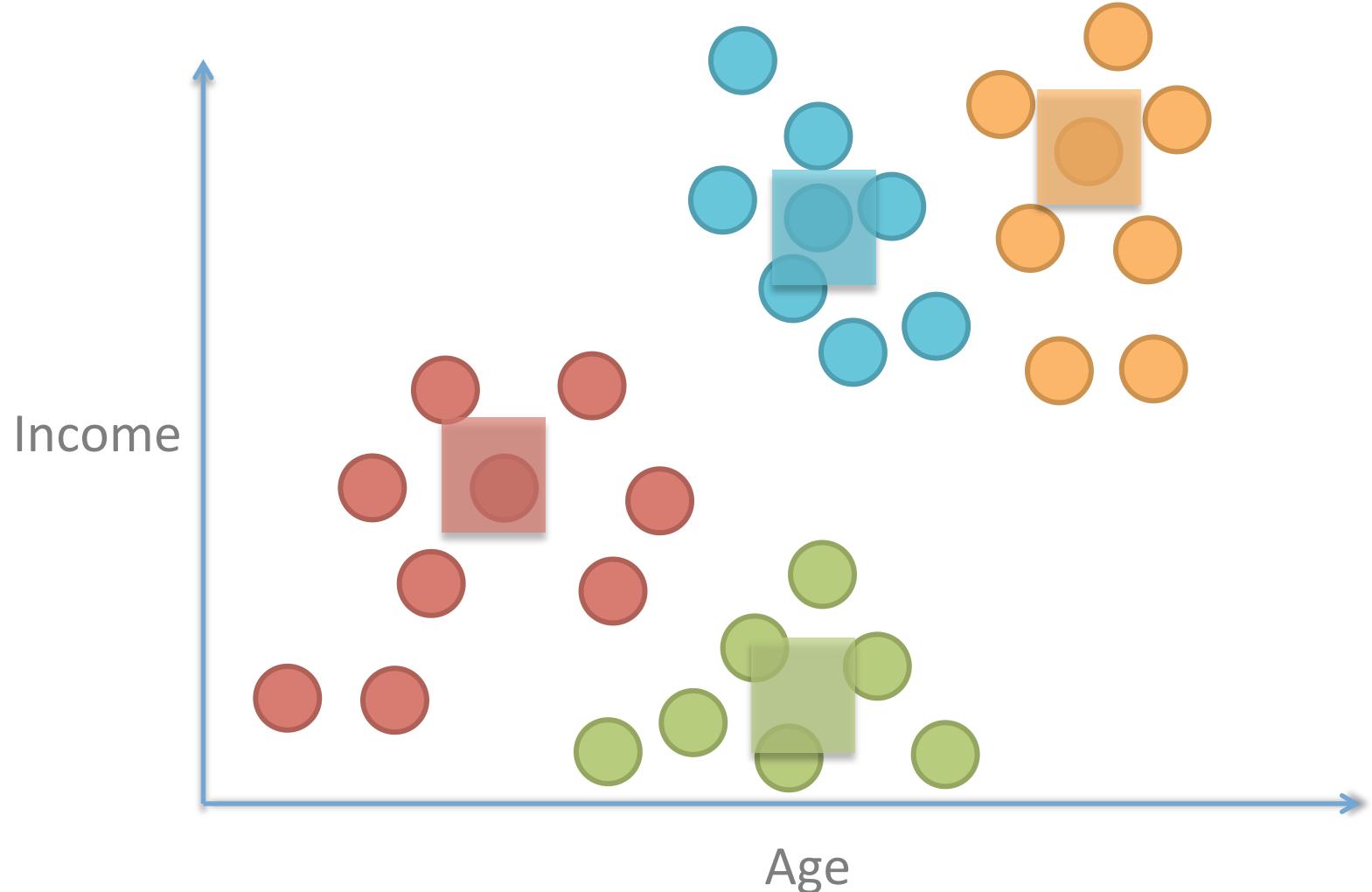
Mean Shift

Assigns points to centroids they fall to



Mean Shift

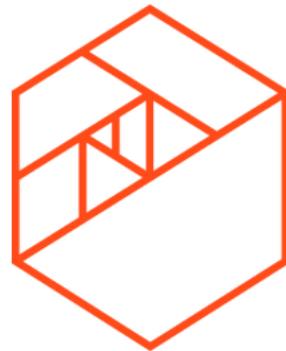
No cluster number or distance parameters



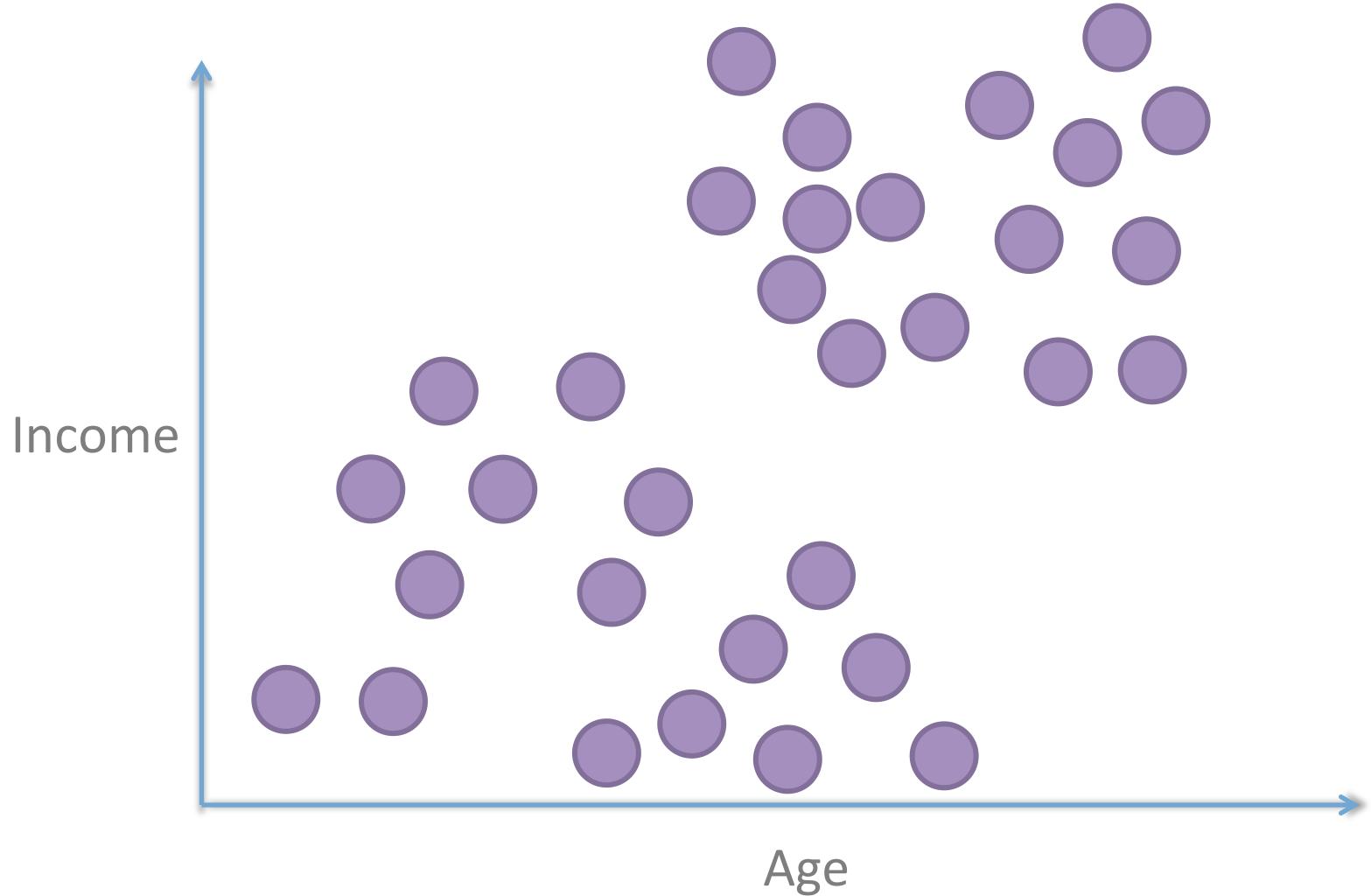
```
from sklearn.cluster import MeanShift
```

```
MeanShift()
```

Spectral Clustering (kernel k--means)

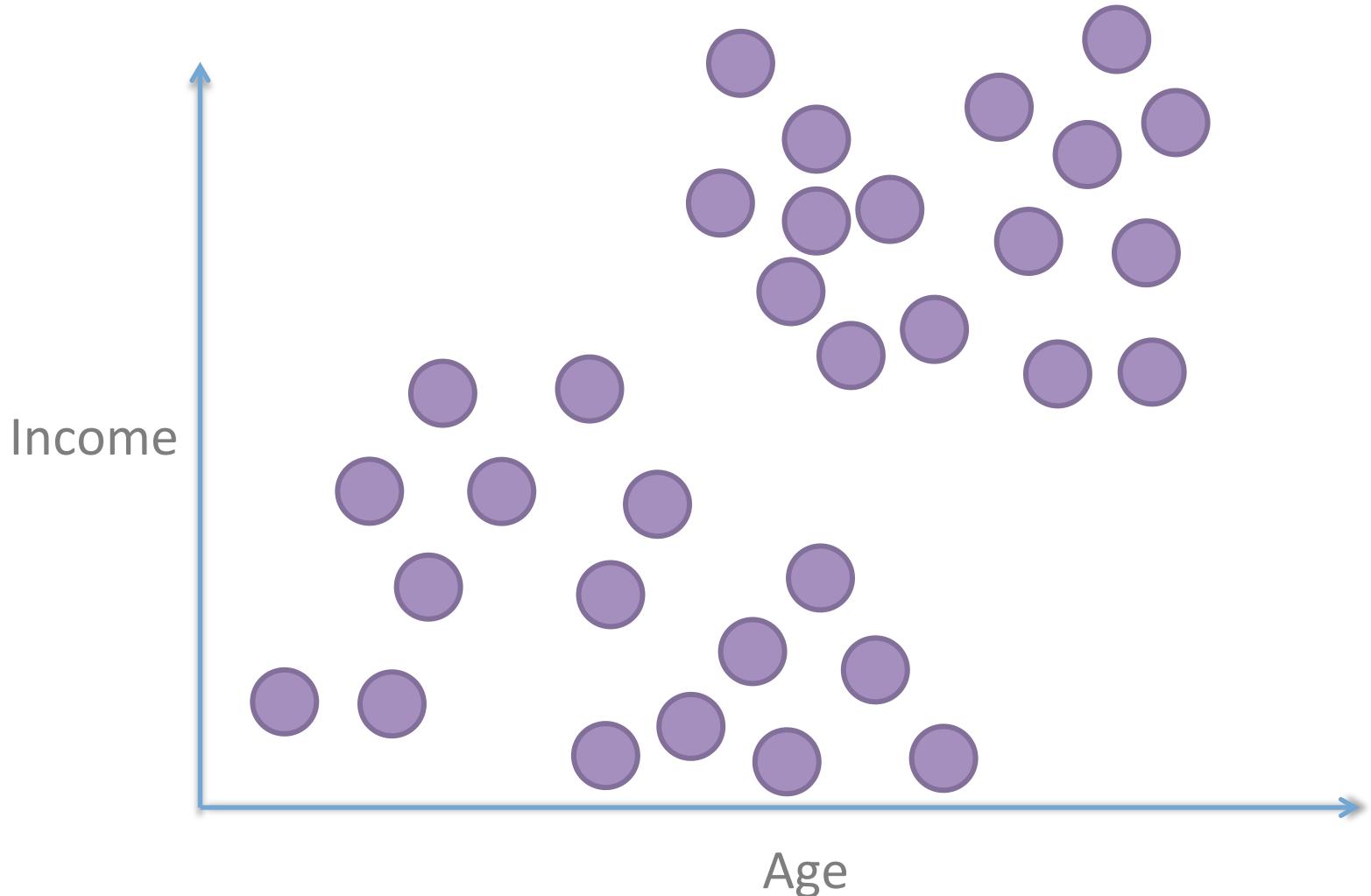


Spectral Clustering



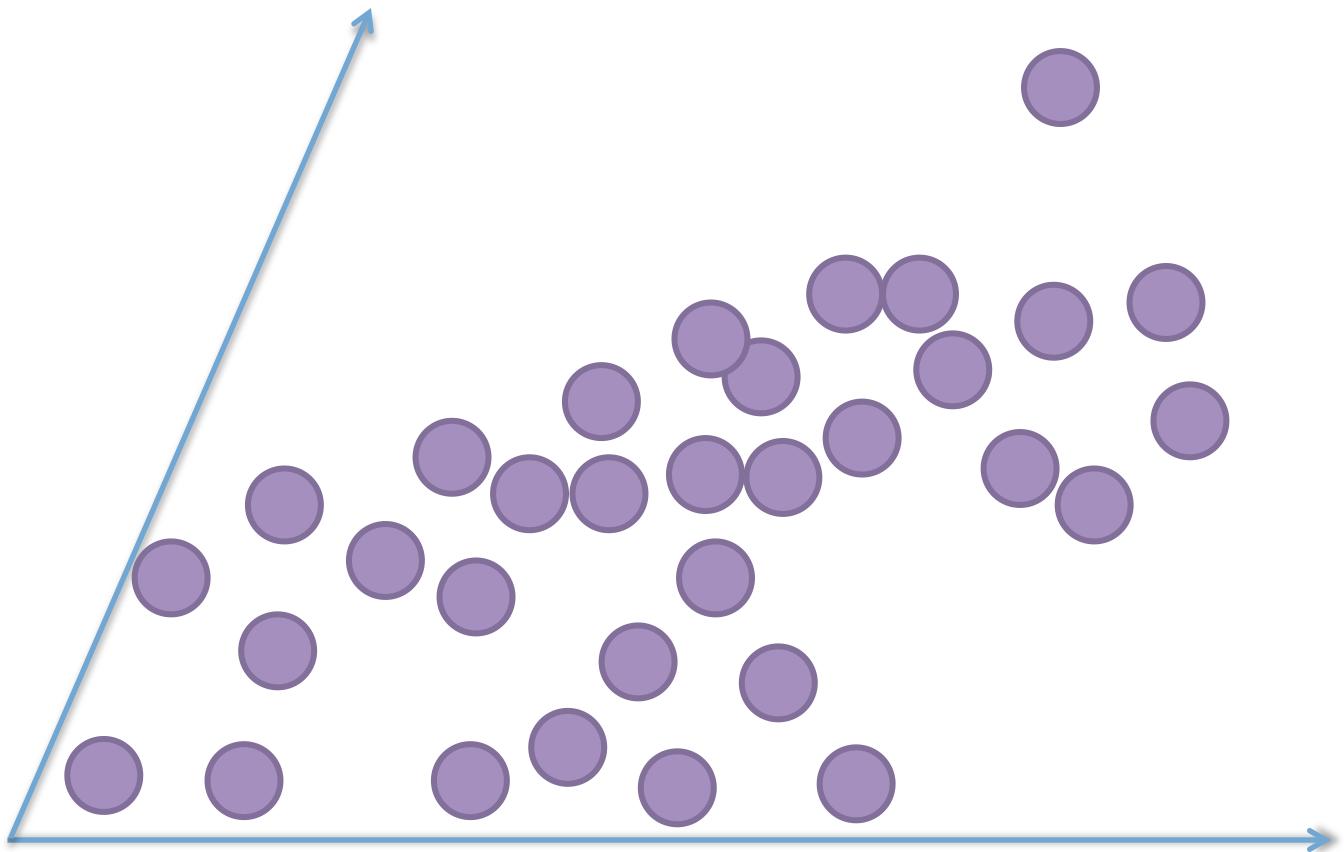
Spectral Clustering

Transform the space into fewer dimensions by taking eigenvectors of the distances matrix



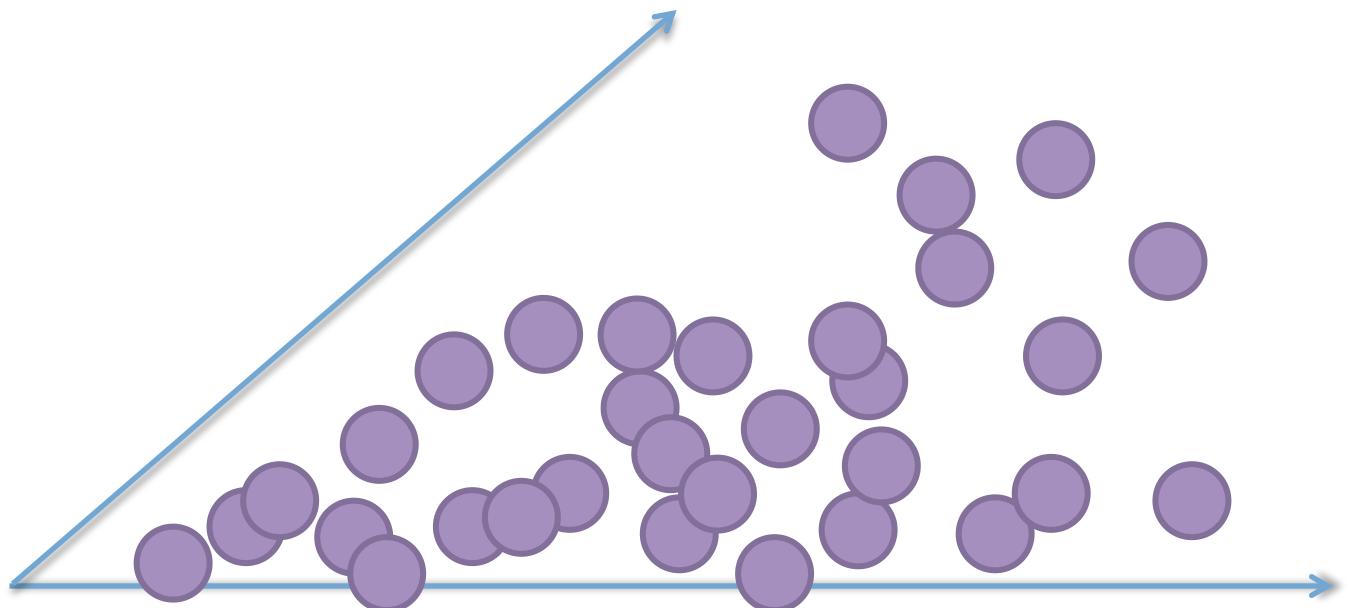
Spectral Clustering

Transform the space into fewer dimensions by taking eigenvectors of the distances matrix



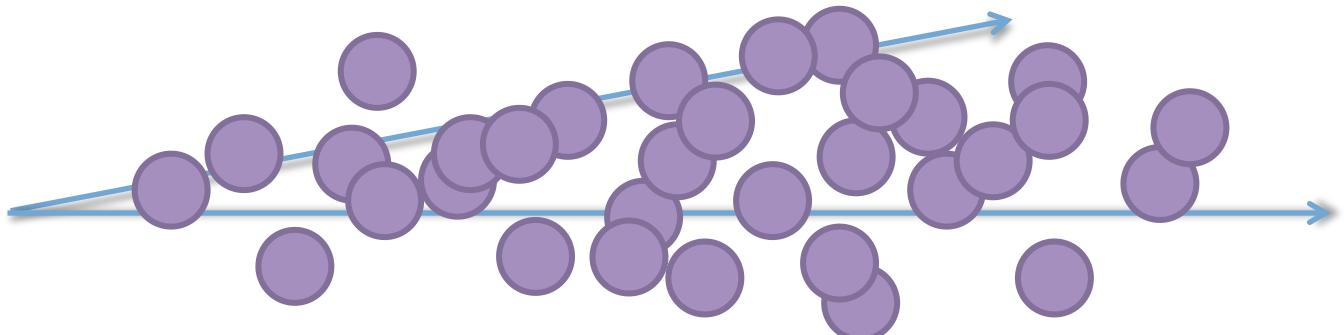
Spectral Clustering

Transform the space into fewer dimensions by taking eigenvectors of the distances matrix



Spectral Clustering

Transform the space into fewer dimensions by taking eigenvectors of the distances matrix



Spectral Clustering

Transform the space into fewer dimensions by taking eigenvectors of the distances matrix



Spectral Clustering

Now perform K-Means on the low dimension space (k=3)



Spectral Clustering

Now perform K-Means on the low dimension space



Spectral Clustering

Now perform K-Means on the low dimension space



Spectral Clustering

Now perform K-Means on the low dimension space



Spectral Clustering

Now perform K-Means on the low dimension space



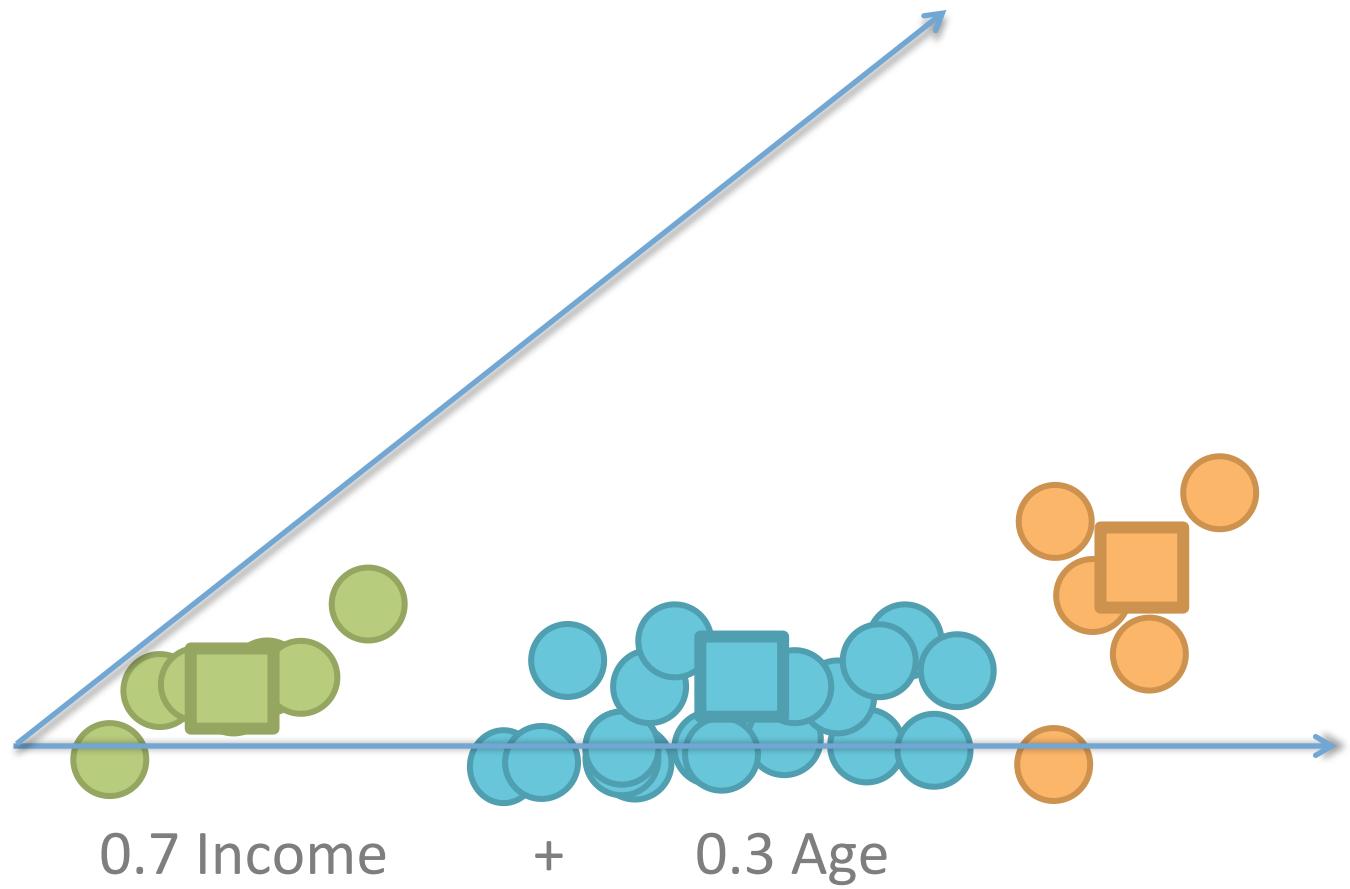
Spectral Clustering

Look at them in the normal space



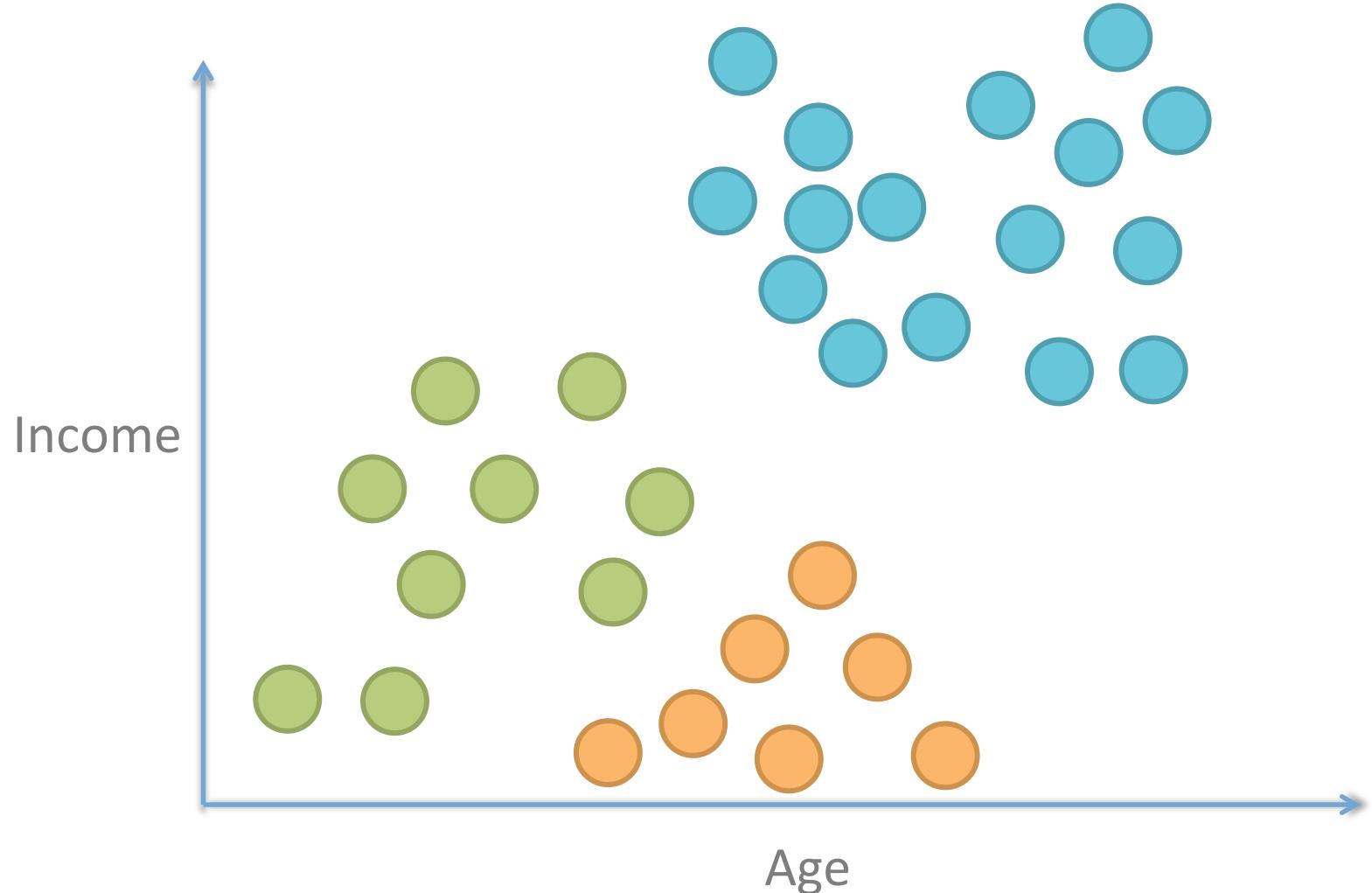
Spectral Clustering

Look at them in the normal space



Spectral Clustering

Look at them in the normal space

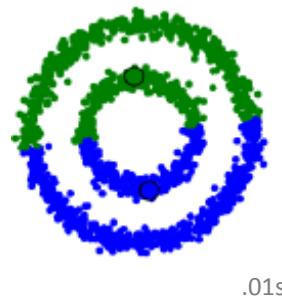


```
from sklearn.cluster import SpectralClustering  
  
SpectralClustering( n_clusters=8, n_init=10 )
```

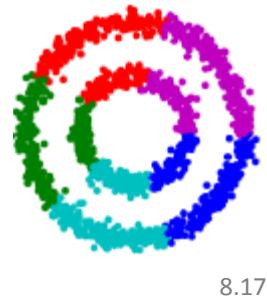
Properties, Advantages, Disadvantages



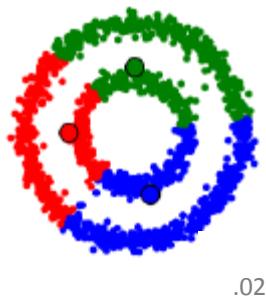
MiniBatchKMeans



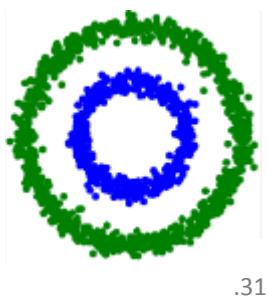
AffinityPropagation



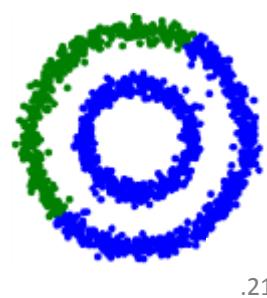
MeanShift



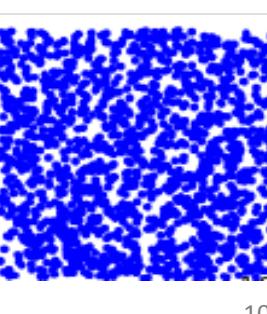
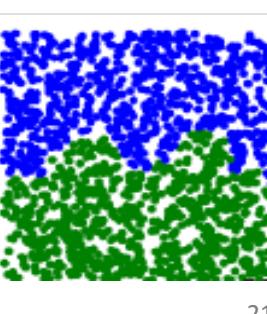
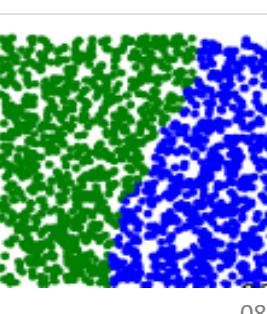
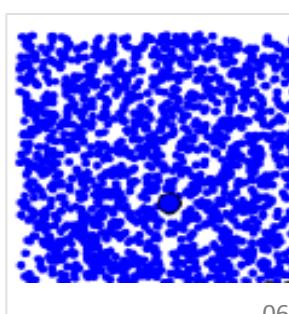
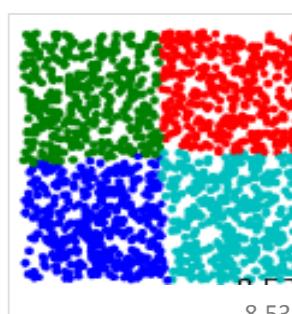
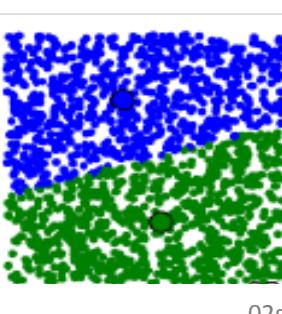
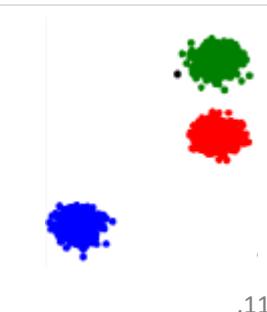
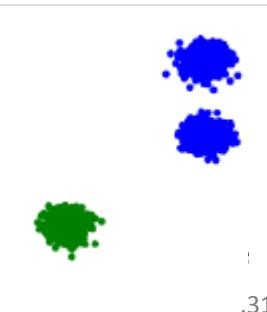
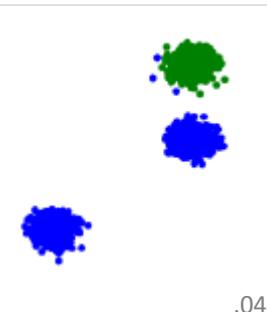
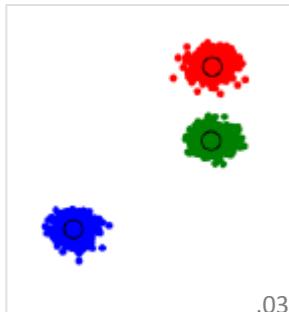
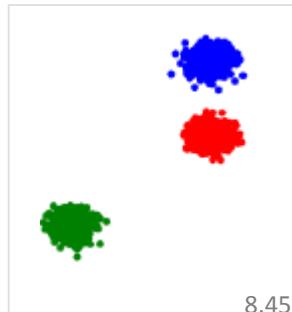
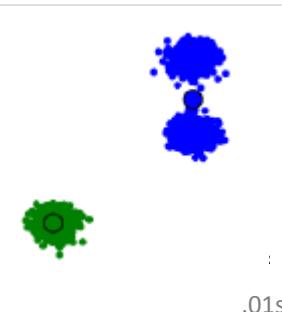
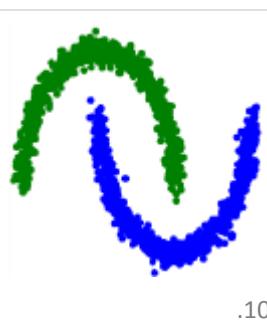
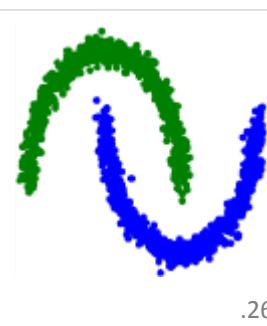
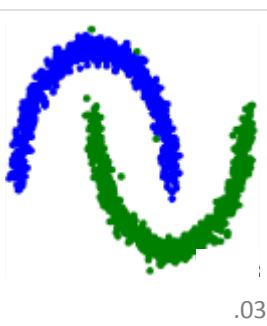
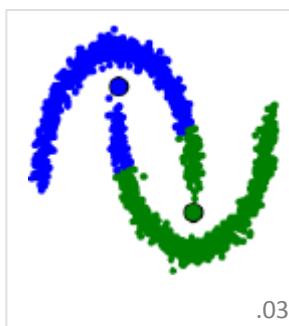
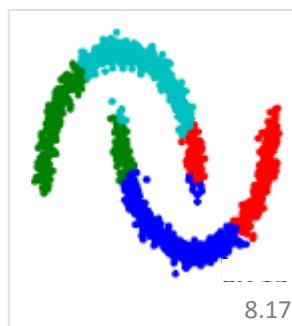
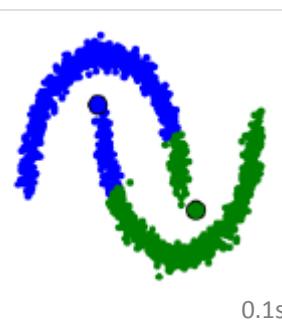
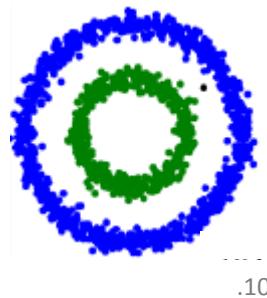
SpectralClustering



Ward



DBSCAN



Method name	Parameters	Scalability	Usecase	Geometry (metric used)
<i>K-means</i>	Number of clusters	Very large $n_{samples}$ medium $n_{clusters}$ with <i>MiniBatch code</i>	General purpose, even cluster size, flat geometry, not too many clusters	Distances between points
<i>Affinity propagation</i>	Damping, sample preferences	Not scalable with $n_{samples}$	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
<i>Mean-shift</i>	Bandwidth	Not scalable with $n_{samples}$	Many clusters, uneven cluster size, non-flat geometry	Distances between points
<i>Spectral clustering</i>	Number of clusters	Medium $n_{samples}$, small $n_{clusters}$	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
<i>Hierarchical clustering</i>	Number of clusters	Large $n_{samples}$ and $n_{clusters}$	Many clusters, possibly connectivity constraints	Distances between points
<i>DBSCAN</i>	Neighborhood size	Very large $n_{samples}$, medium $n_{clusters}$	Non-flat geometry, uneven cluster sizes	Distances between nearest points

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-means	Number of clusters	Very large $n_{samples}$ medium $n_{clusters}$ with <i>MiniBatch code</i>	General purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	Damping, sample preferences	Not scalable with $n_{samples}$	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	Bandwidth	Not scalable with $n_{samples}$	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	Number of clusters	Medium $n_{samples}$, small $n_{clusters}$	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Hierarchical clustering	Number of clusters	Large $n_{samples}$ and $n_{clusters}$	Many clusters, possibly connectivity constraints	Distances between points
DBSCAN	Neighborhood size	Very large $n_{samples}$, medium $n_{clusters}$	Non-flat geometry, uneven cluster sizes	Distances between nearest points

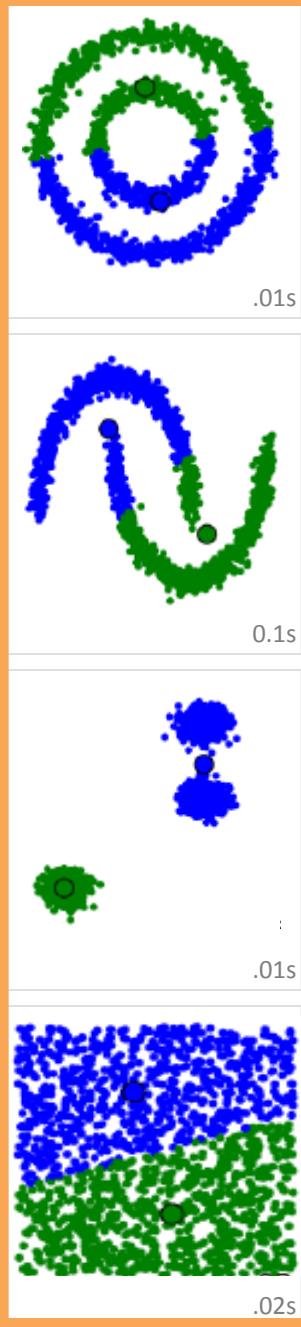
K Means

Minibatch version is super fast (big data).

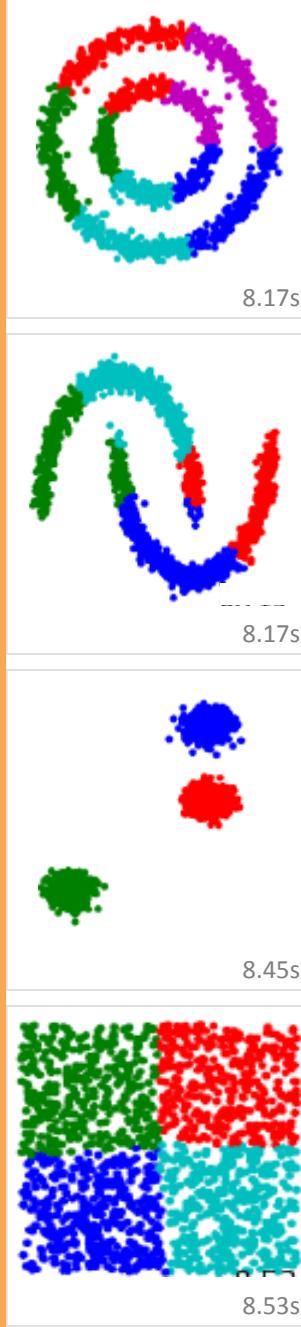
Have to try k values (k not too big).
Tends to find even sized.

Bad with non---spherical cluster shapes.
Only euclidean distance.

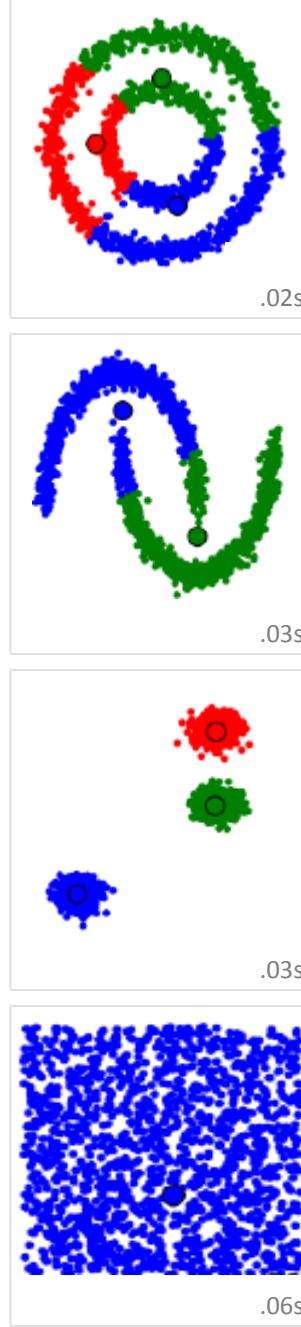
MiniBatchKMeans



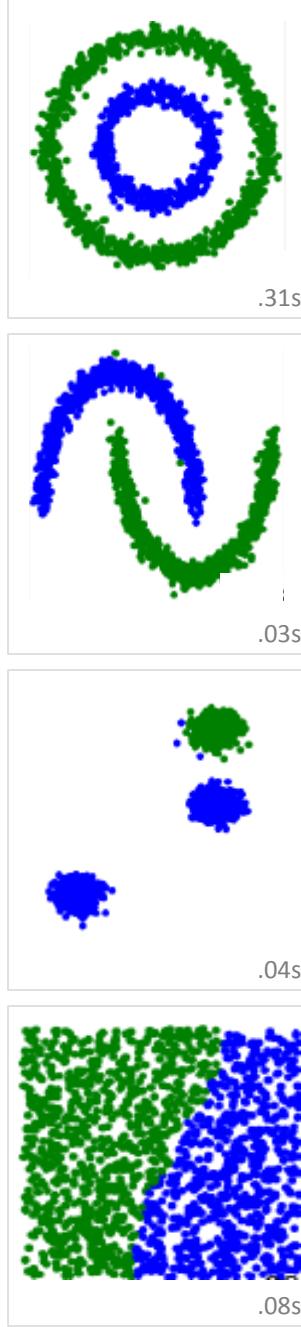
AffinityPropagation



MeanShift



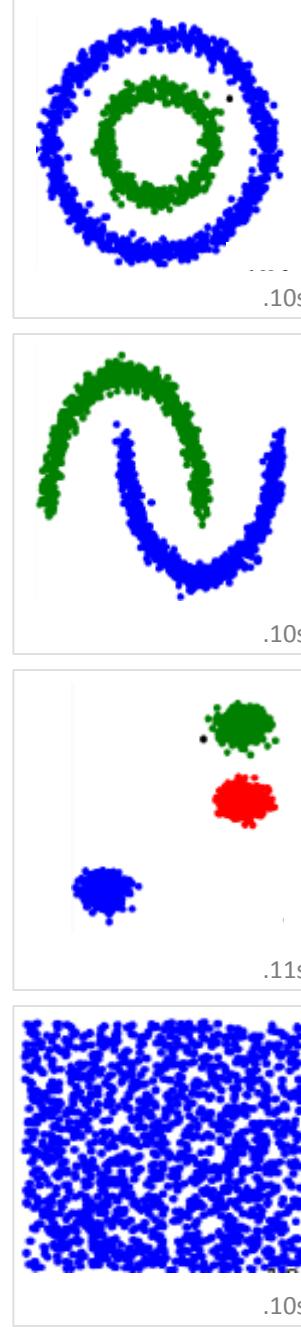
SpectralClustering



Ward



DBSCAN



Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-means	Number of clusters	Very large $n_{samples}$ medium $n_{clusters}$ with <i>MiniBatch code</i>	General purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	Damping, sample preferences	Not scalable with $n_{samples}$	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	Bandwidth	Not scalable with $n_{samples}$	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	Number of clusters	Medium $n_{samples}$, small $n_{clusters}$	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Hierarchical clustering	Number of clusters	Large $n_{samples}$ and $n_{clusters}$	Many clusters, possibly connectivity constraints	Distances between points
DBSCAN	Neighborhood size	Very large $n_{samples}$, medium $n_{clusters}$	Non-flat geometry, uneven cluster sizes	Distances between nearest points

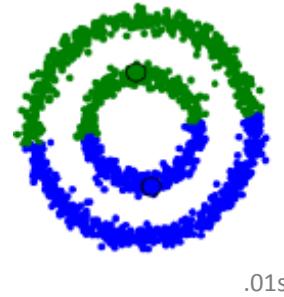
Hierarchical Clustering

You get a full hierarchy tree. Useful for some problems.
Have to try k values.

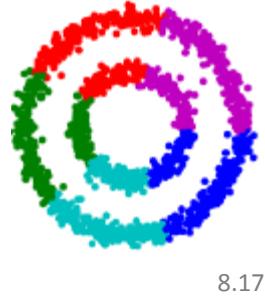
Finds uneven cluster sizes (one is big, some are tiny, etc.)
This is due to the “the rich get richer effect.” A lot of
distance metric and linkage options.

Fast. Can easily handle lots of data and clusters.

MiniBatchKMeans

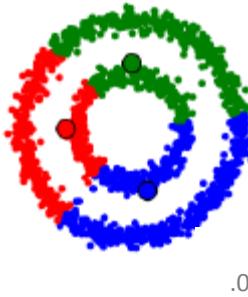


AffinityPropagation



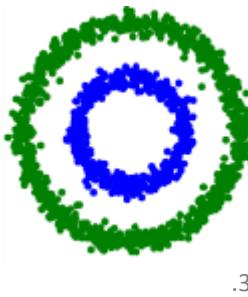
8.17s

MeanShift



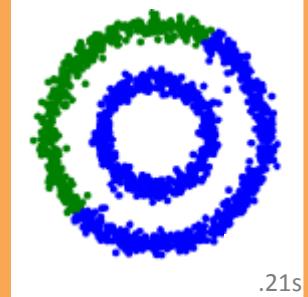
.02s

SpectralClustering



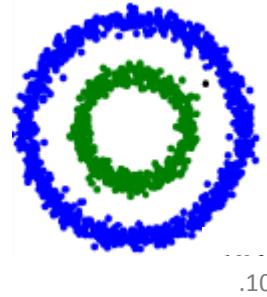
.31s

Ward

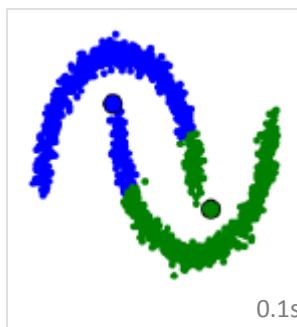


.21s

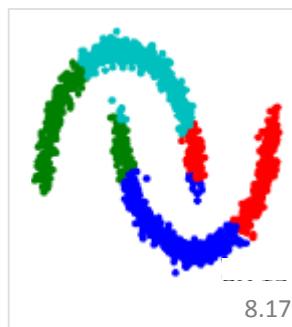
DBSCAN



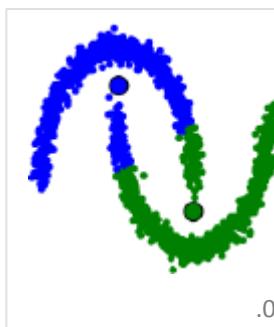
.10s



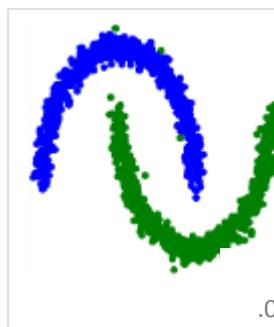
0.1s



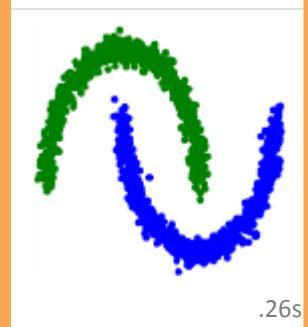
8.17s



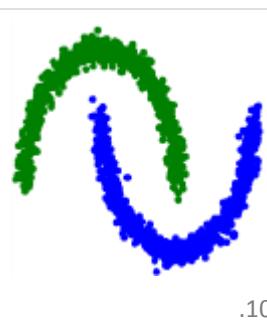
.03s



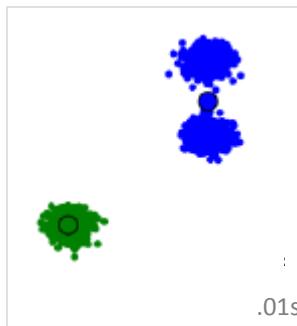
.03s



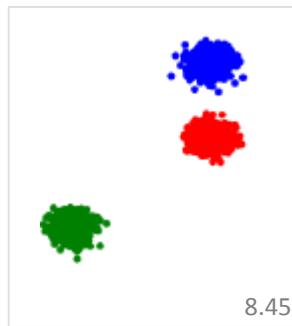
.26s



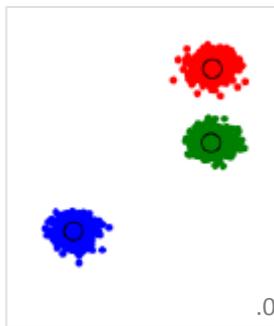
.10s



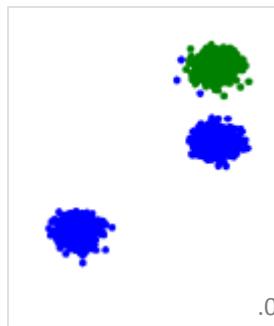
.01s



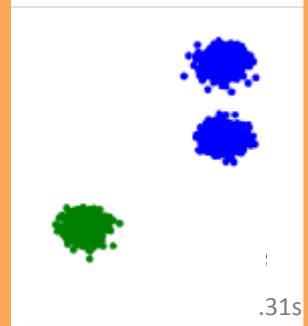
8.45s



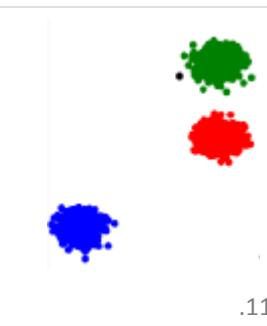
.03s



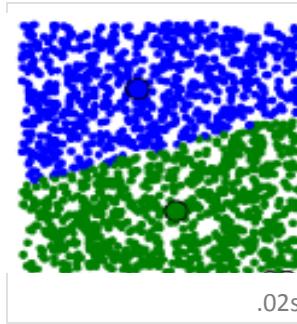
.04s



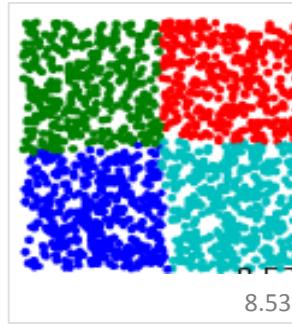
.31s



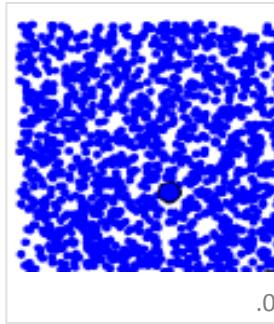
.11s



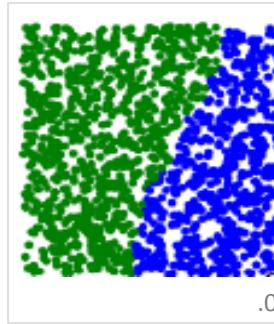
.02s



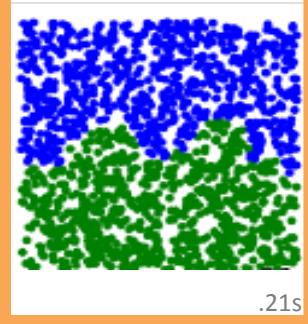
8.53s



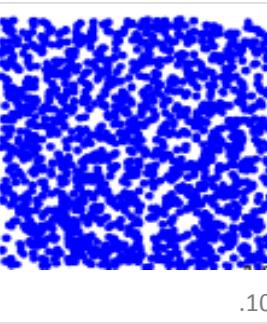
.06s



.08s



.21s



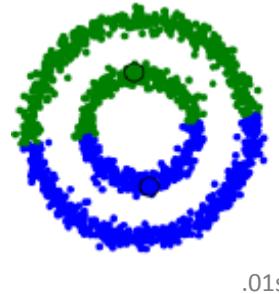
.10s

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-means	Number of clusters	Very large $n_{samples}$ medium $n_{clusters}$ with <i>MiniBatch code</i>	General purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	Damping, sample preferences	Not scalable with $n_{samples}$	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	Bandwidth	Not scalable with $n_{samples}$	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	Number of clusters	Medium $n_{samples}$, small $n_{clusters}$	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Hierarchical clustering	Number of clusters	Large $n_{samples}$ and $n_{clusters}$	Many clusters, possibly connectivity constraints	Distances between points
DBSCAN	Neighborhood size	Very large $n_{samples}$, medium $n_{clusters}$	Non-flat geometry, uneven cluster sizes	Distances between nearest points

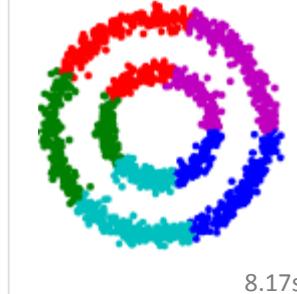
DBSCAN

Density based, on the money with the right parameters.
 Have to try *epsilon* (and *num_clu*) values.
 Can find **uneven cluster sizes**.
Full distance metric options.
 Can handle tons of data and weird shapes.
 Too small epsilon (too many clusters) is a bit shaky.

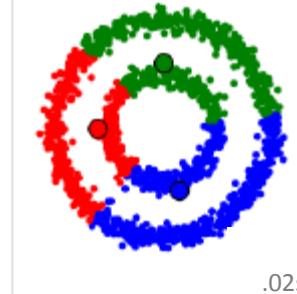
MiniBatchKMeans



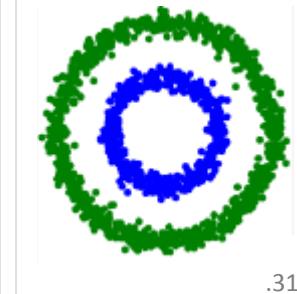
AffinityPropagation



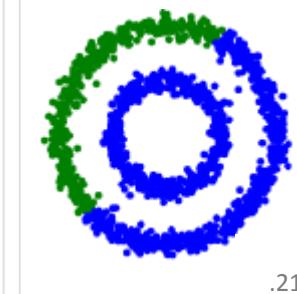
MeanShift



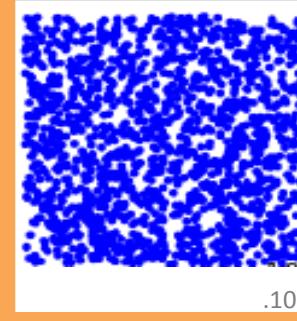
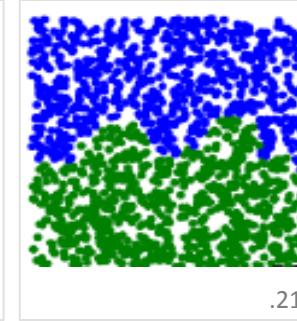
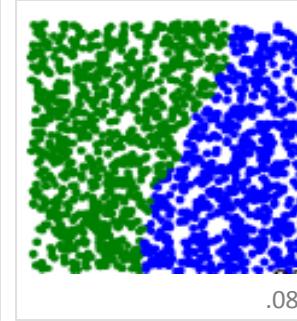
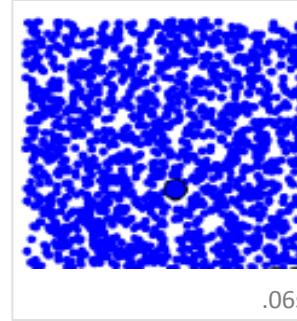
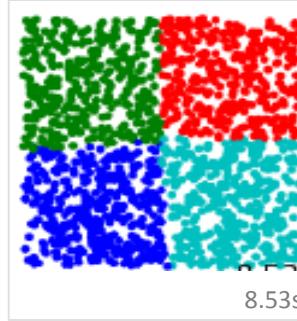
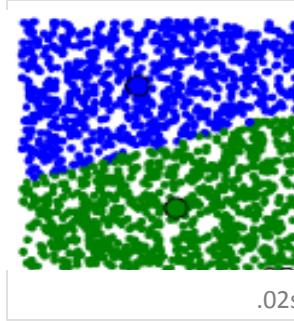
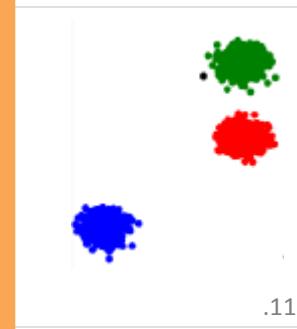
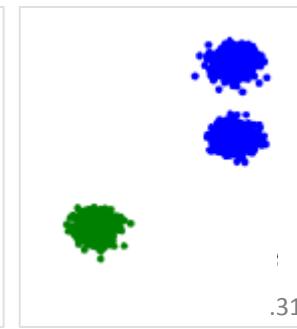
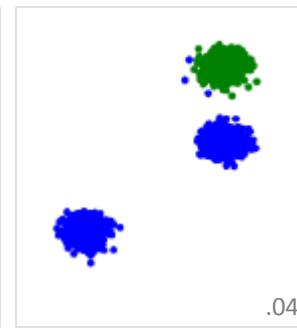
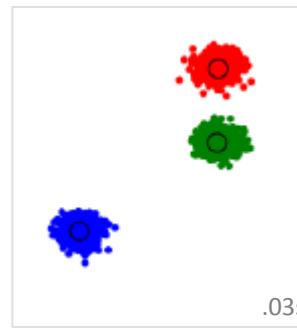
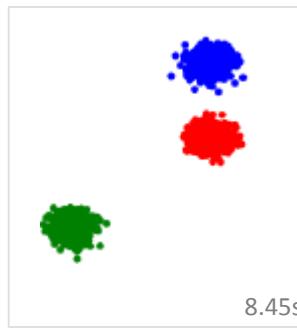
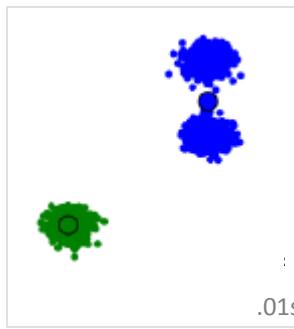
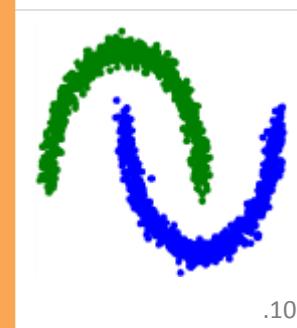
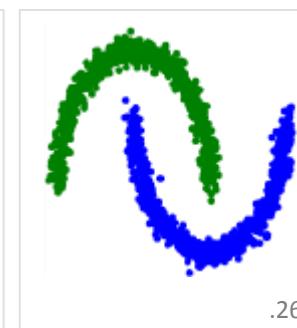
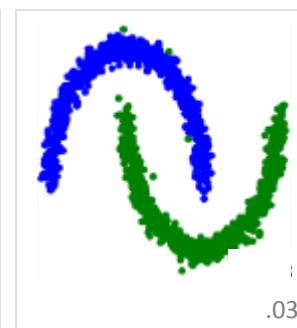
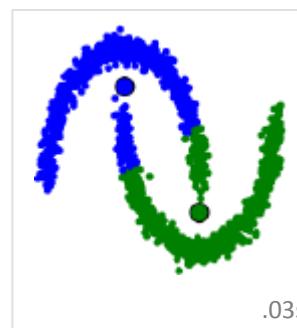
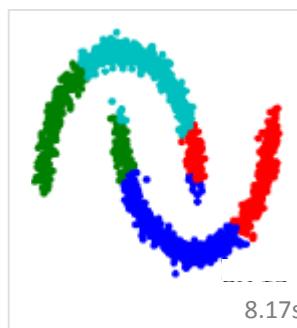
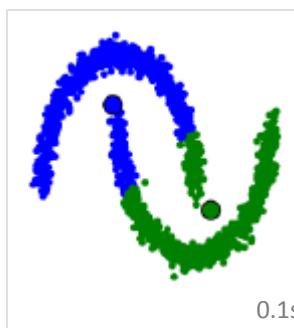
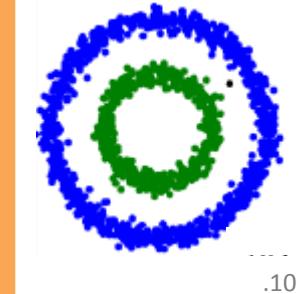
SpectralClustering



Ward



DBSCAN



Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-means	Number of clusters	Very large $n_{samples}$ medium $n_{clusters}$ with <i>MiniBatch code</i>	General purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	Damping, sample preferences	Not scalable with $n_{samples}$	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	Bandwidth	Not scalable with $n_{samples}$	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	Number of clusters	Medium $n_{samples}$, small $n_{clusters}$	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Hierarchical clustering	Number of clusters	Large $n_{samples}$ and $n_{clusters}$	Many clusters, possibly connectivity constraints	Distances between points
DBSCAN	Neighborhood size	Very large $n_{samples}$, medium $n_{clusters}$	Non-flat geometry, uneven cluster sizes	Distances between nearest points

Mean-shift

Slow.

You DON'T have to guess k or anything. Fire & forget.

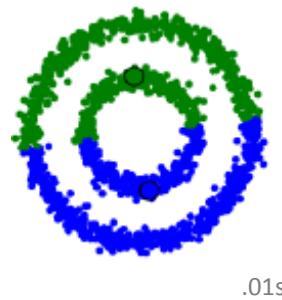
Can find uneven cluster sizes.

Slow with a lot of data, but lots of clusters no problem

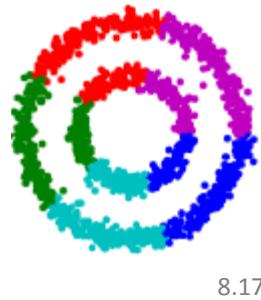
Doesn't handle weird shapes well.

Euclidean distance only.

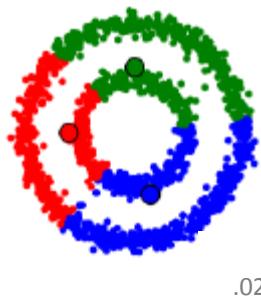
MiniBatchKMeans



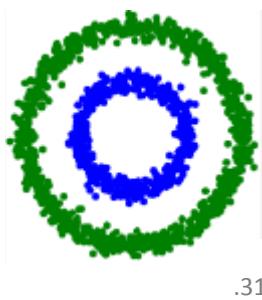
AffinityPropagation



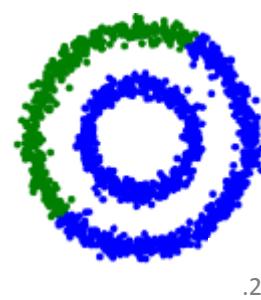
MeanShift



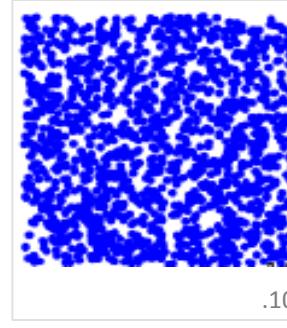
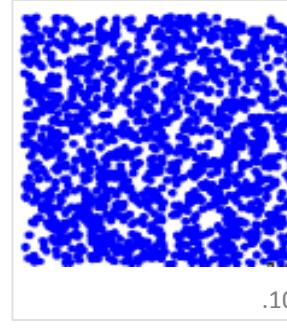
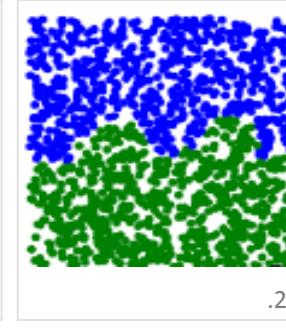
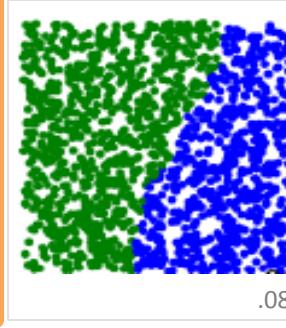
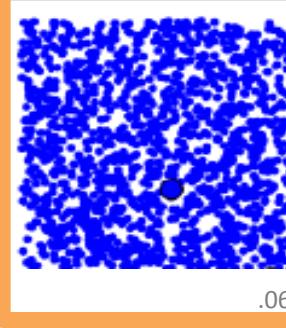
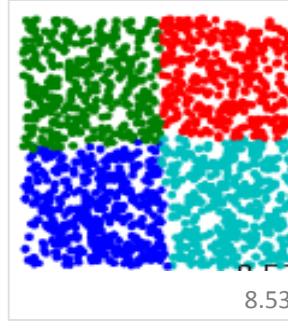
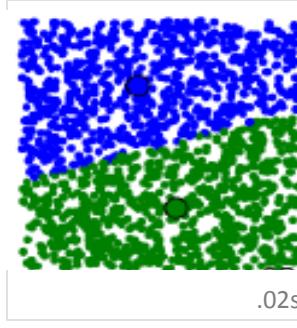
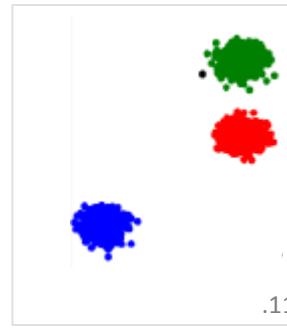
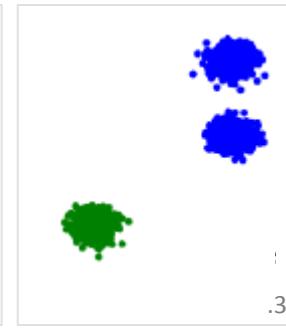
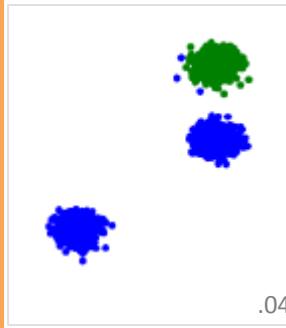
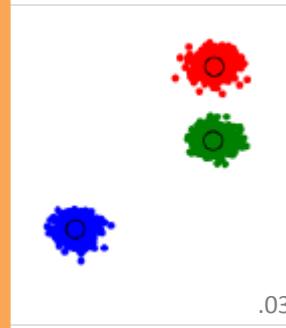
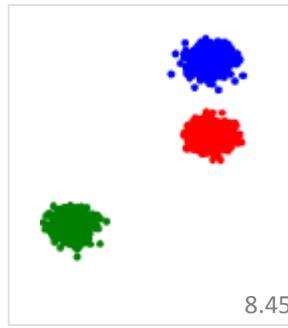
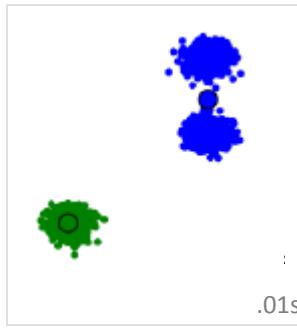
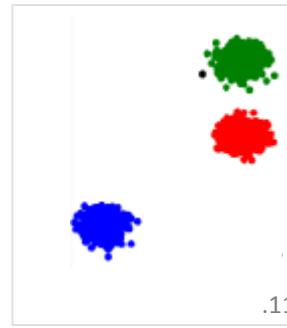
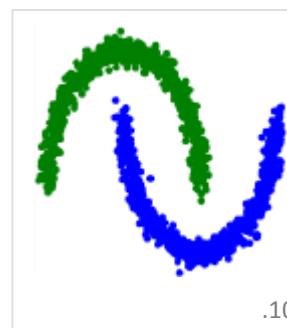
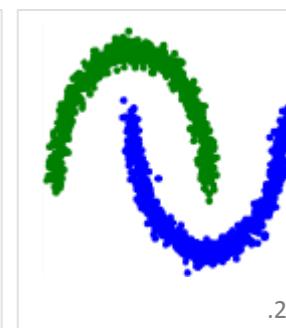
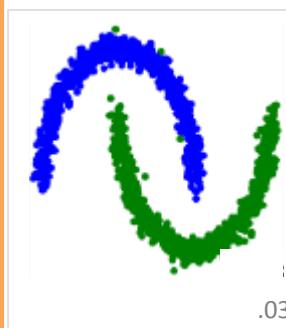
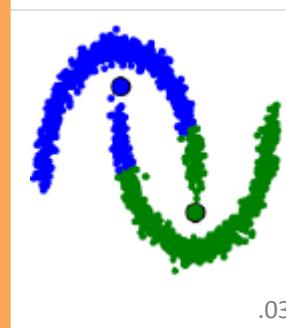
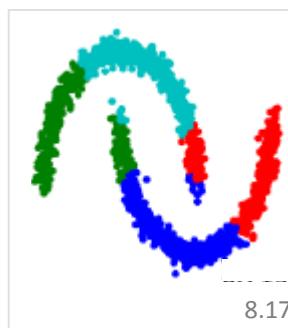
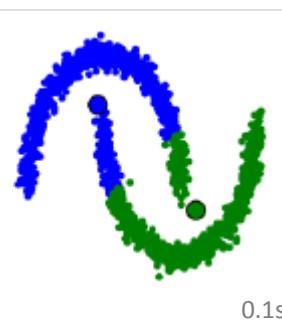
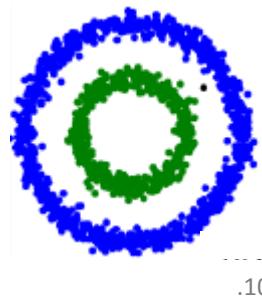
SpectralClustering



Ward



DBSCAN



Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-means	Number of clusters	Very large $n_{samples}$ medium $n_{clusters}$ with <i>MiniBatch code</i>	General purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	Damping, sample preferences	Not scalable with $n_{samples}$	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	Bandwidth	Not scalable with $n_{samples}$	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	Number of clusters	Medium $n_{samples}$, small $n_{clusters}$	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Hierarchical clustering	Number of clusters	Large $n_{samples}$ and $n_{clusters}$	Many clusters, possibly connectivity constraints	Distances between points
DBSCAN	Neighborhood size	Very large $n_{samples}$, medium $n_{clusters}$	Non-flat geometry, uneven cluster sizes	Distances between nearest points

Spectral clustering

Not fast, but not the slowest either.

Works best if distances are sparse.

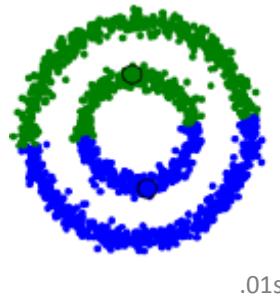
Tends to find even sized clusters.

Good with only a few clusters, rubbish with many.

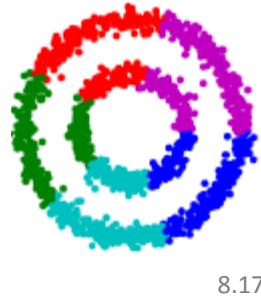
Can handle **weirdest connectivity shapes** (like concentric circles)

Euclidean distance only.

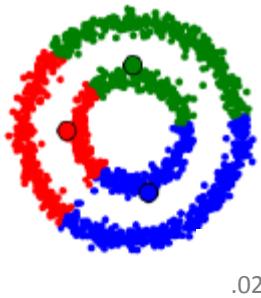
MiniBatchKMeans



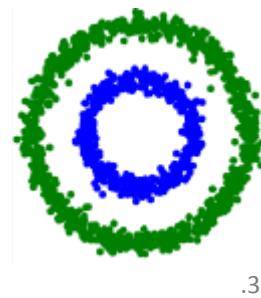
AffinityPropagation



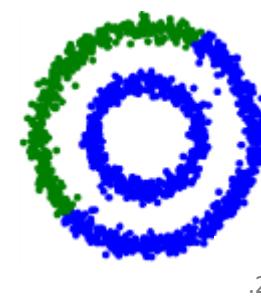
MeanShift



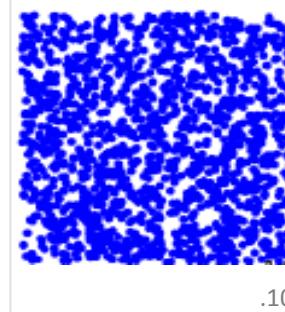
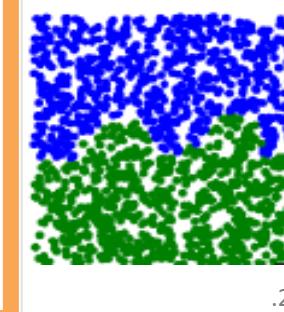
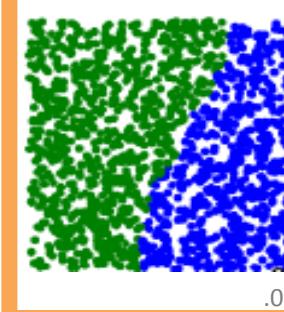
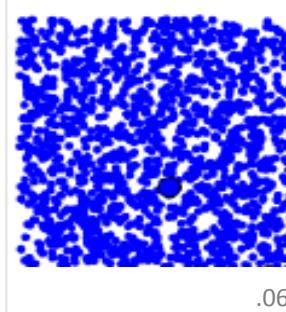
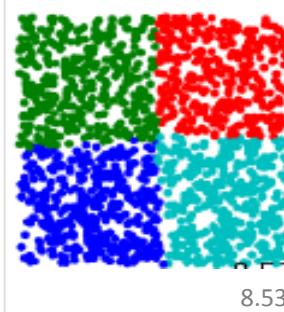
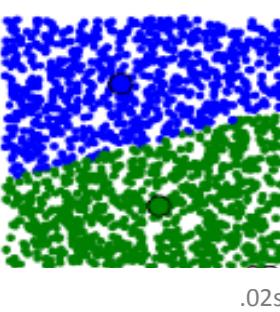
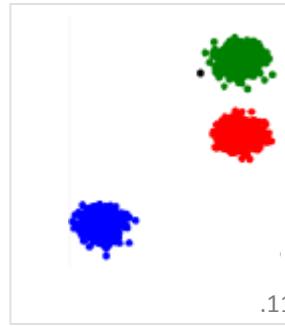
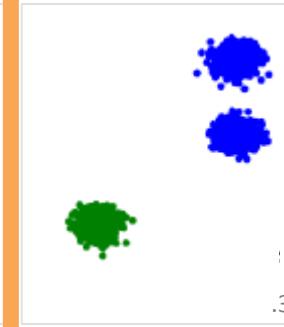
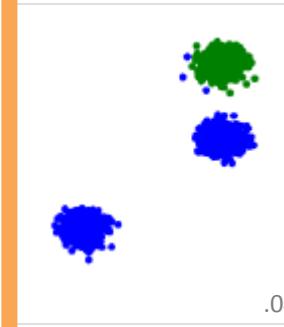
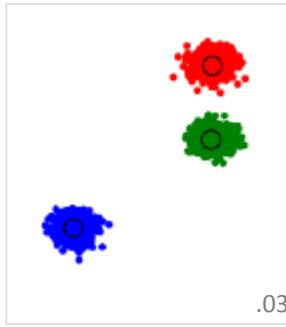
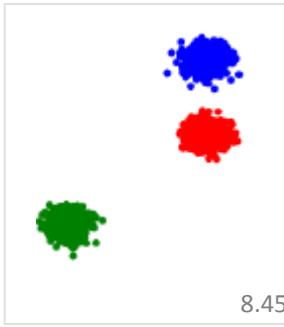
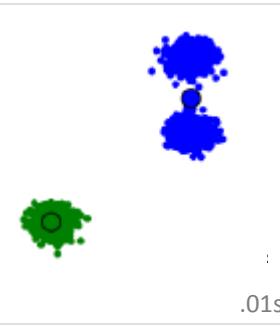
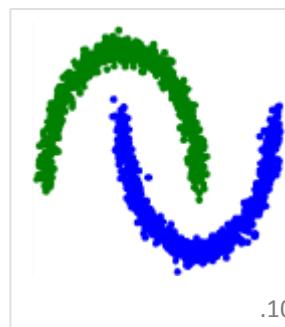
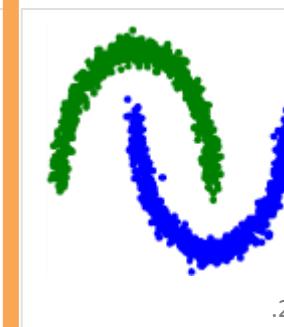
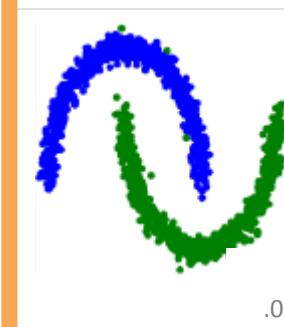
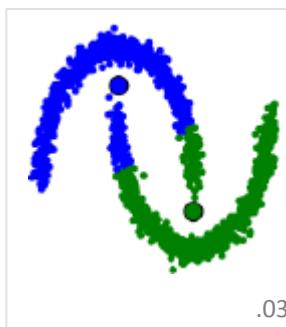
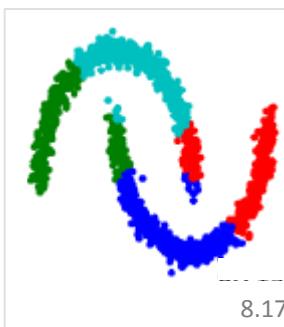
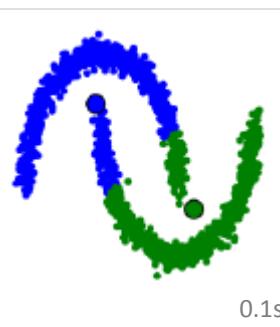
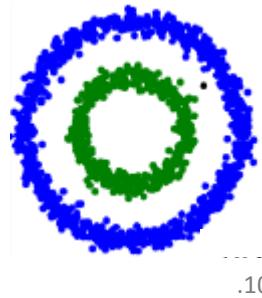
SpectralClustering



Ward



DBSCAN

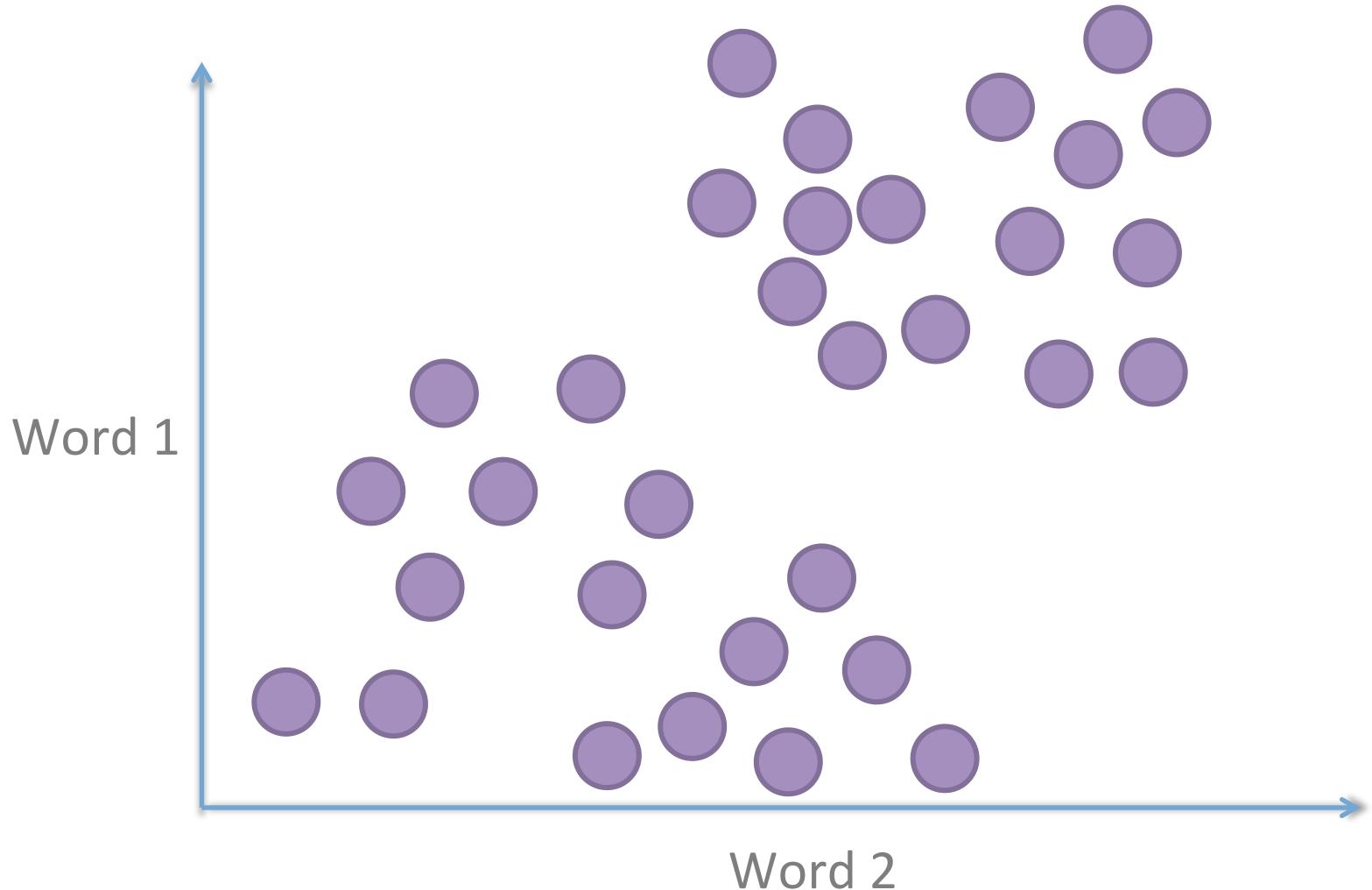


What does clustering
with cosine distance
look like?



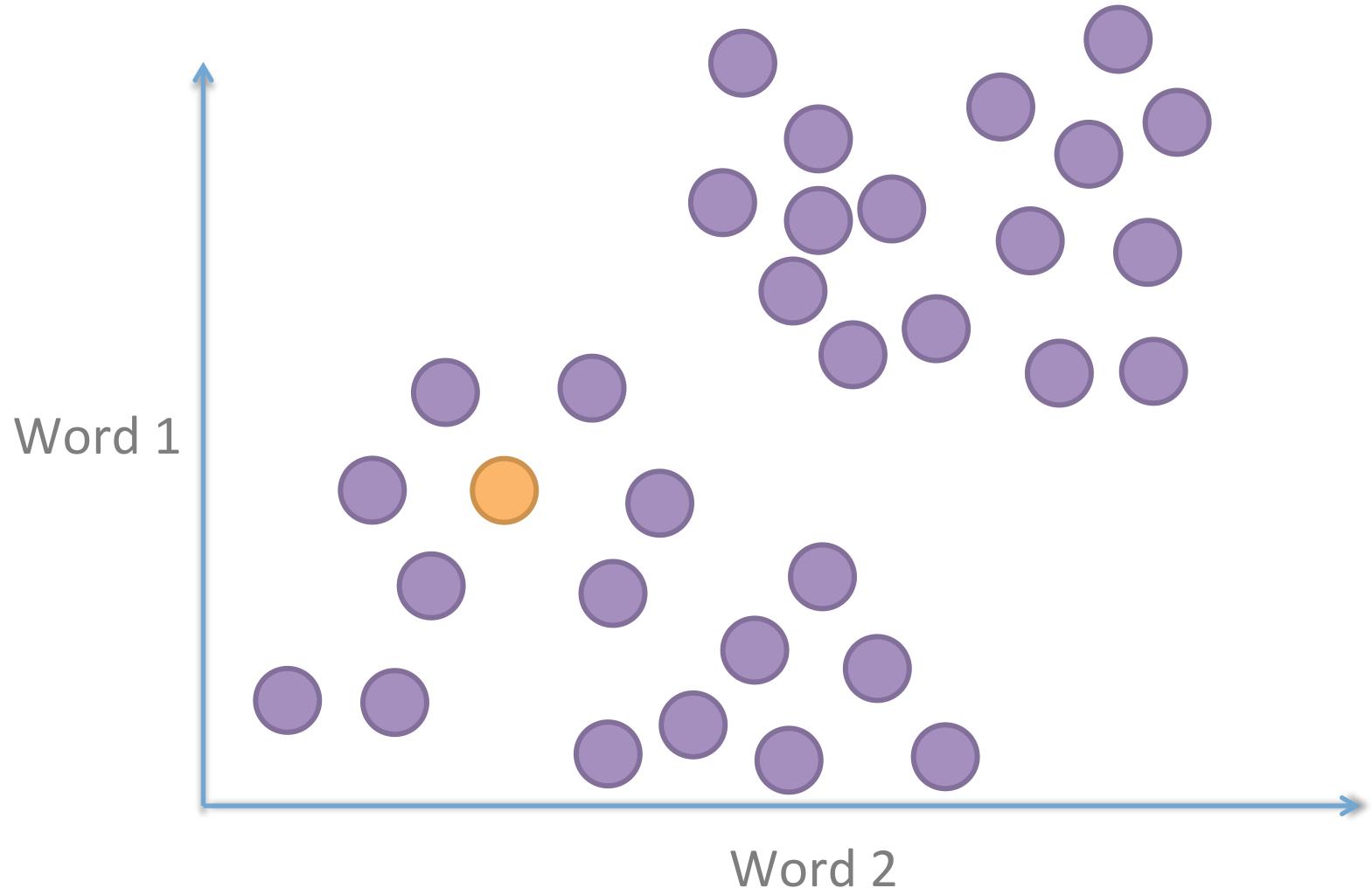
DBSCAN

With cosine distance



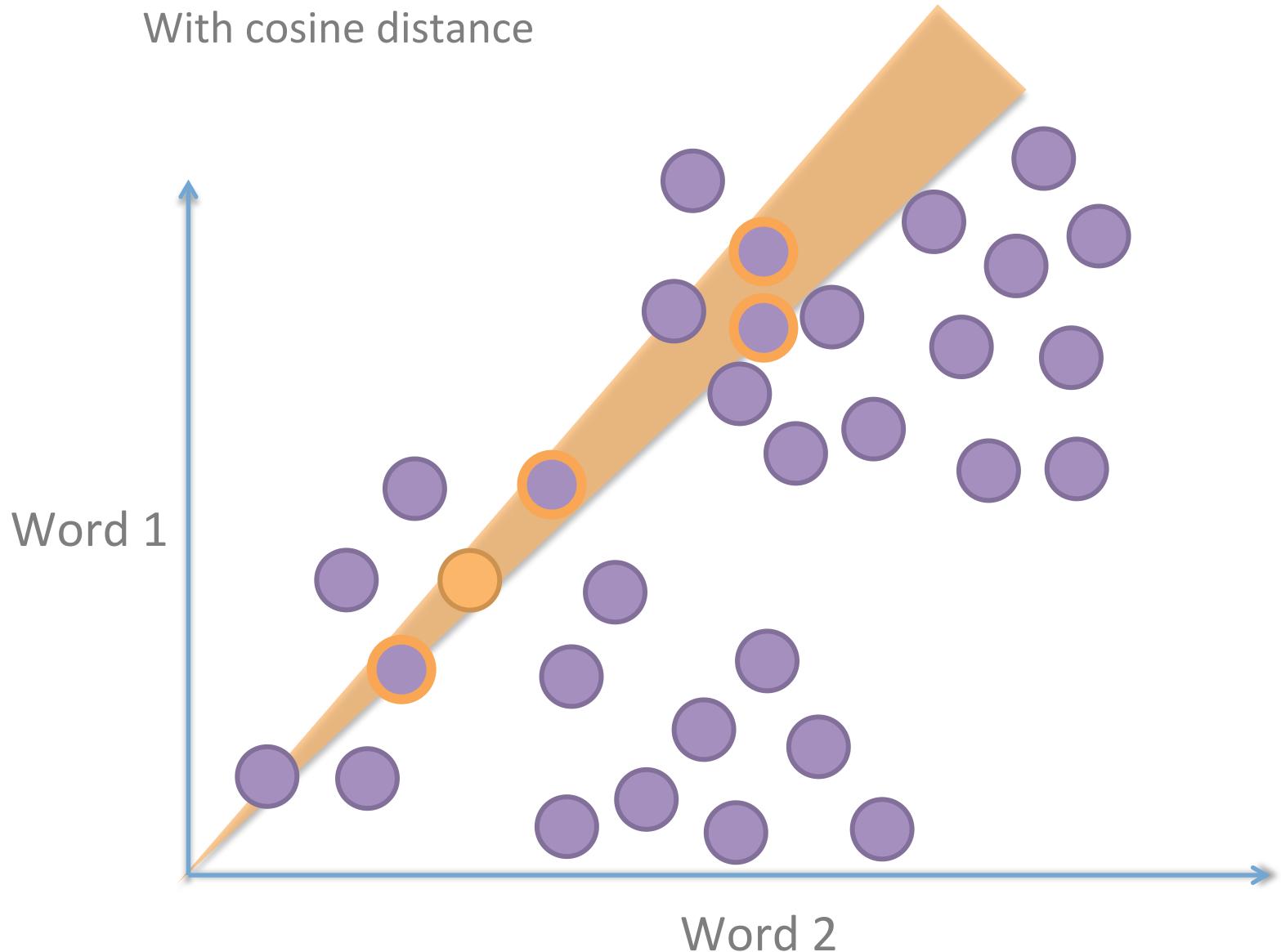
DBSCAN

With cosine distance

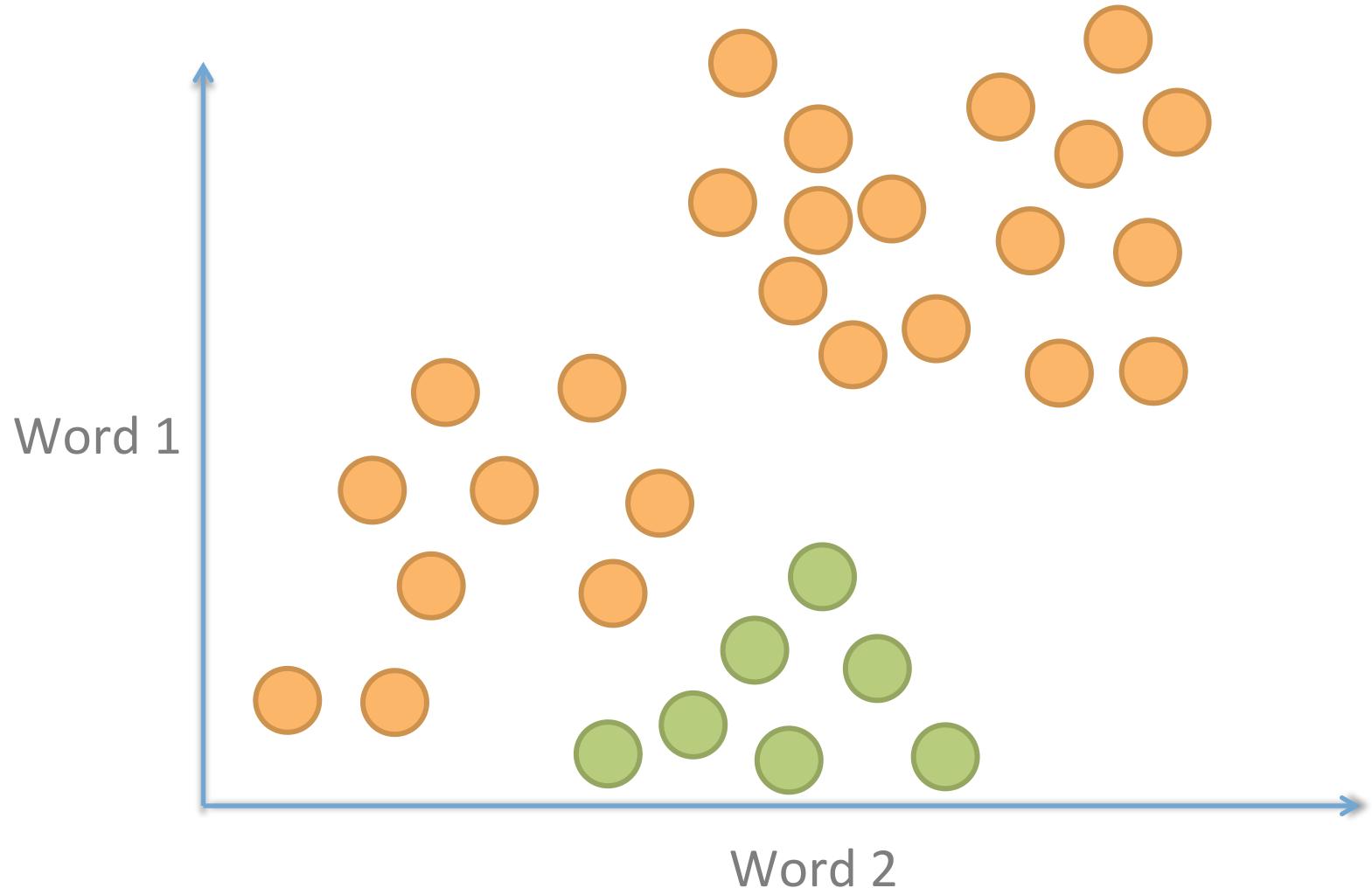


DBSCAN

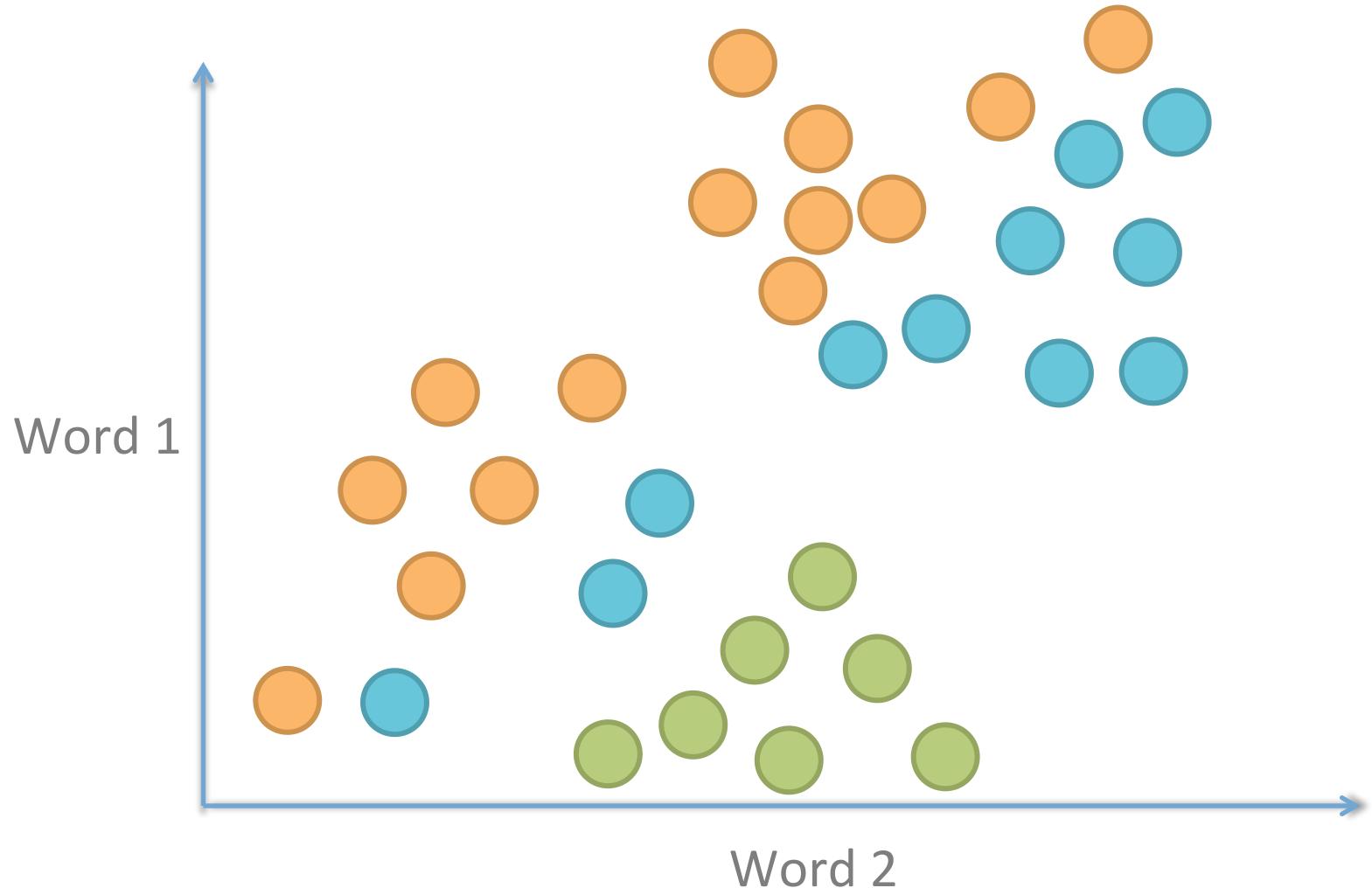
With cosine distance



Potential clusters with cosine distance



Potential clusters with cosine distance



Potential clusters with cosine distance

