Instance-based learning

Lecture 7

Important:

Similarity and distance metrics

Memory-based reasoning

Reasoning from experience: ability to recognize similar examples from the past

- This person is from Australia, since it speaks like other Australians I have met before
- This mushroom is poisonous since it seems similar to

Amanita muscaria

Memory-based reasoning

Reasoning from experience:

- The most effective treatment for a given patient is the treatment which was effective in similar cases, on similar patients
- The next customers who are likely to respond to a promotion are similar to the previous customers who have responded

Classification by similarity

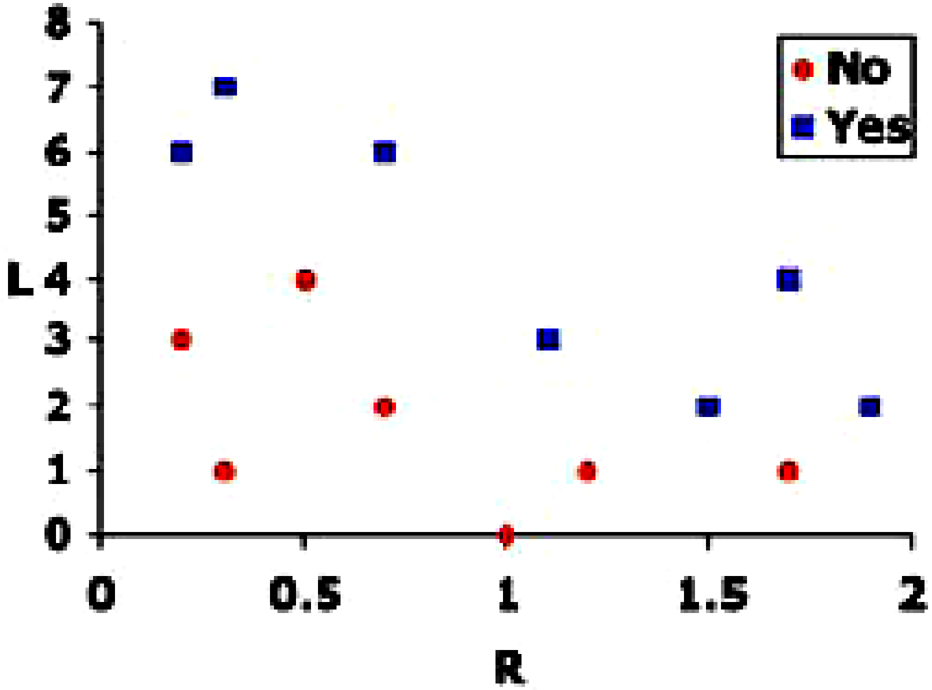
“ If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck. ”

New classifier: nearest neighbor

- Remember all your data
- When a new record comes:
  - Find the most similar old data point (the nearest neighbor)
  - Return the answer associated with it

Predicting Bankruptcy

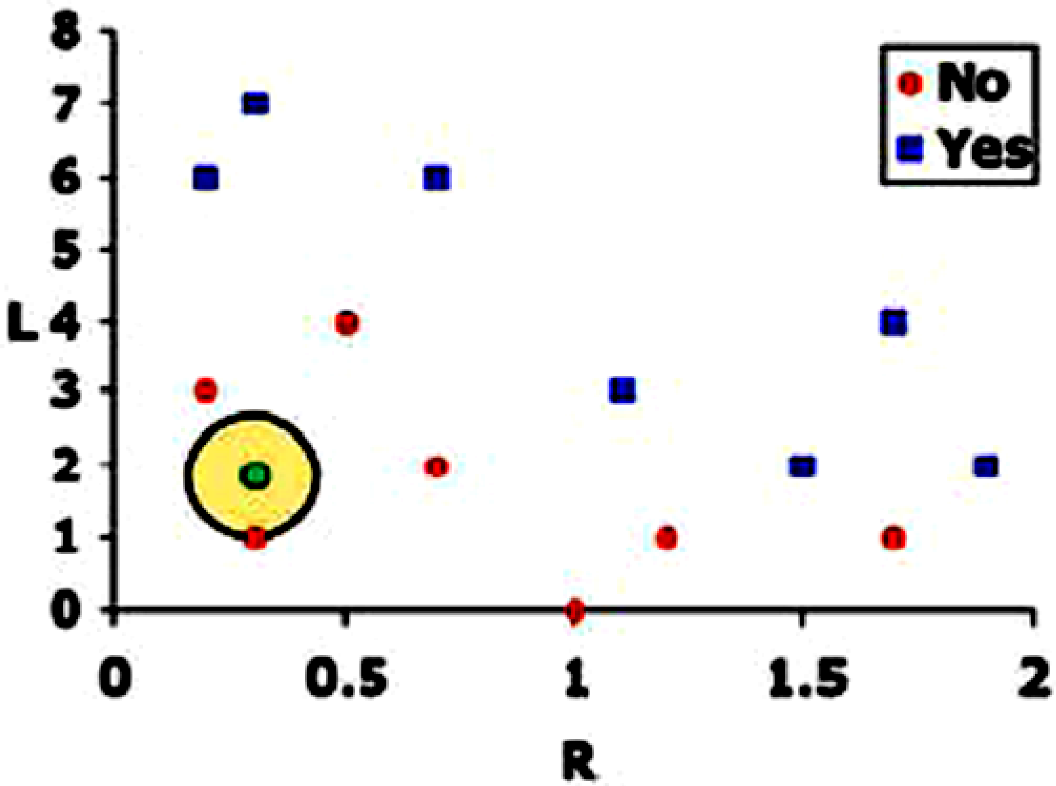
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1.0	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



L: #late payments / year
R: expenses / income

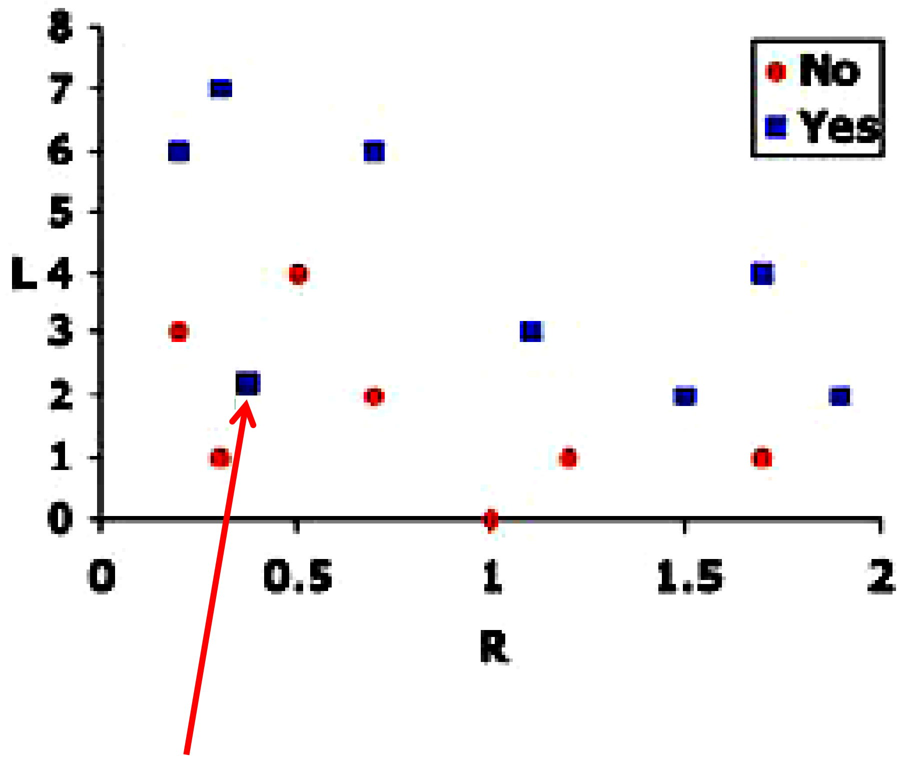
Predicting Bankruptcy

- Now, let's say we have a new person with R equal to 0.3 and L equal to 2.



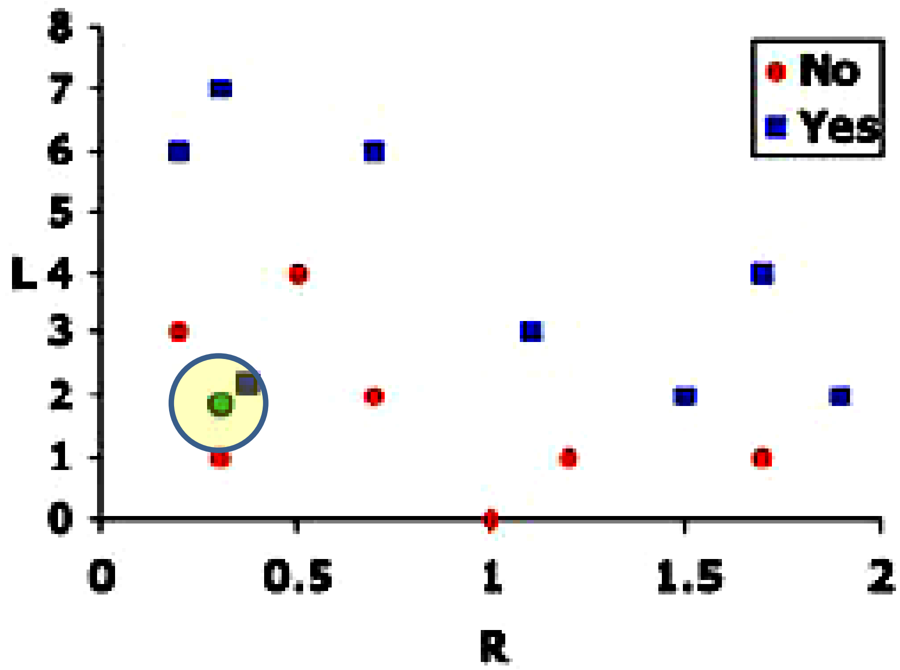
And so our answer would be "no".

Noise



Someone with an apparently healthy financial record goes bankrupt.

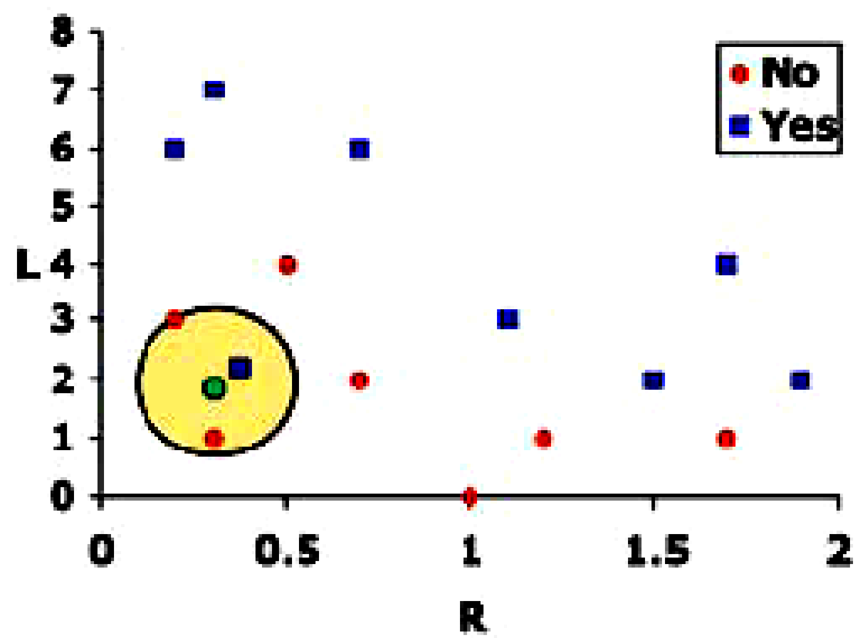
Noise



Classification goes wrong.

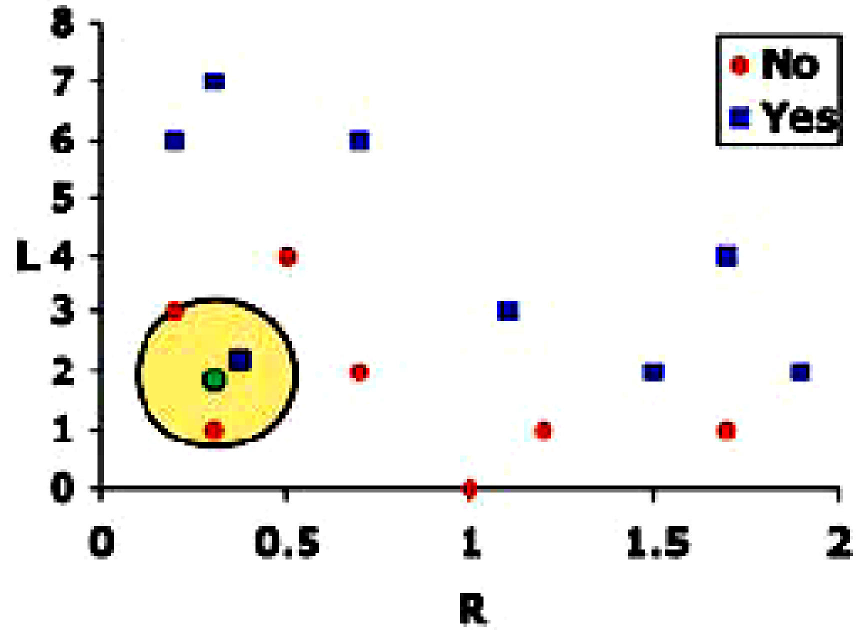
Remedy: K-Nearest Neighbors

- *K*-nearest neighbor algorithm:
  - Just like the old algorithm, except that when we get a query, we'll search for the *k* closest points to the query points.



Remedy: K-Nearest Neighbors

- Combine the output from k neighbors:
  - In this case, we've chosen *k* to be 3.
  - The three closest points consist of two "no"s and a "yes", so we combine them using the majority voting



K-Nearest Neighbor: design issues

- Choosing the *distance* function
- Choosing optimal *number of neighbors*
- Choosing the *combination function* to combine neighbors’ class

K-Nearest Neighbor: design issues

- Choosing the distance function
- Choosing optimal number of neighbors
- Choosing the combination function to combine neighbors’ class

Distance function: requirements

1. Well-defined: the distance between two instances is always defined and is a non-negative real number: $d(A,B) \geq 0$
2. Identity: The distance from one instance to itself is always zero: $d(A,A)=0$
3. Commutativity: Direction does not change the distance, so the distance from A to B is the same as the distance from B to A (no one-way roads)
4. Triangle inequality: Visiting an intermediate point C on the way from A to B never shortens the distance, so $d(A,B) \leq d(A,C)+d(C,B)$

Distance function: requirements

- Well-defined: $d(A,B) \geq 0$

Every record has a neighbor somewhere in the database

Distance function: requirements

- Identity: d(A,A)=0

The most similar record to a given record is the original record itself

Distance function: requirements

- Commutativity: $d(A,B)=d(B,A)$
- Triangle inequality: $d(A,B) \le d(A,C)+d(C,B)$

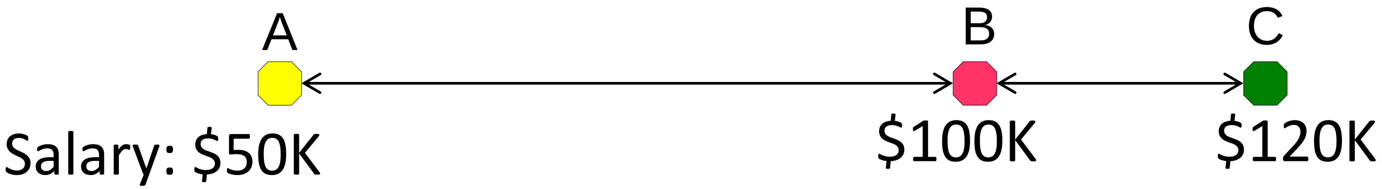
Make nearest-neighbors local and well-behaved: adding a new record into a database will not bring an existing record any closer

Building a distance function: one field at a time

- Distance between 2 points in geometry is well defined
- What is *the distance between instances*?

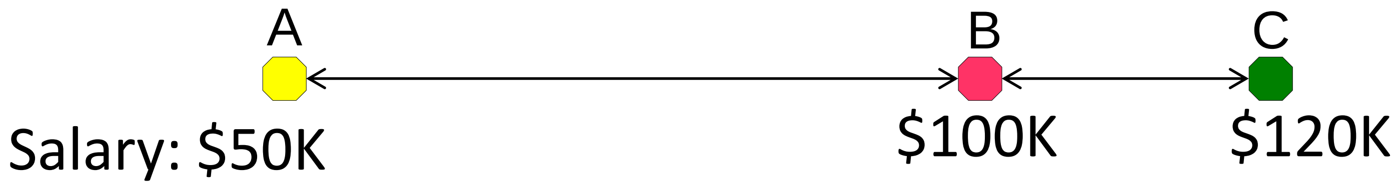
- The answer is: find distance for each attribute separately and combine

Distance between numeric values in one dimension (for a single attribute)



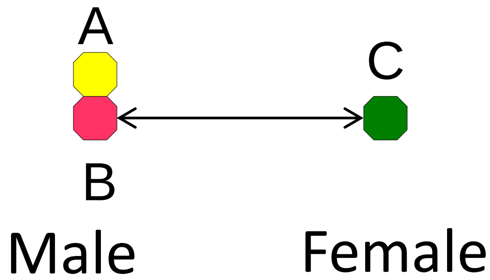
- D(A,B)=?
- D(B,C)=?
- D(A,C)=?

Distance between numeric values in one dimension (for a single attribute)



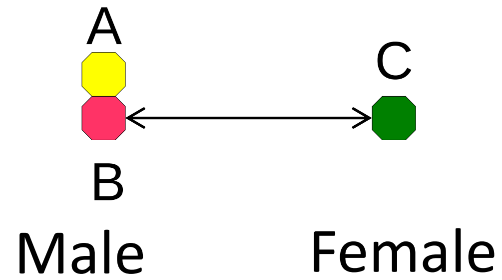
- D(A,B)=|50,000-100,000|=50,000
- D(B,C)=|100,000-120,000|=20,000
- D(A,C)=|50,000-120,000|=70,000

Distance between categorical (binary) values in one dimension



- D(A,B)=?
- D(B,C)=?
- D(A,C)=?

Distance between categorical (binary) values in one dimension



- D(A,B)=0
- D(B,C)=100%
- D(A,C)=100%

Distance for other data types: American ZIP codes

- A 5-digit American ZIP code cannot be treated as numeric
- A ZIP code does encode location information
  – First 3 digits represent a postal zone: (100,101,102 – Manhattan)
  – Zip codes increase from East to West: begin with 0 in New England and with 9 on the west coast

Task: Design customer distance function for ZIP codes

Distance for other data types: American ZIP codes

- Design customer distance for ZIP codes:
  - D(A,B)=0 if A=B
  - D(A,B)=0.1 if 3 first digits are identical (20008 – 20015)
  - D(A,B)=0.5 if the first digits are identical (95050 – 98125)
  - D(A,B)=1.0 if the first digits differ

Note: Real distance can be retrieved as the longitude and the latitude from the Postal Codes database

Meaningful choice

The choice of a distance metric depends on context, and is not mechanical

Combined distance metrics for multiple dimensions

- Manhattan distance
- Euclidean distance
- Pearson correlation
- Cosine
- Jaccard index

Combined distance metrics for multiple dimensions

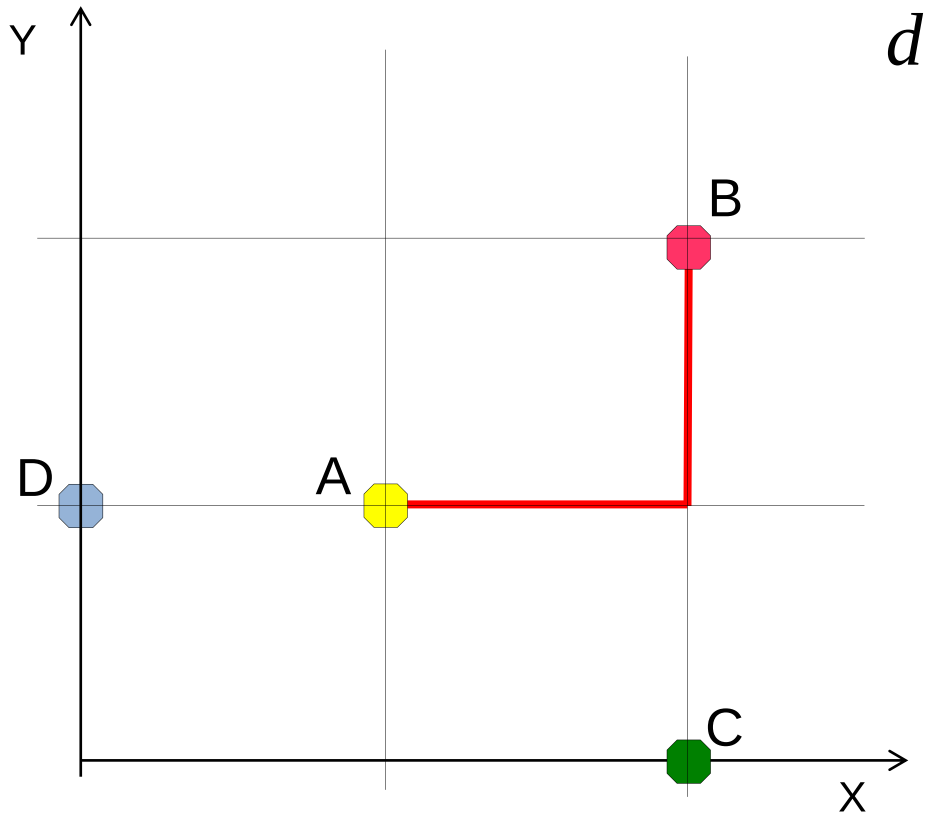
- Manhattan distance
- Euclidean distance
- Pearson correlation

The examples are in 2 dimensions, but the definitions are for *n* dimensions (attributes)

Taxicab (Manhattan) distance

- The sum of the distances across each dimension
- The distance between two points when only movement along the grid lines is allowed

Distance metrics. Manhattan distance



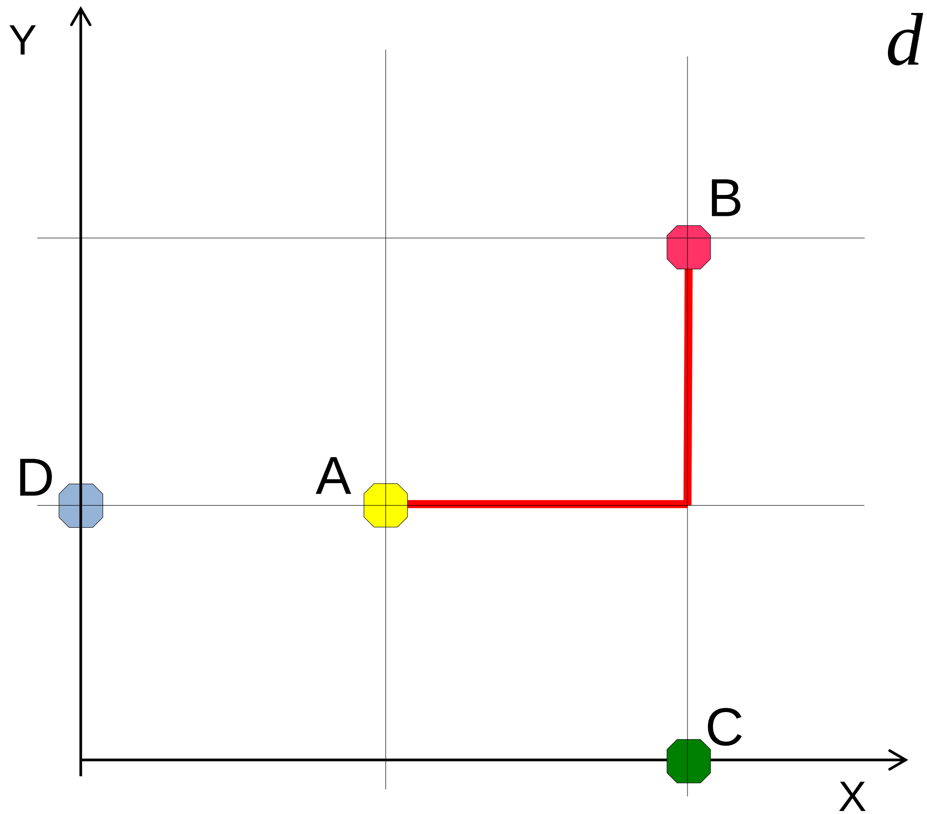
$$d\ (A,B) = d(A,B)_X + d(A,B)_Y$$

$$d\ (A,B) = ?$$

$$d\ (B,C) = ?$$

$$d\ (C,D) = ?$$

Distance metrics. Manhattan distance



$$d\,(A,B) = d(A,B)_X + d(A,B)_Y$$

$$d\,(A,B) = 2$$
$$d\,(B,C) = 2$$

$$d\,(C,D) = 3$$

For *N* dimensions:

$$d(A,B) = \sum_{i=1}^{N} |\, A_i - B_i \,|$$

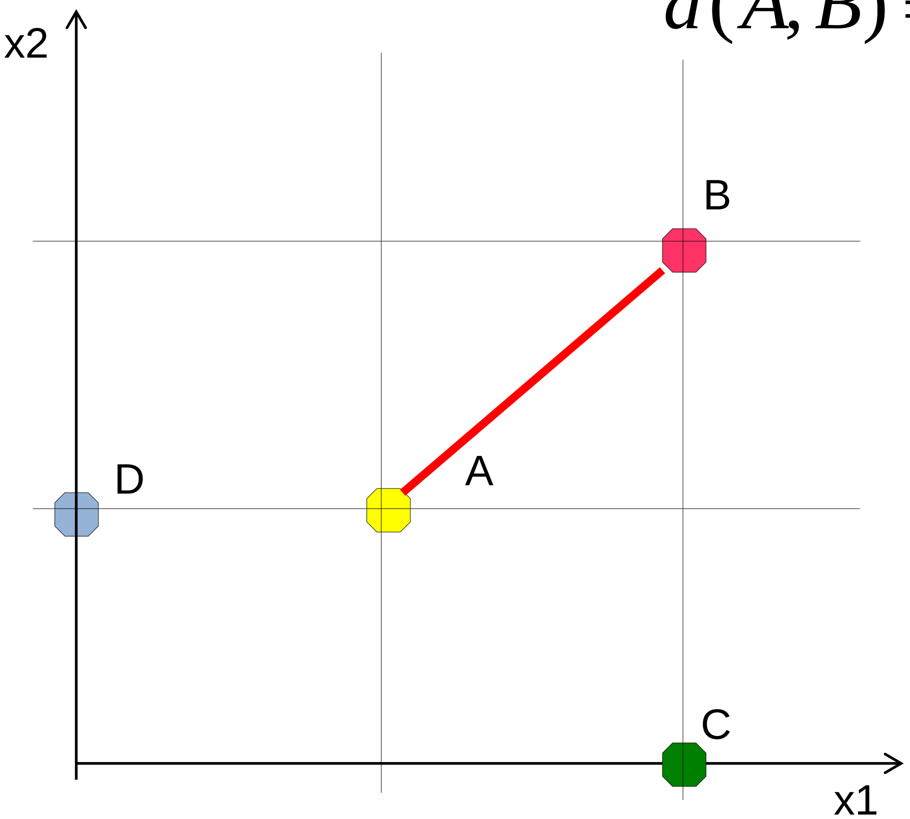
Euclidean distance

- The “ordinary” distance between two points that one would measure with the ruler

- Computed from coordinates using Pythagorean theorem

Distance metrics.
Euclidean distance

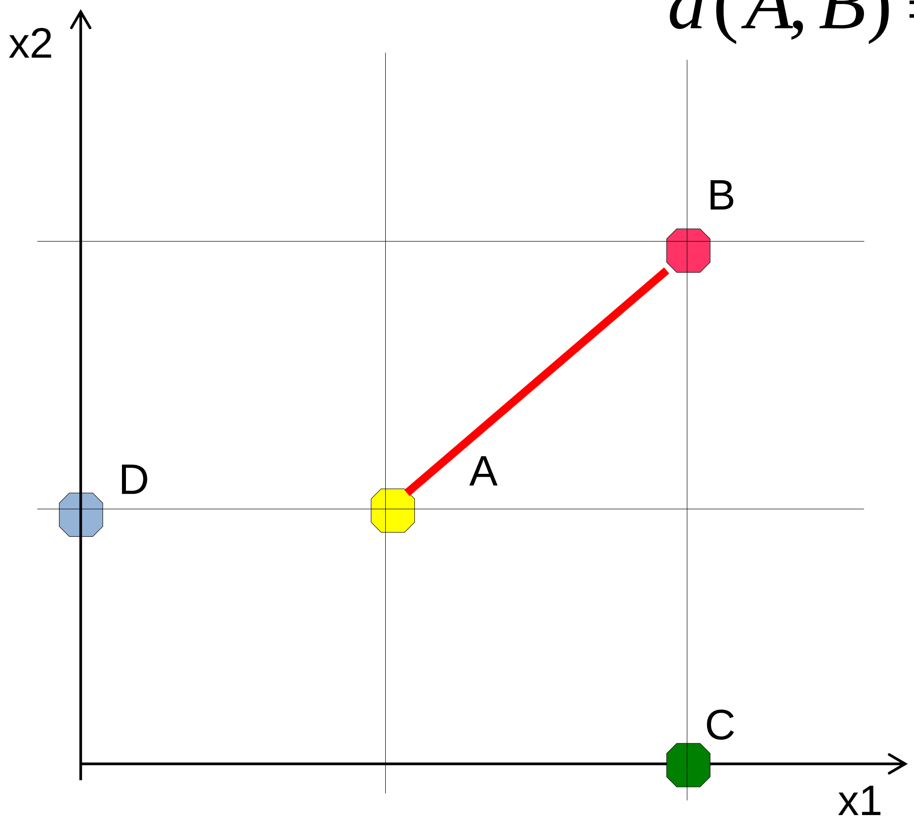
$$d(A,B)=\sqrt{|A_X-B_X|^2+|A_Y-B_Y|^2}$$



$d(A,B)=?$
$d\ (B,C)=?$
$d(C,D)=?$

Distance metrics. Euclidean distance

$$d(A,B) = \sqrt{|A_X - B_X|^2 + |A_Y - B_Y|^2}$$



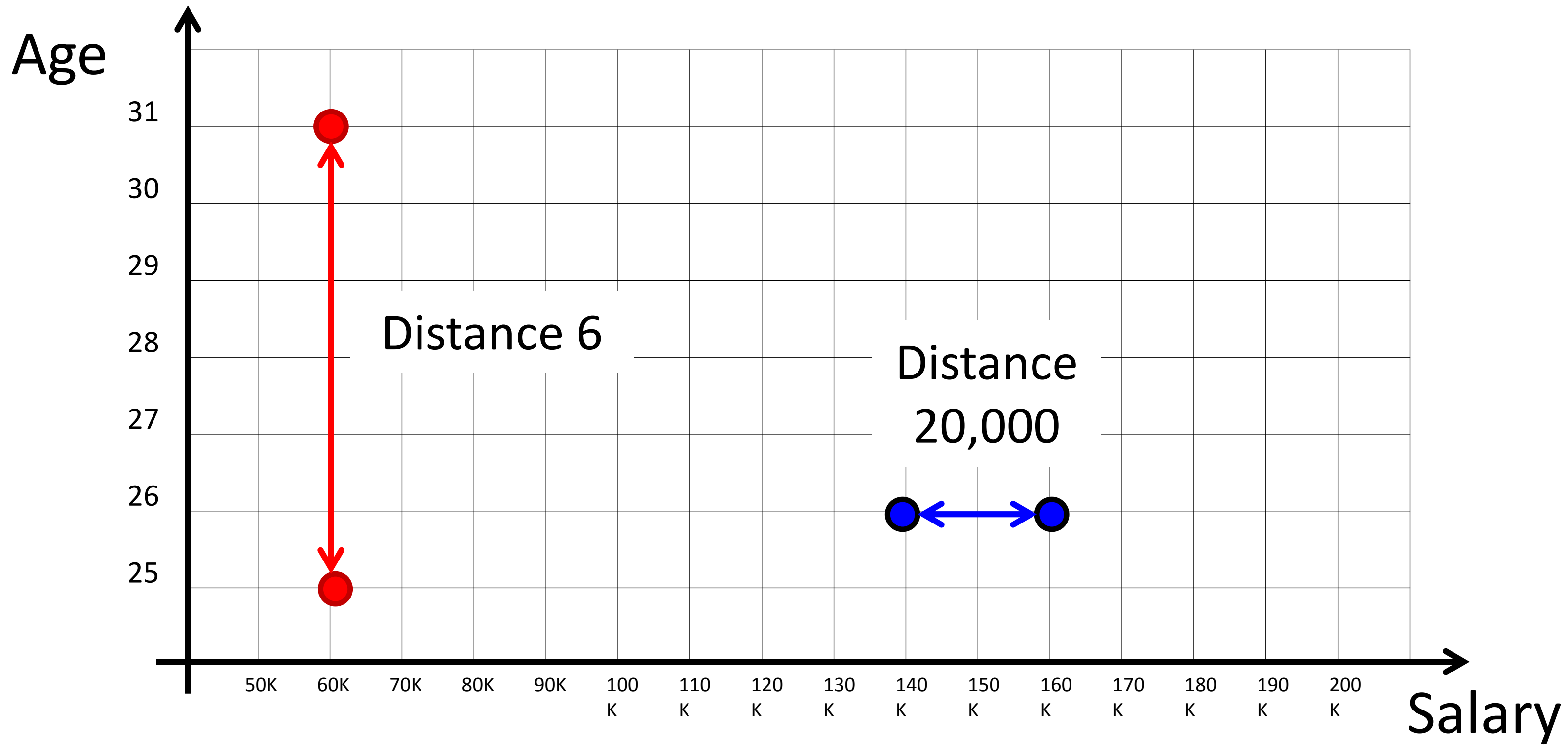
d(A,B)=sqrt(2)
d (B,C)=2
d(C,D)=sqrt(5)

For N dimensions:

$$d(A,B) = \sqrt{\sum_{i=1}^{N} |A_i - B_i|^2}$$

Scaling problem

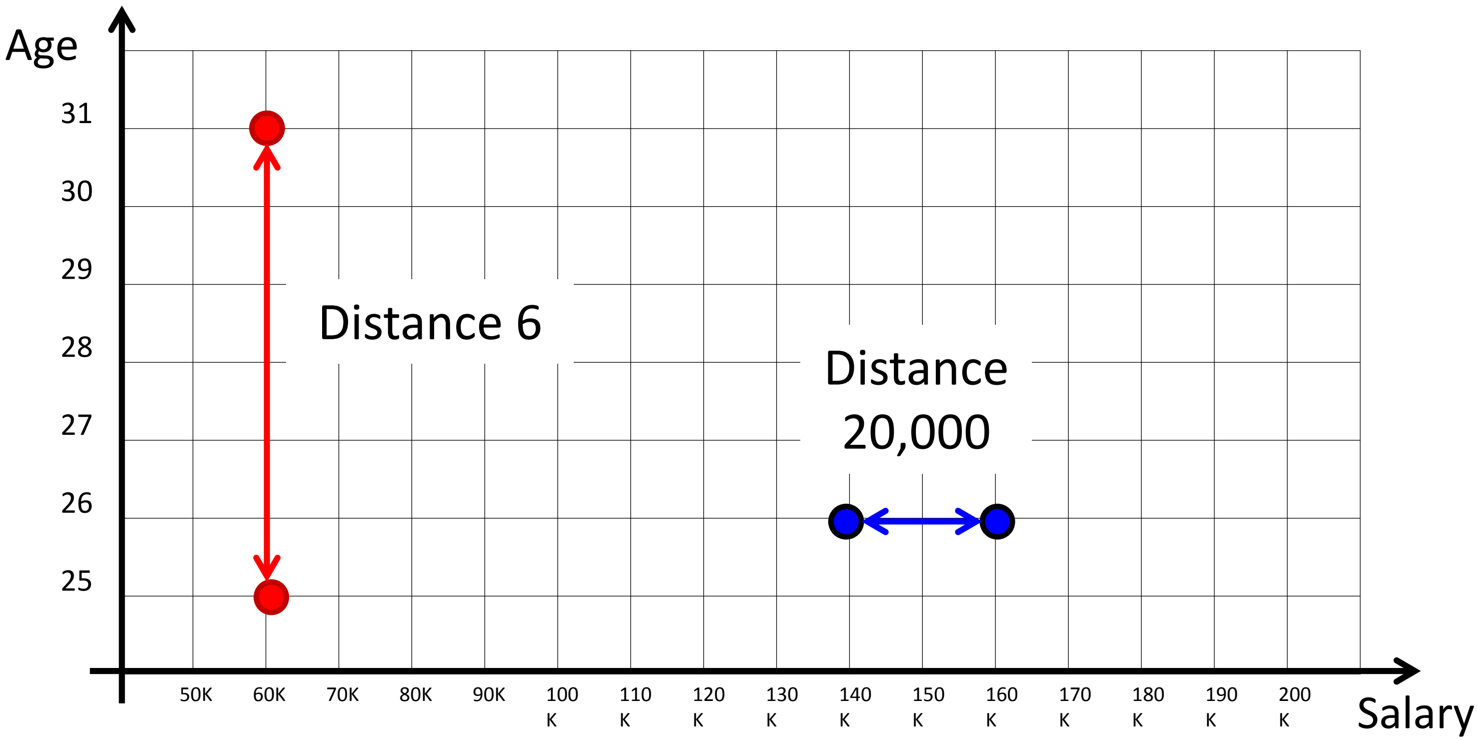
- Heterogeneous variables cause distance problems
- We need a way to normalize actual values so it makes sense to consider them all in the same space



Scaling dimensions

- Rescale the values to put all of the features on about equal footing:

$$a_i = \frac{v_i - \min(all\ v)}{\max(all\ v) - \min(all\ v)}$$

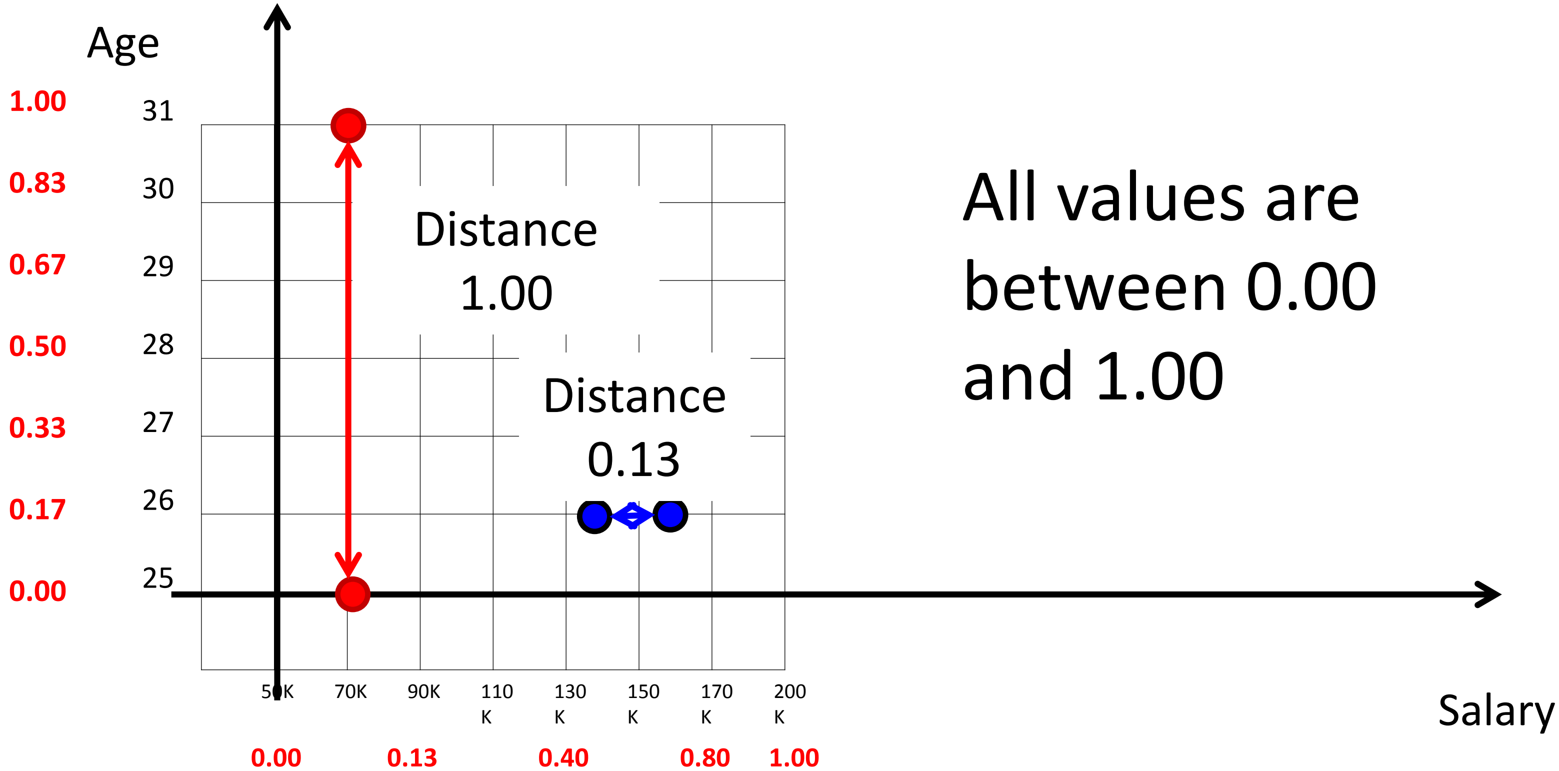


Scaling dimensions

$$a_i = \frac{v_i - \min(all\ v)}{\max(all\ v) - \min(all\ v)}$$

For Age: a_i=(v_i-25)/(31-25)

For Salary: a_i=(v_i-50,000)/(200,00-50,00)



All values are between 0.00 and 1.00

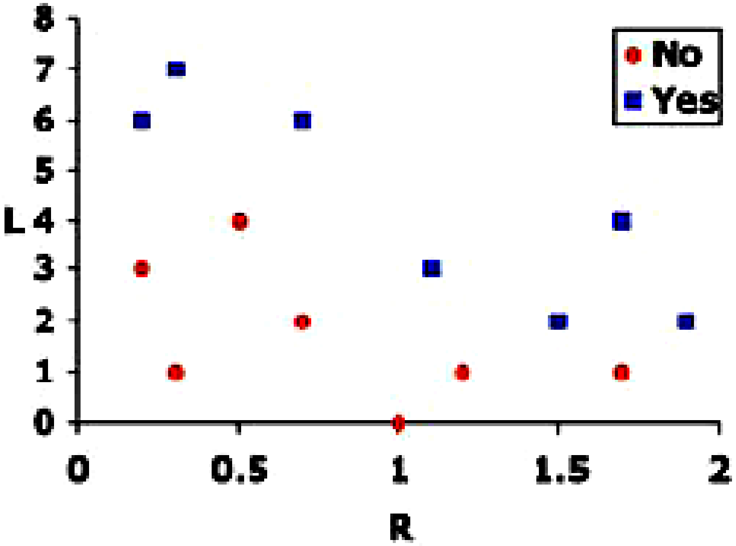
K-Nearest Neighbor: design issues

- Choosing the distance function
- Choosing optimal number of neighbors
- Choosing the combination function to combine neighbors' class

How many neighbors?

- Vary *K* from 1 to N
- Use cross-validation to find optimal value of *K*

L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1.0	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



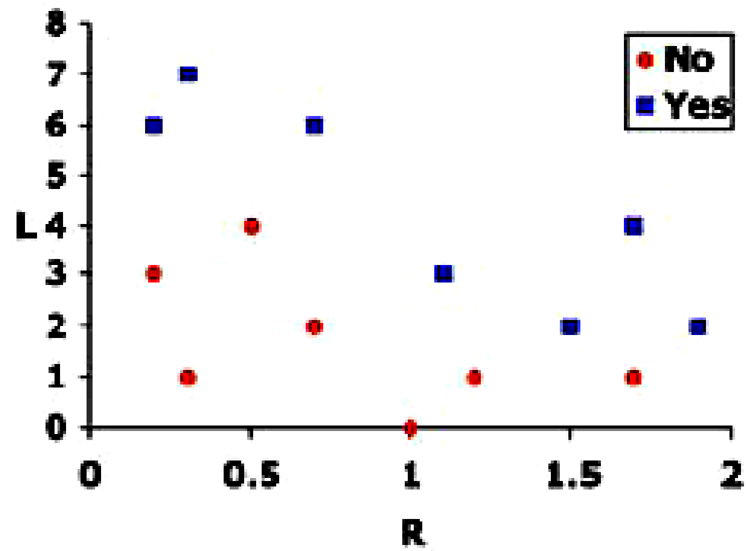
L: #late payments / year
R: expenses / income

Example:
Leave-one-out cross validation

K=1

Error rate: 5/14

	L	R	B
	3	0.2	No
	1	0.3	No
	4	0.5	No
	2	0.7	No
	0	1.0	No
	1	1.2	No
x	1	1.7	No
	6	0.2	Yes
	7	0.3	Yes
x	6	0.7	Yes
x	3	1.1	Yes
x	2	1.5	Yes
	4	1.7	Yes
x	2	1.9	Yes



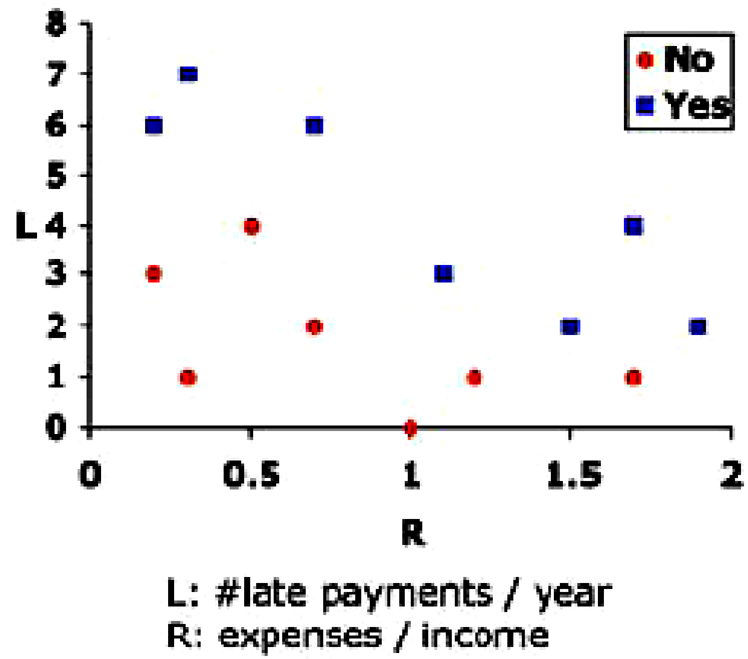
L: #late payments / year
R: expenses / income

Example:
Leave-one-out cross validation

K=3

Error rate: 3/14

	L	R	B
	3	0.2	No
	1	0.3	No
	4	0.5	No
	2	0.7	No
	0	1.0	No
x	1	1.2	No
x	1	1.7	No
	6	0.2	Yes
	7	0.3	Yes
	6	0.7	Yes
x	3	1.1	Yes
	2	1.5	Yes
	4	1.7	Yes
	2	1.9	Yes



Choose K for which the error rate is minimized

K-Nearest Neighbor: design issues

- Choosing the distance function
- Choosing optimal number of neighbors
- Choosing the combination function to combine neighbors’ class

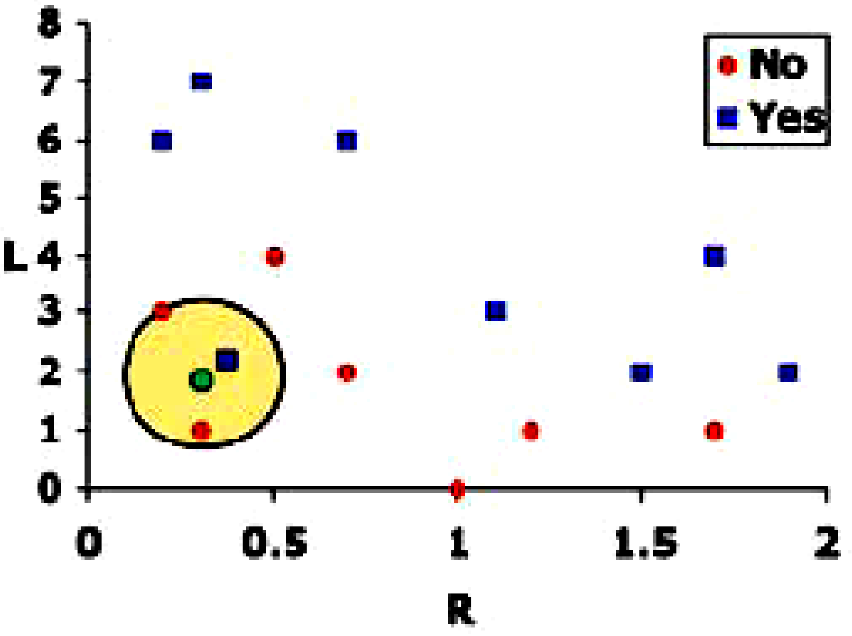
The combination function: asking neighbors for answer

- Majority voting
- Weighted voting

The combination function: democracy

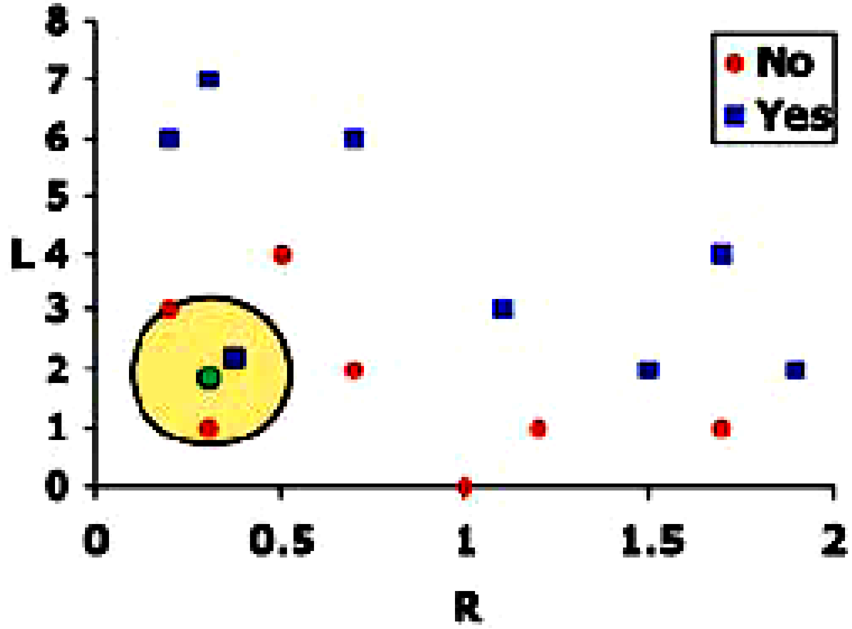
The basic approach:
 each neighbor casts its vote for its own class:

 Predicted class is “No”



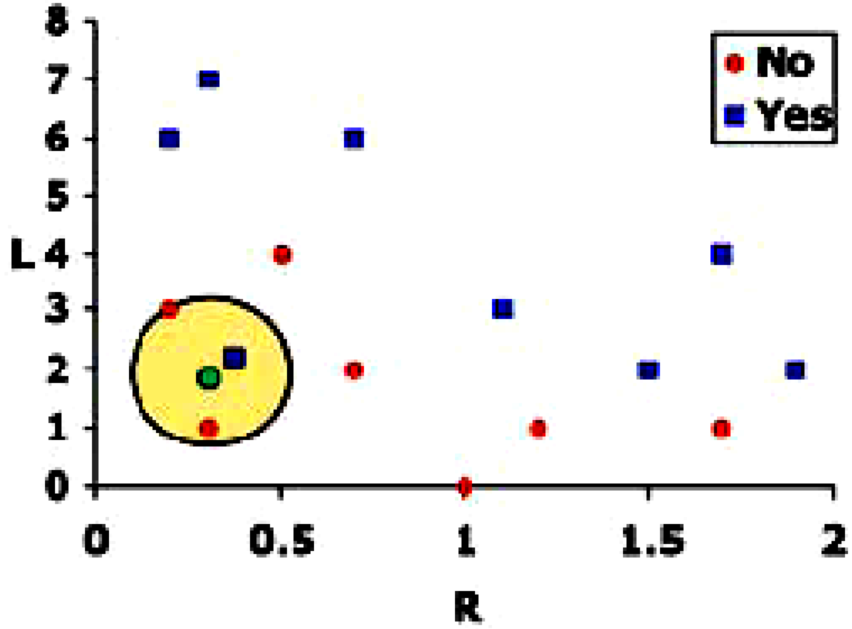
The combination function: democracy

For a binary class use odd number of neighbors to avoid ties



The combination function: democracy

The proportion of votes can be used as a probability that a new instance belongs to the majority class



The combination function: weighted voting

- The neighbors are not all created equal – more like shareholder democracy
- The size of the vote is inversely proportional to the distance from a new record, so closer neighbors have stronger votes than neighbors farther away do

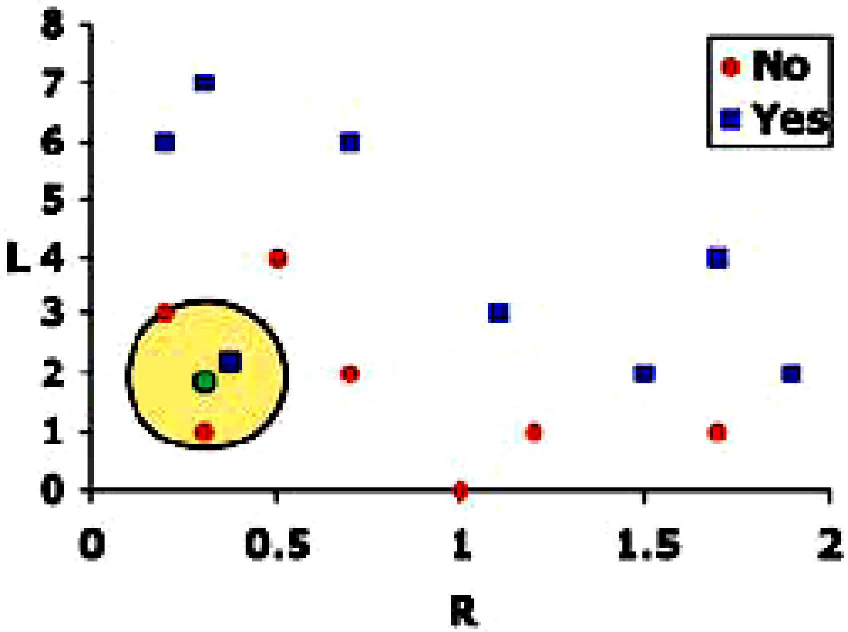
---

To prevent problems when the distance might be 0, it is common to add 1 to the distance before scaling and before taking an inverse

The combination function: weighted voting

1/0.5 Yes+1/1 No + 1/1.5 No=2 Yes + 1.7 No = Yes

The closets neighbor outweighs the majority class



K-NN algorithm. Summary

- The training set ***is a*** model
- Advantages:
  - No need to build a model
  - Adding new records – no need to rebuild the model
  - Interpretable: we always know why– we know all the neighbors
  - Good in predicting numeric values
  - More flexible (non-rectilinear) decision boundaries
- Disadvantages:
  - The query is computationally expensive

K-NN Algorithm. Time and space

- Learning is fast
  - We just have to remember the training data.
- Space is *N*.

- What takes longer is answering a query: If we do it naïvely, we have to, for each point in our training set (and there are *N* of them) compute the distance to the query point (which takes about *M* computations, since there are *M* features to compare).

- So, overall, this takes about *M x N* time.

K-NN improvements

1. IB2: save memory, speed up classification
2. IB3: deal with noise

---

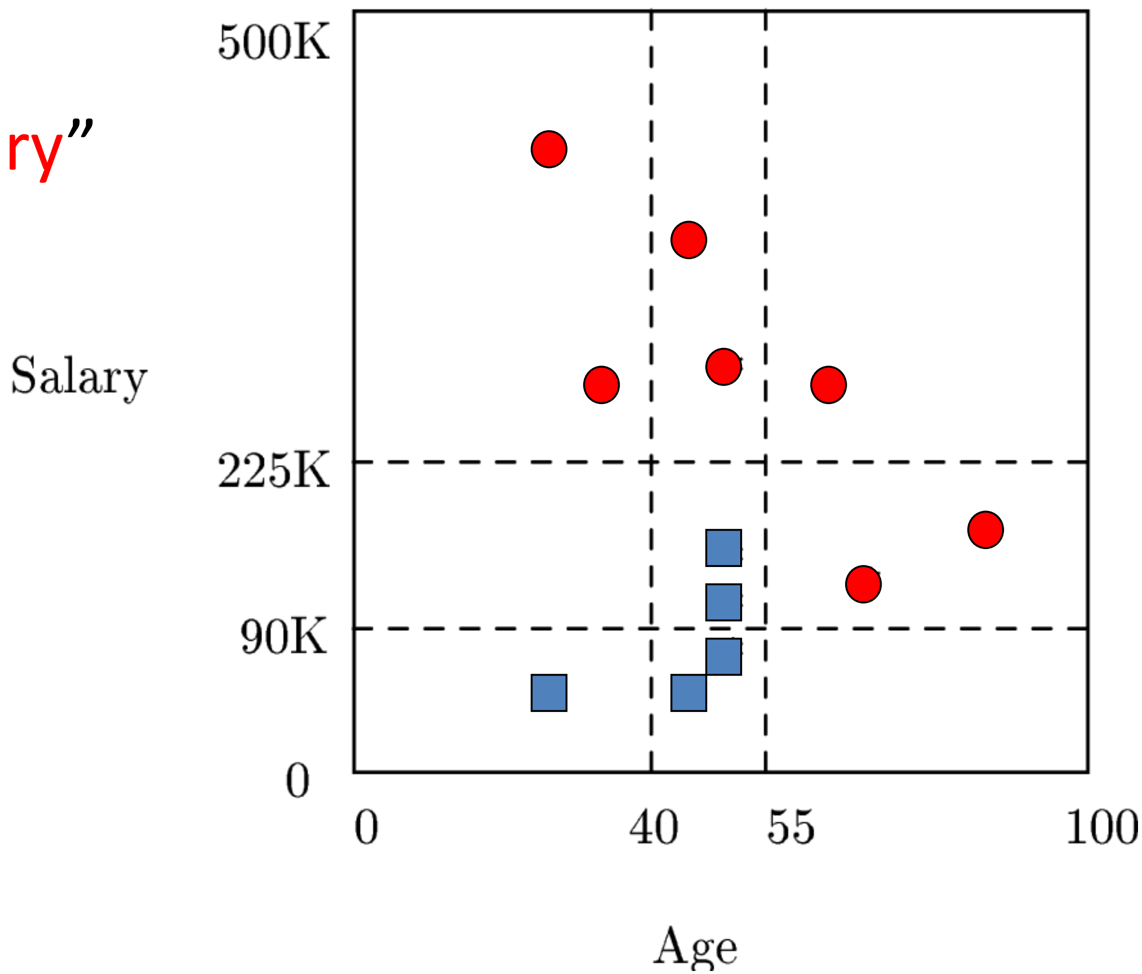
*) IB – Instance Based

IB2 main idea

- Work incrementally
- Only incorporate misclassified instances

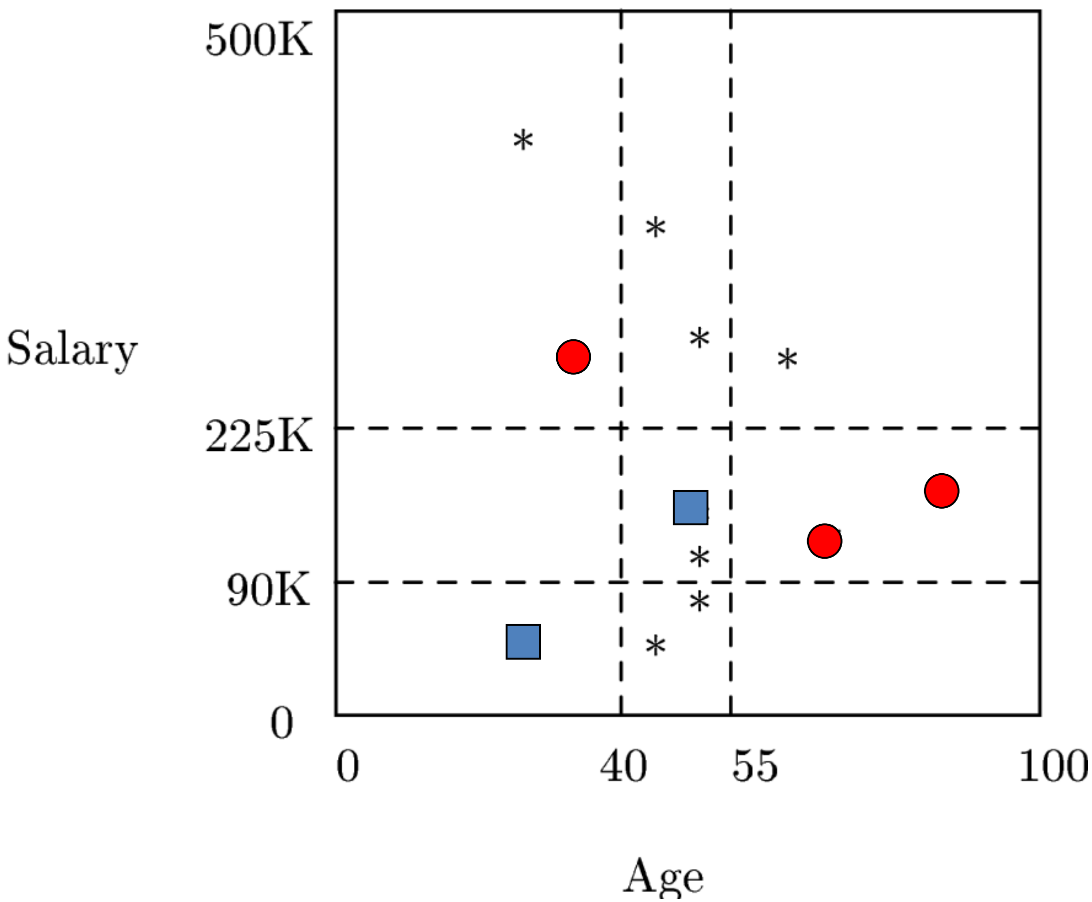
Example: IB2

Dataset:
"Who buys gold jewelry"



IB2 example

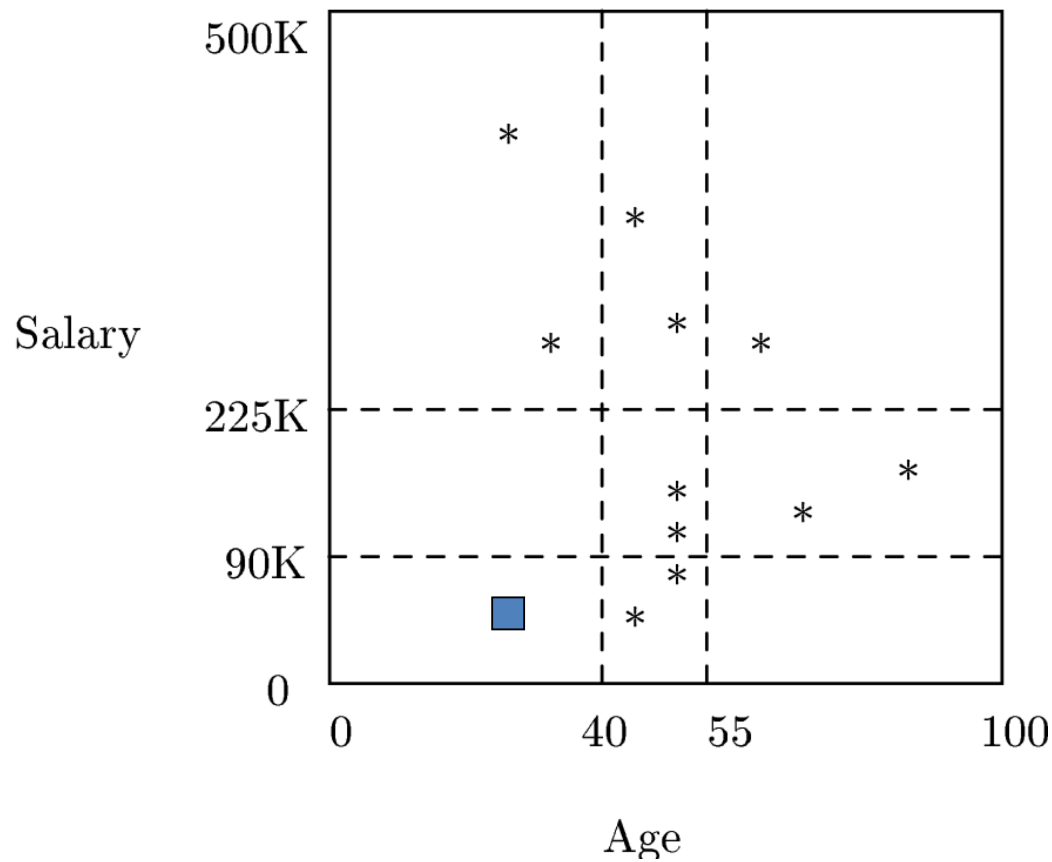
- Data:
  - (25,60,no)
  - (85,140,yes)
  - (45,60,no)
  - (30,260,yes)
  - (50,75,no)
  - (50,120,no)
  - (70,110,yes)
  - (25,400,yes)
  - (50,100,no)
  - (45,350,yes)
  - (50,275,yes)
  - (60,260,yes)



IB2 output: We memorize only these 5 points.

IB2 example

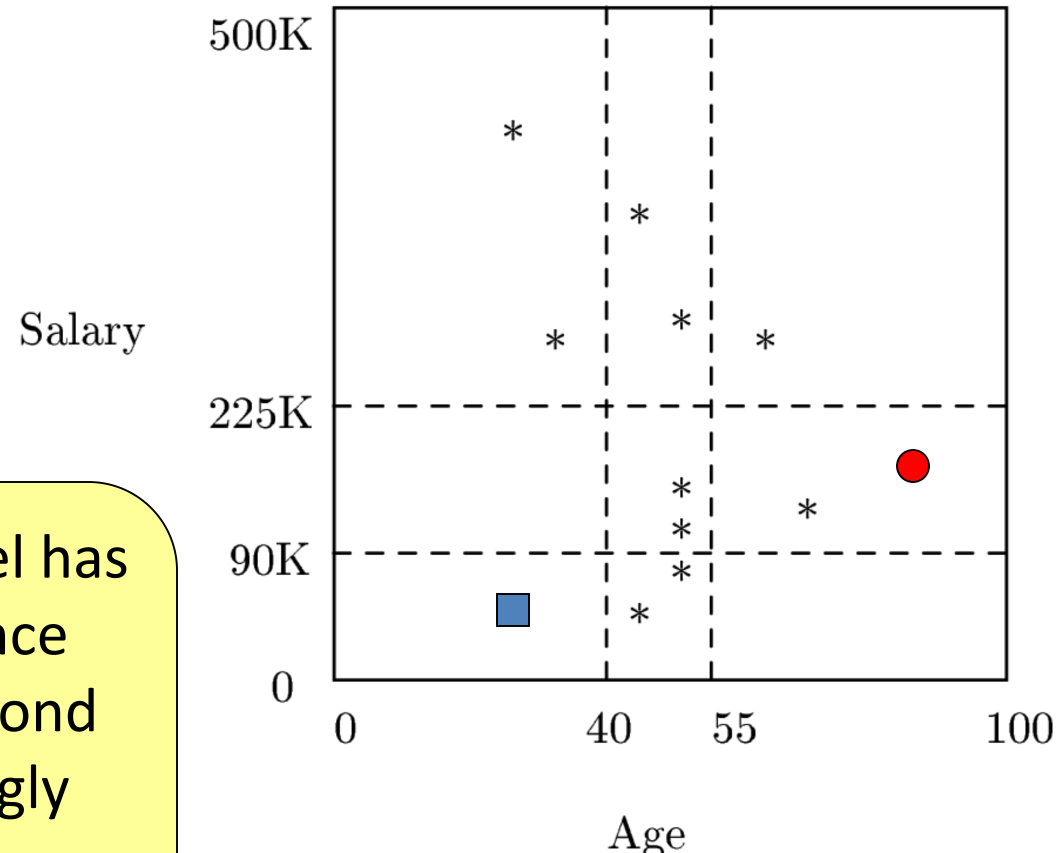
- Data:
  - (25,60,no)



IB2 example

- Data:
  - (25,60,no)
  - *(85,140,yes)*

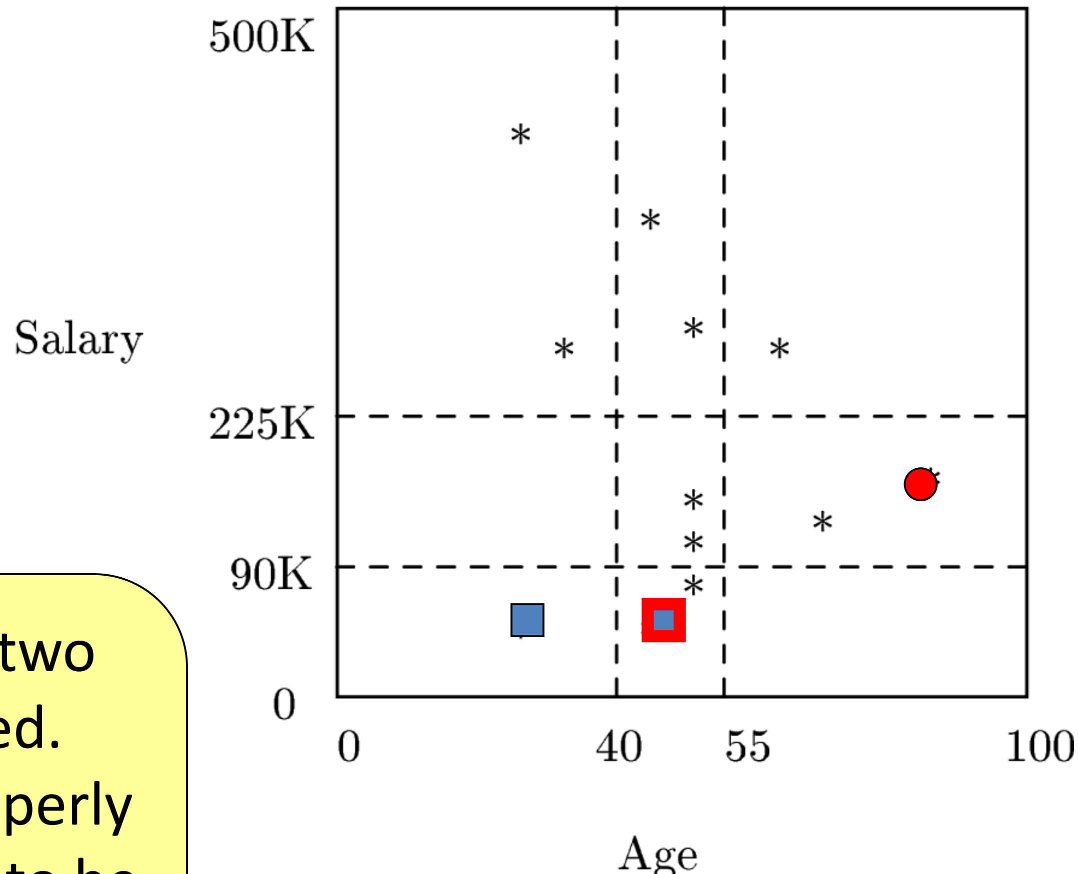
Since so far the model has only the first instance memorized, this second instance gets wrongly classified. So, we memorize it as well.



IB2 example

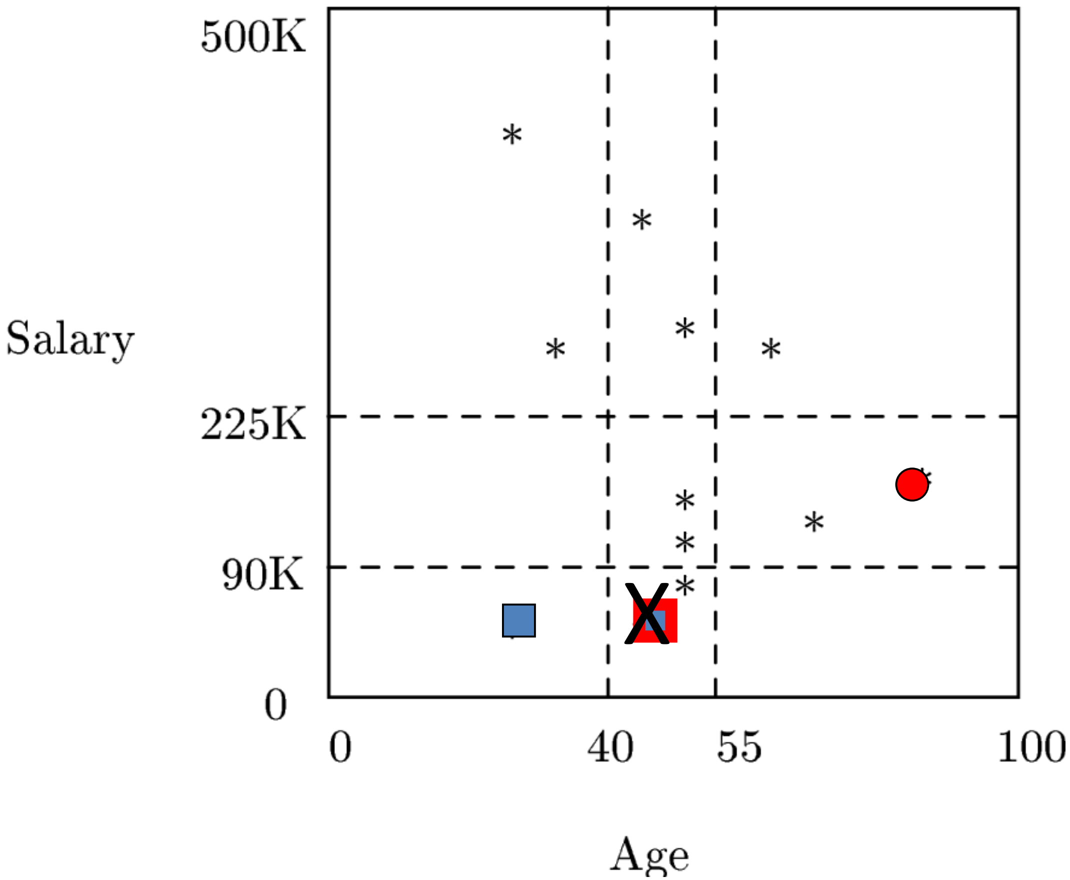
- Data:
  - (25,60,no)
  - (85,140,yes)
  - *(45,60,no)*

So far the model has the two first instances memorized.
The third instance gets properly classified, since it happens to be closer with the first.
So, we don't memorize it.



IB2 example

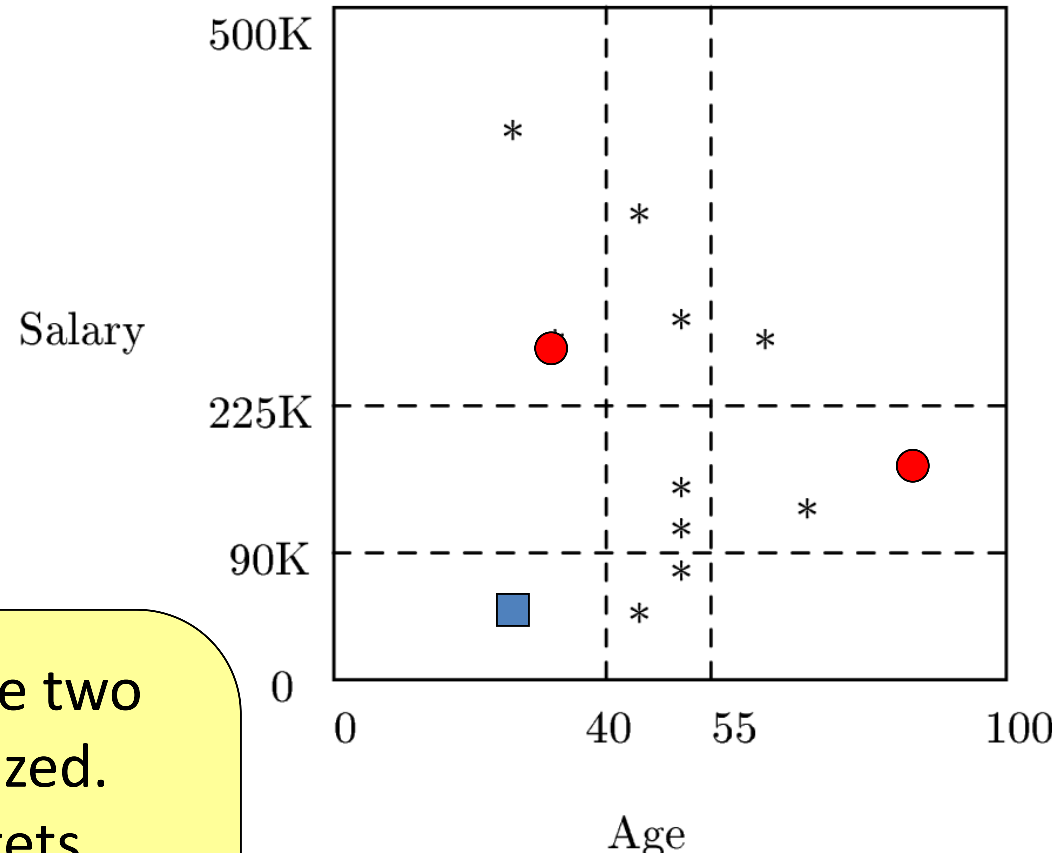
- Data:
  - (25,60,no)
  - (85,140,yes)
  - *(45,60,no)*



IB2 example

- Data:
  - (25,60,no)
  - (85,140,yes)
  - (45,60,no)
  - *(30,260,yes)*

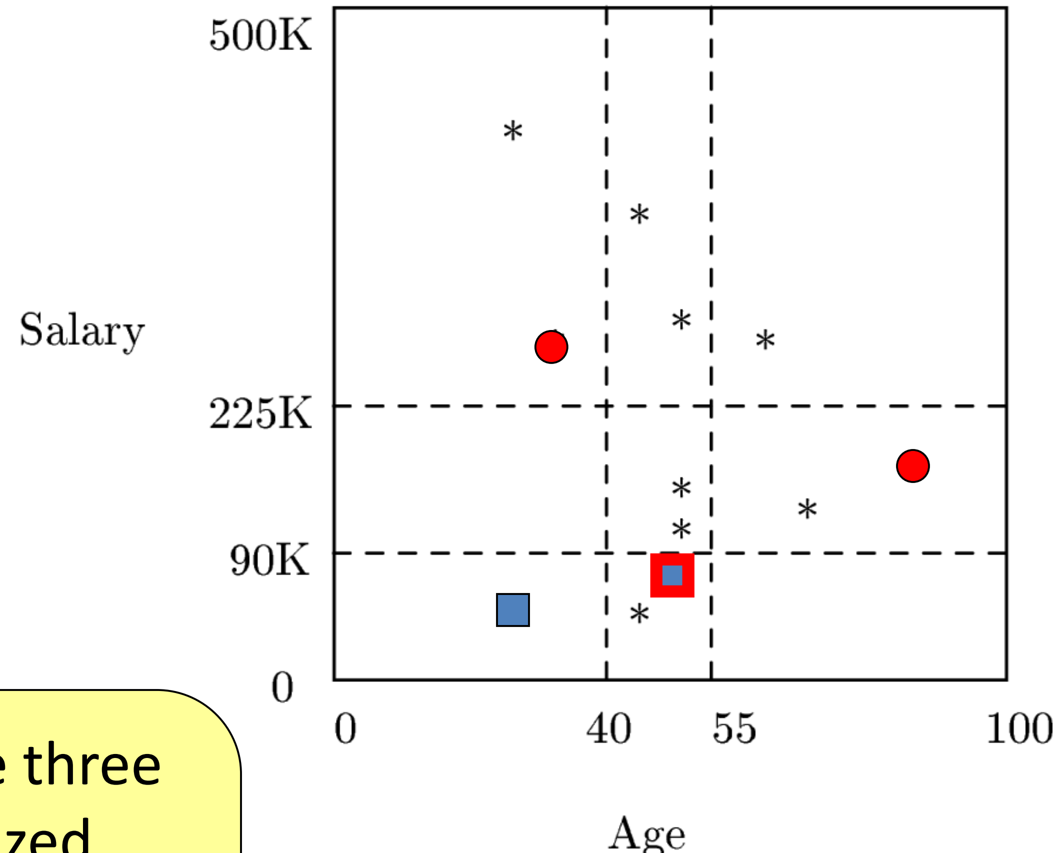
So far the model has the two first instances memorized. The fourth instance gets misclassified, since it happens to be closer with the first. So, we memorize it.



IB2 example

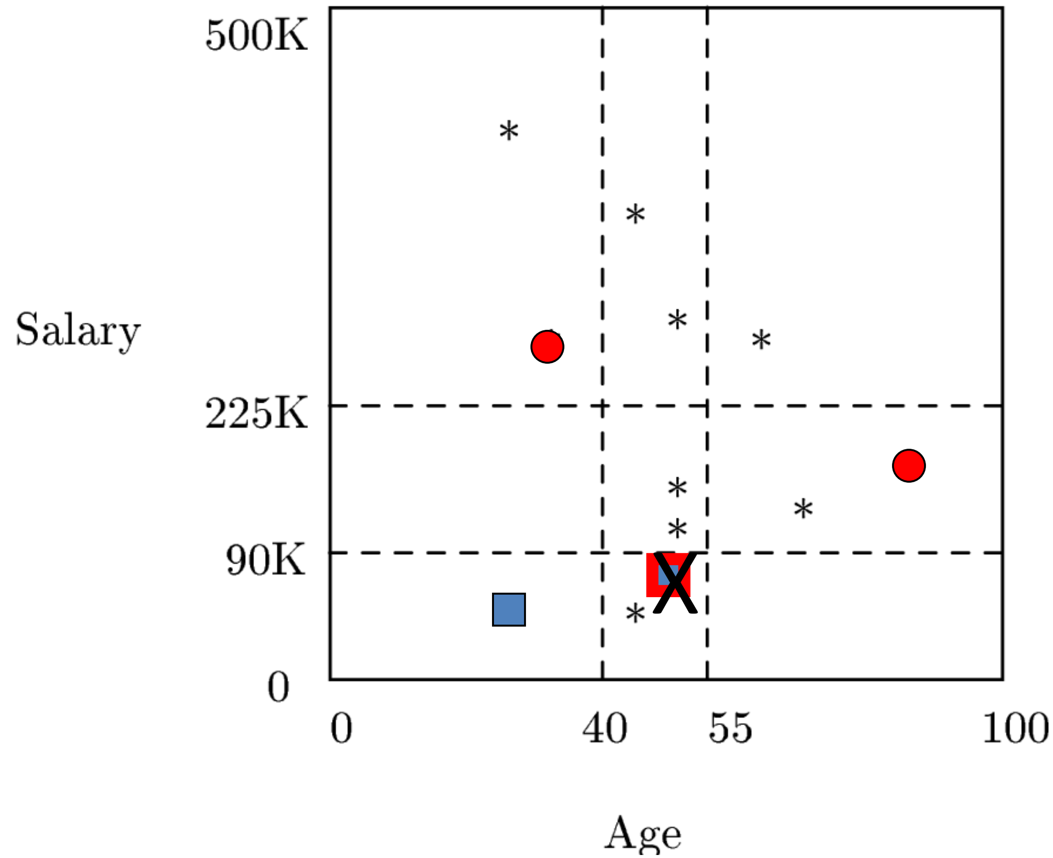
- Data:
  - (25,60,no)
  - (85,140,yes)
  - (45,60,no)
  - (30,260,yes)
  - *(50,75,no)*

So far the model has the three first instances memorized.
The fifth instance gets properly classified, since it happens to be closer with the first.
So, we don't memorize it.



IB2 example

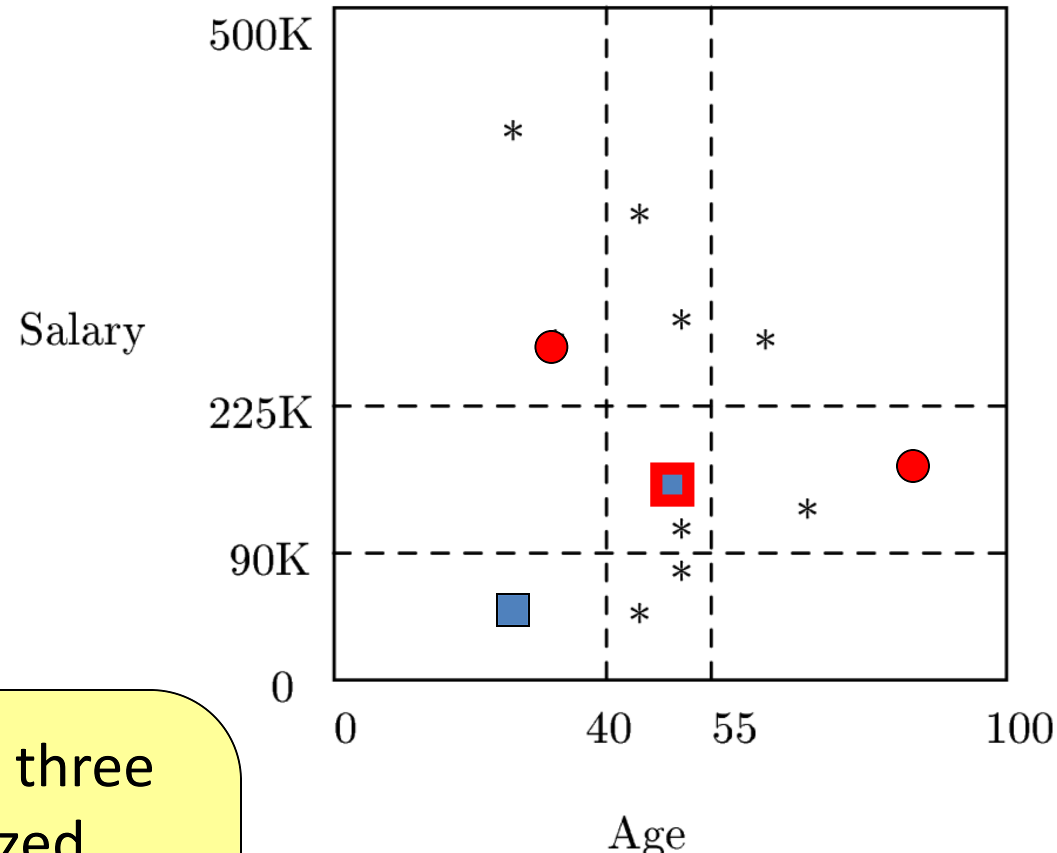
- Data:
  - (25,60,no)
  - (85,140,yes)
  - (45,60,no)
  - (30,260,yes)
  - *(50,75,no)*



IB2 example

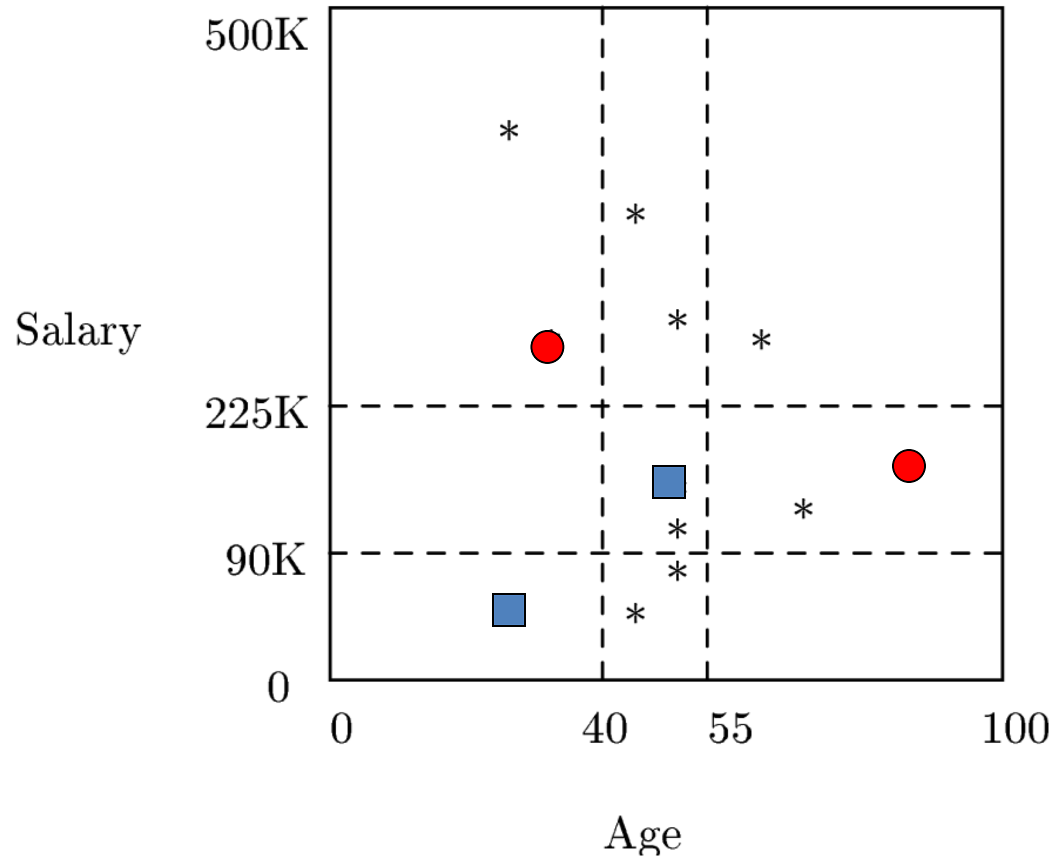
- Data:
  - (25,60,no)
  - (85,140,yes)
  - (45,60,no)
  - (30,260,yes)
  - (50,75,no)
  - *(50,120,no)*

So far the model has the three first instances memorized.
The sixth instance gets wrongly classified, since it happens to be closer with the second.
So, we memorize it.



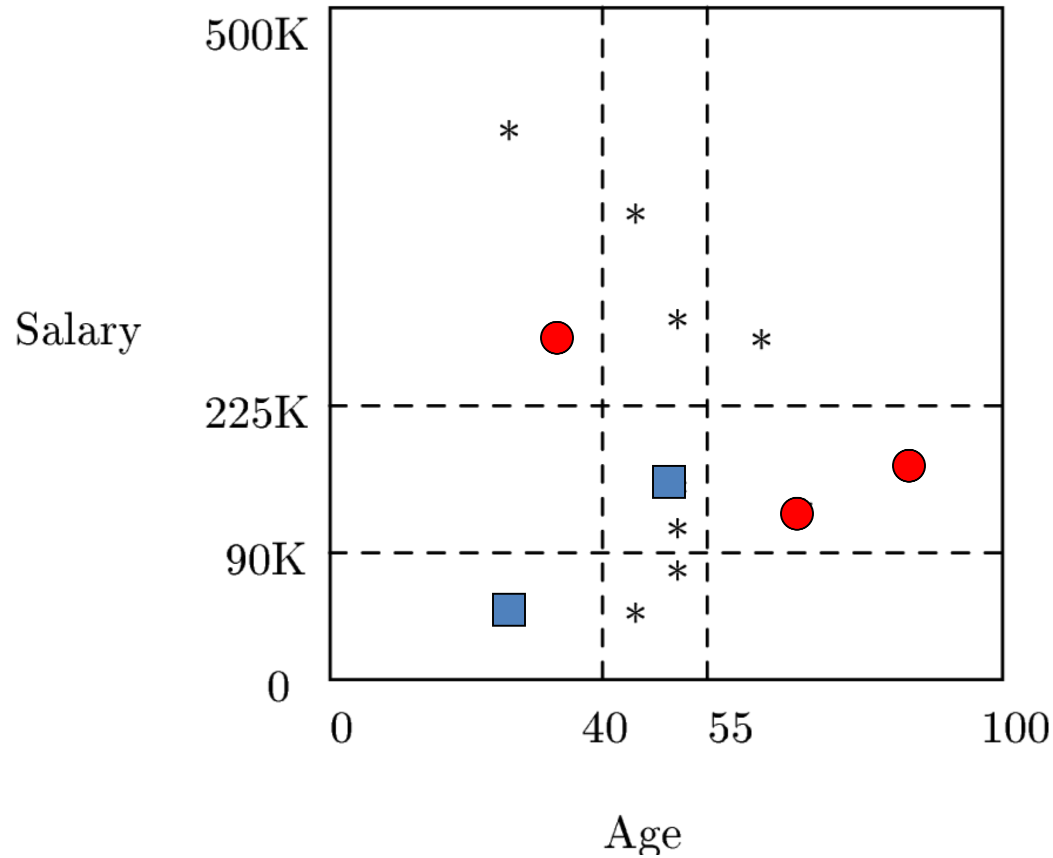
IB2 example

- Data:
  - (25,60,no)
  - (85,140,yes)
  - (45,60,no)
  - (30,260,yes)
  - (50,75,no)
  - *(50,120,no)*



IB2 example

- Continuing in a similar way, we finally get a smaller set to memoriz:.
  - The colored points are the ones that get memorized.



This is the final answer.
I.e. we memorize only these 5 points.

IB2 summary

- Work incrementally
- Only incorporate misclassified instances

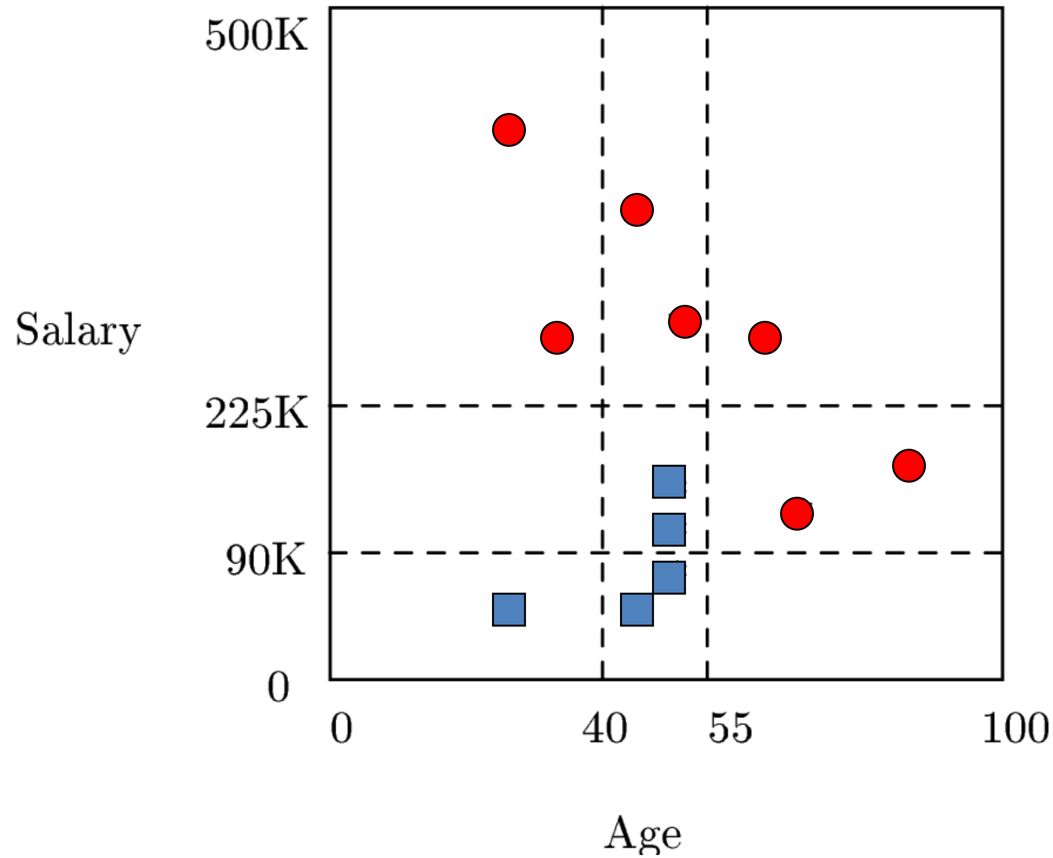
- Problem: noisy data might get incorporated

IB3 main idea

- Discard instances that don't perform well
- Keep a record of the number of correct and incorrect classification decisions that each exemplar makes.
- Two predetermined thresholds are set on success ratio. An instance is kept if:
  - If the number of incorrect classifications is $\leq$ the negative threshold ε
  - If the number of correct classifications $\geq$ the positive threshold γ.

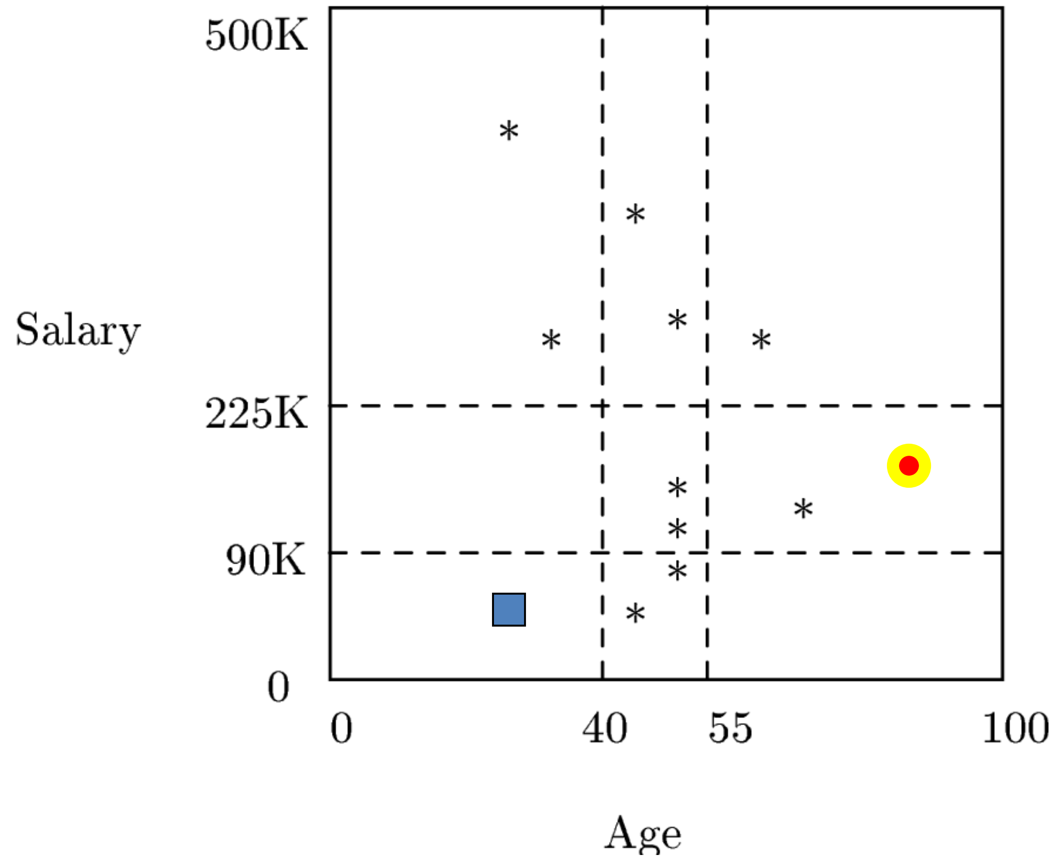
IB3 example

- Suppose the lower threshold ε=0, and the upper threshold γ=1.
- Shuffle the original dataset:
  - (25,60,no)
  - (85,140,yes)
  - (45,60,no)
  - (30,260,yes)
  - (50,75,no)
  - (50,120,no)
  - (70,110,yes)
  - (25,400,yes)
  - (50,100,no)
  - (45,350,yes)
  - (50,275,yes)
  - (60,260,yes)



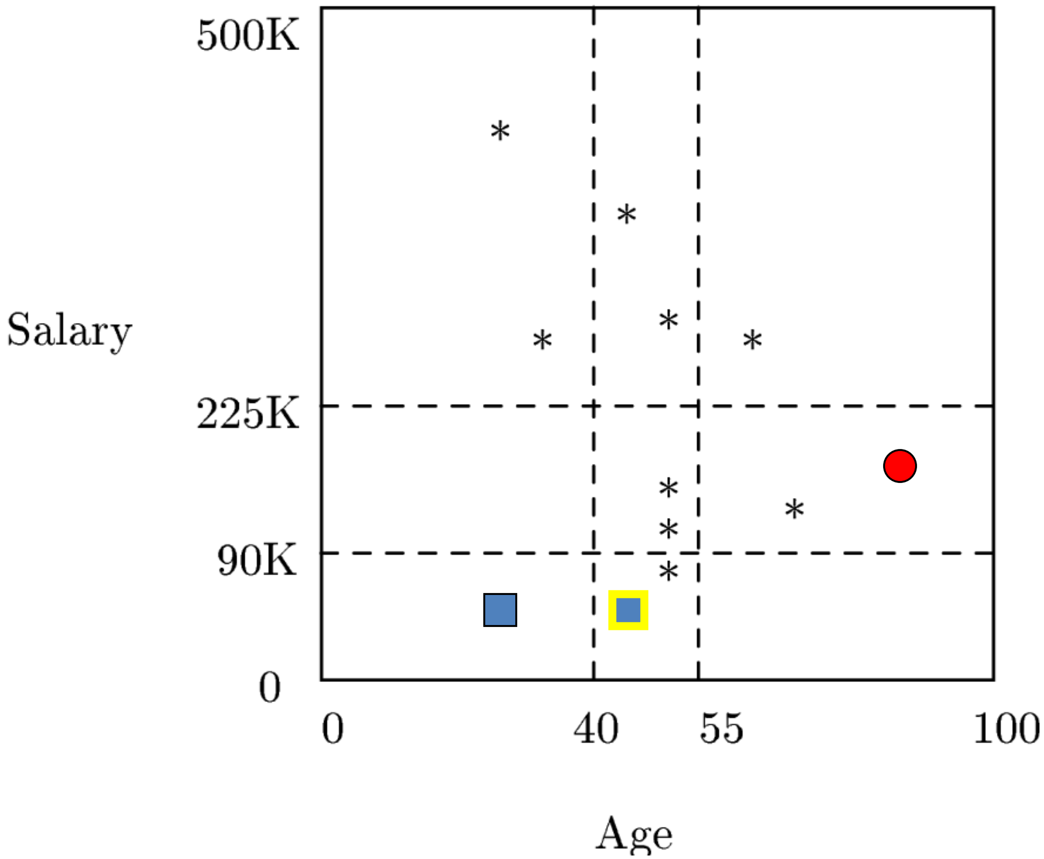
IB3 example

- ε=0, γ=1.
- Adding two first instances:
  - (25,60,no) [1, 0]
  - (85,140,yes)



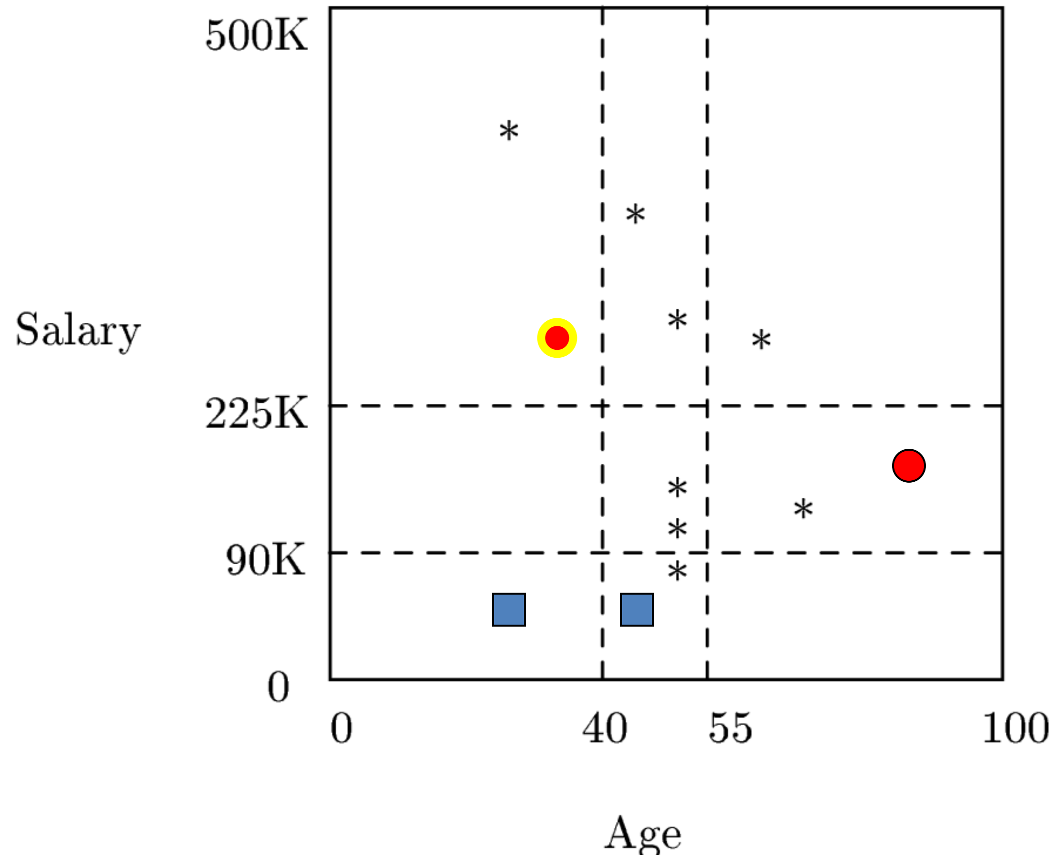
IB3 example

- ε=0, γ=1.
- Adding next:
  - (25,60,no) [1, 1]
  - (85,140,yes) [0, 0]
  - (45,60,no)



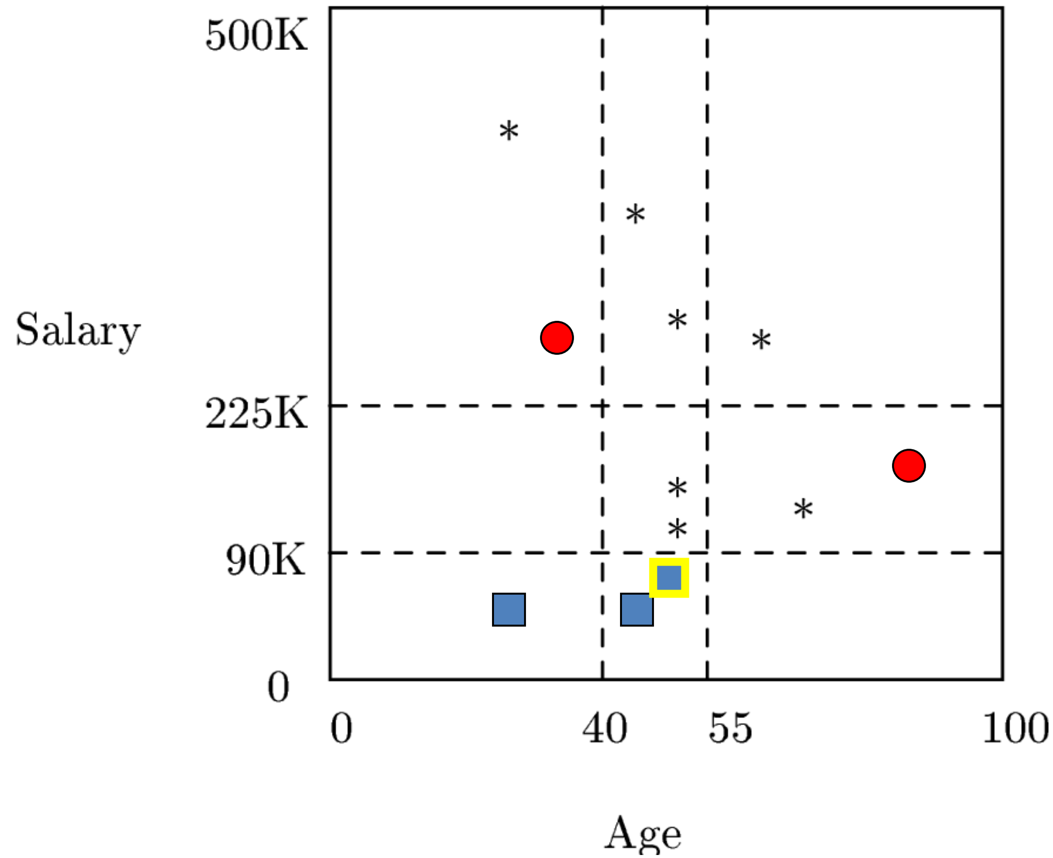
IB3 example

- ε=0, γ=1.
  - (25,60,no) [2,1]
  - (85,140,yes) [0,0]
  - (45,60,no) [0,0]
  - (30,260,yes)



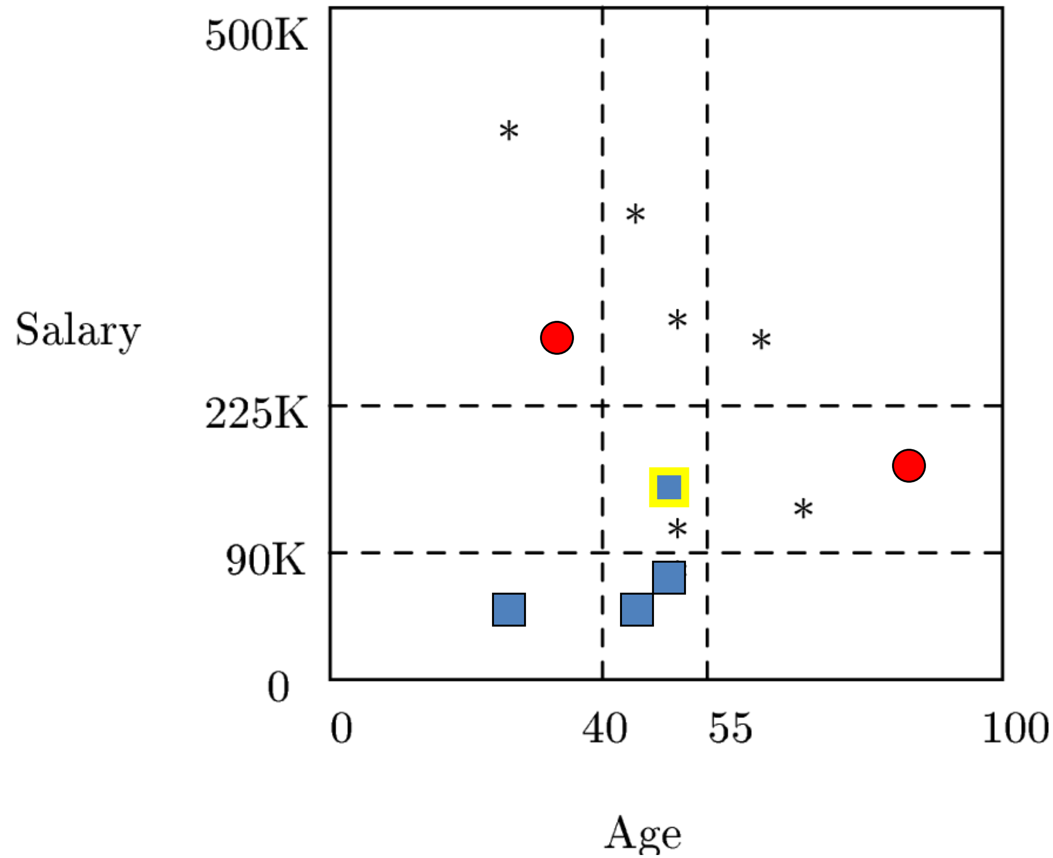
IB3 example

- ε=0, γ=1.
  - (25,60,no) [2,1]
  - (85,140,yes) [0,0]
  - (45,60,no) [0,1]
  - (30,260,yes) [0,0]
  - (50,75,no)



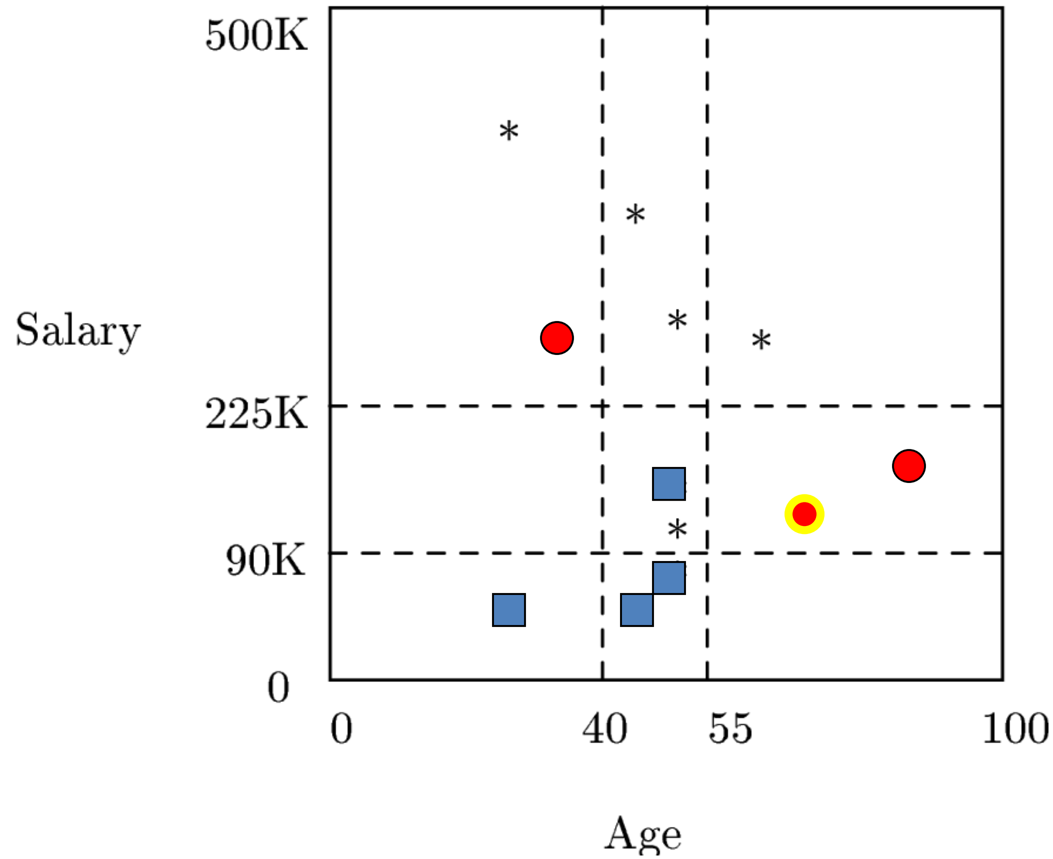
IB3 example

- ε=0, γ=1.
  - (25,60,no) [2,1]
  - (85,140,yes) [0,0]
  - (45,60,no) [0,1]
  - (30,260,yes) [0,0]
  - (50,75,no) [0,1]
  - (50,120,no)
  -



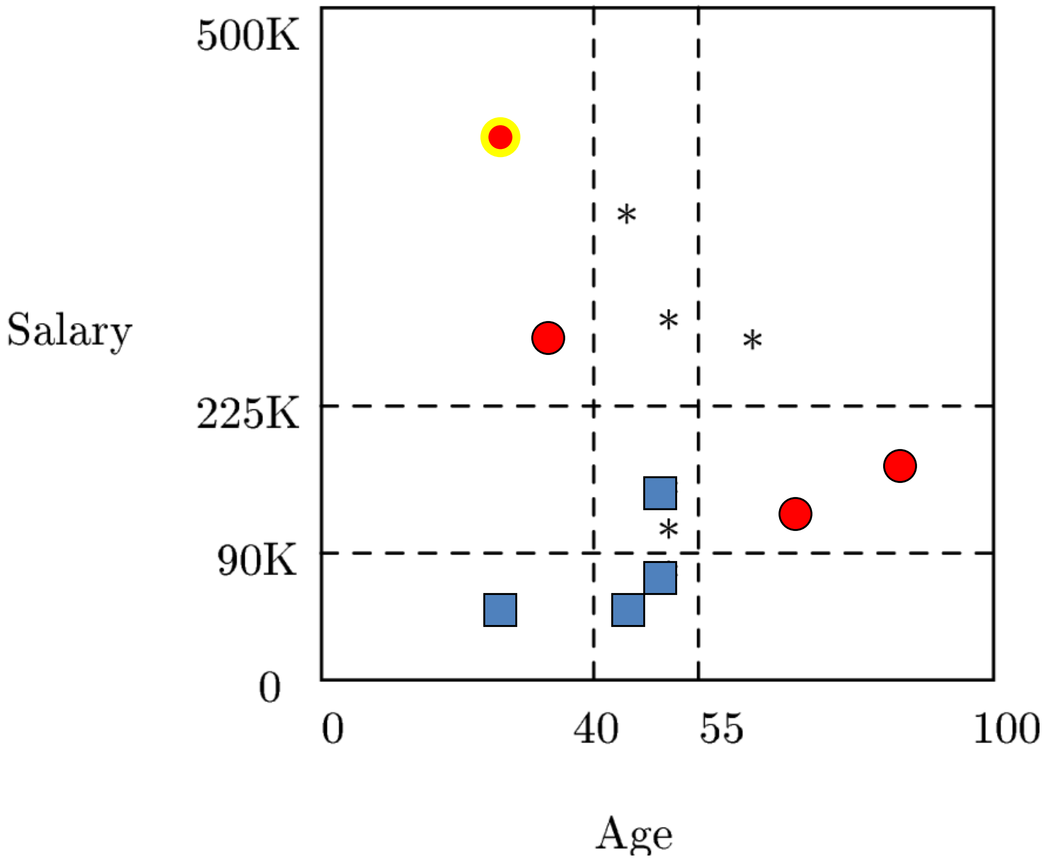
IB3 example

- ε=0, γ=1.
  - (25,60,no) [2,1]
  - (85,140,yes) [0,1]
  - (45,60,no) [0,1]
  - (30,260,yes) [0,0]
  - (50,75,no) [0,1]
  - (50,120,no) [0,0]
  - (70,110,yes)



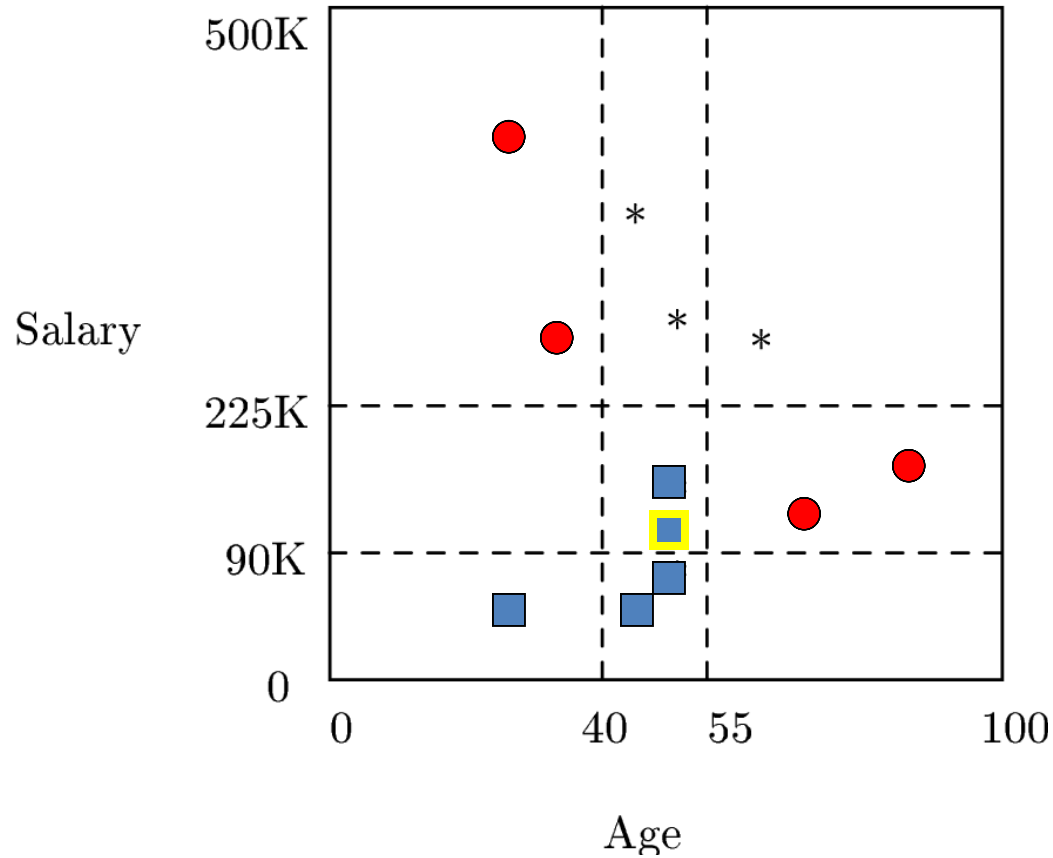
IB3 example

- ε=0, γ=1.
  - (25,60,no) [2,1]
  - (85,140,yes) [0,1]
  - (45,60,no) [0,1]
  - (30,260,yes) [0, 1]
  - (50,75,no) [0, 1]
  - (50,120,no) [0, 0]
  - (70,110,yes) [0, 0]
  - (25,400,yes)



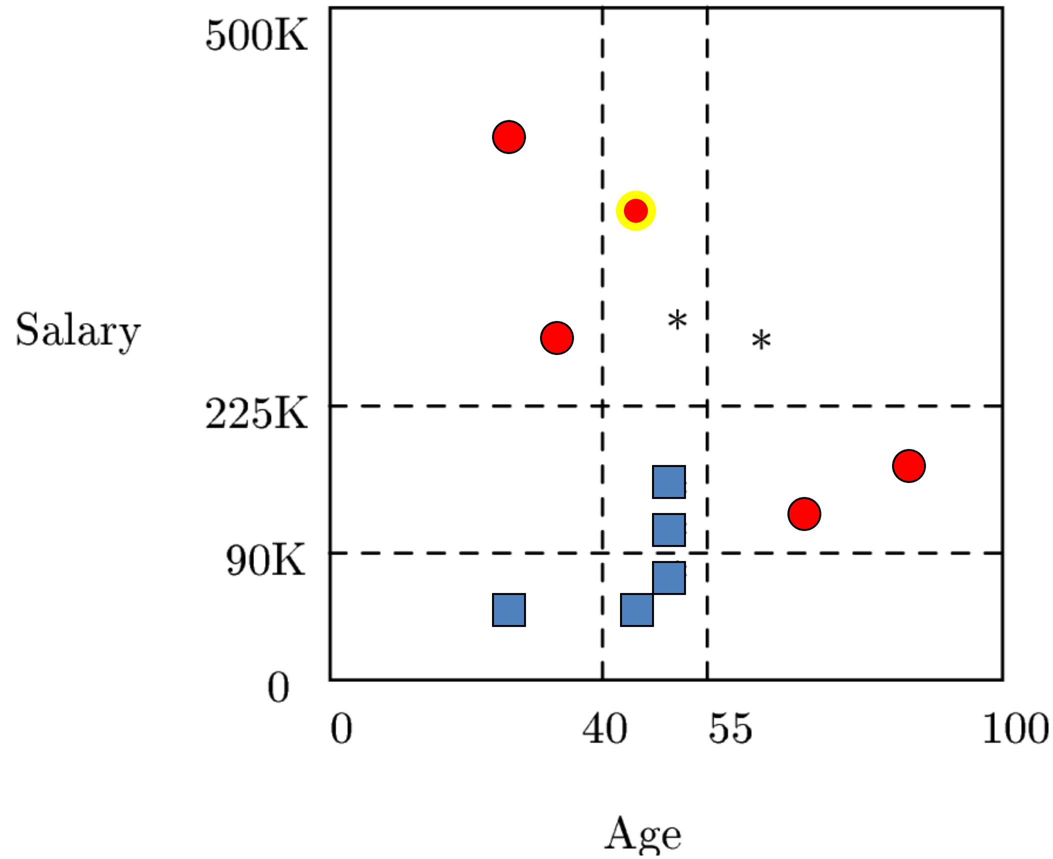
IB3 example

- ε=0, γ=1.
  - (25,60,no) [2, 1]
  - (85,140,yes) [0, 1]
  - (45,60,no) [0, 1]
  - (30,260,yes) [0, 1]
  - (50,75,no) [0, 1]
  - (50,120,no) [0, 1]
  - (70,110,yes) [0, 0]
  - (25,400,yes) [0, 0]
  - (50,100,no)



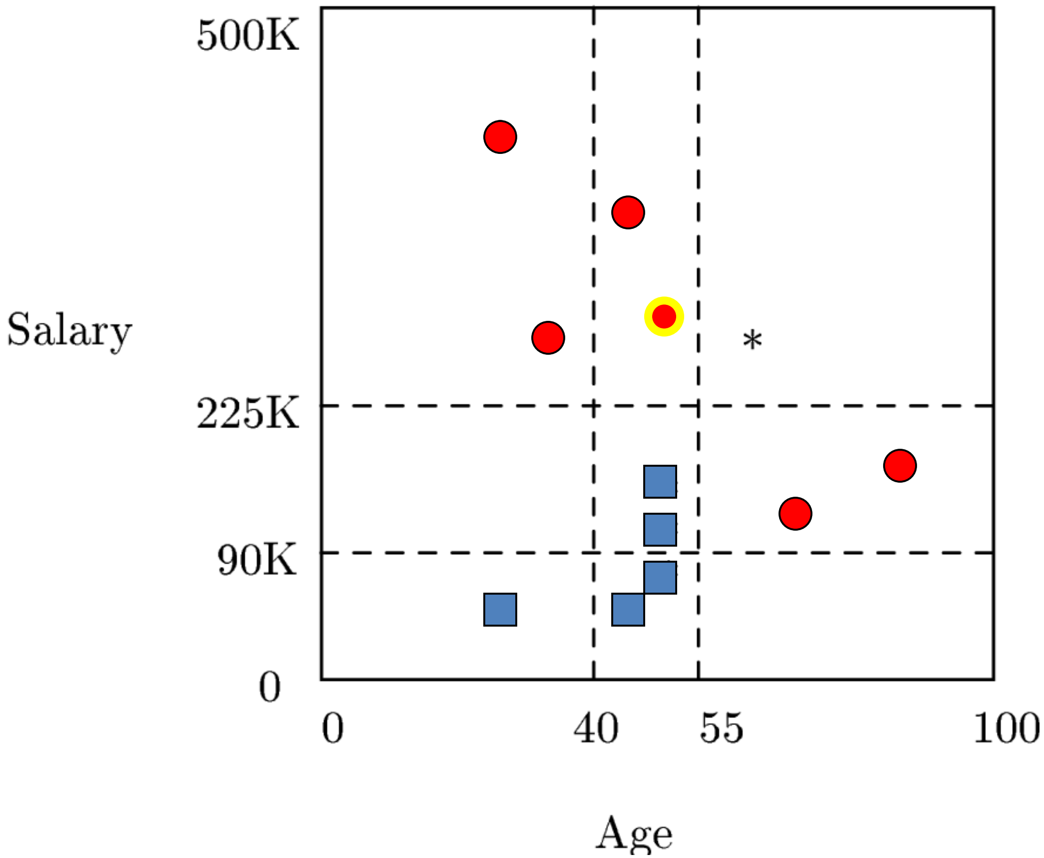
IB3 example

- ε=0, γ=1.
  - (25,60,no) [2, 1]
  - (85,140,yes) [0, 1]
  - (45,60,no) [0, 1]
  - (30,260,yes) [0, 1]
  - (50,75,no) [0, 1]
  - (50,120,no) [0, 1]
  - (70,110,yes) [0, 0]
  - (25,400,yes) [0, 1]
  - (50,100,no) [0, 0]
  - (45,350,yes)



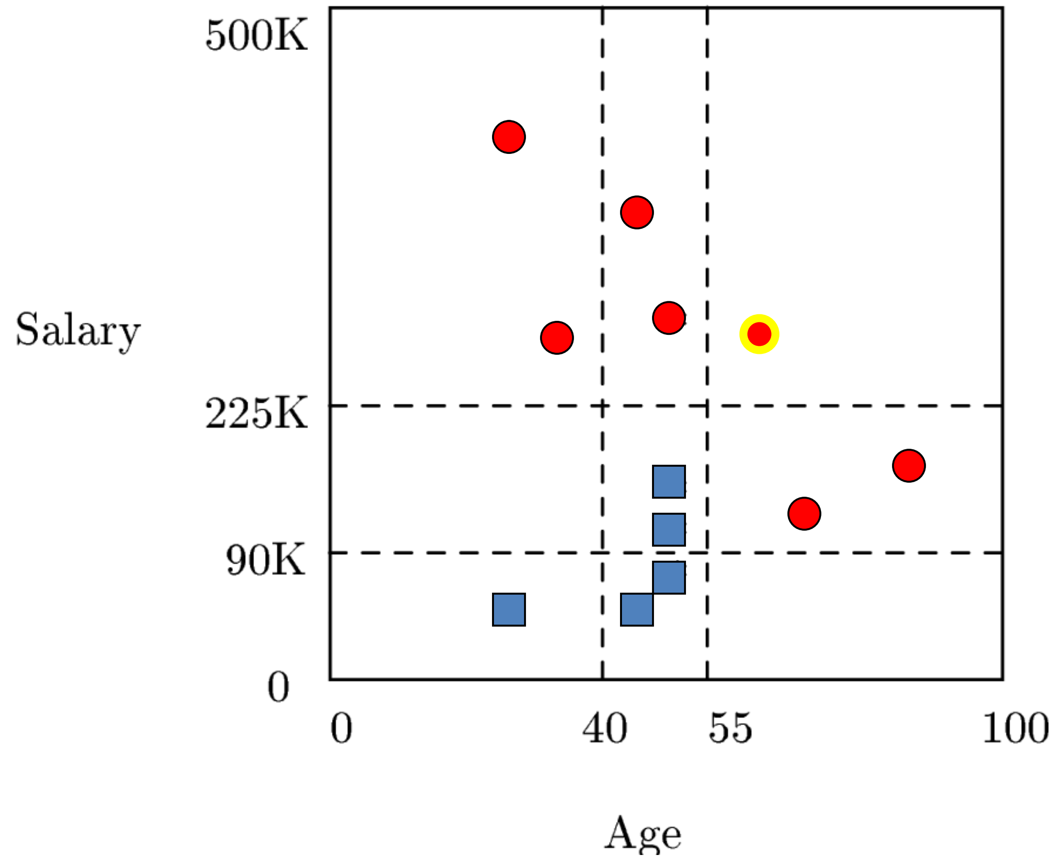
IB3 example

- ε=0, γ=1.
  - (25,60,no) [2, 1]
  - (85,140,yes) [0, 1]
  - (45,60,no) [0, 1]
  - (30,260,yes) [0, 2]
  - (50,75,no) [0, 1]
  - (50,120,no) [0, 1]
  - (70,110,yes) [0, 0]
  - (25,400,yes) [0, 1]
  - (50,100,no) [0, 0]
  - (45,350,yes) [0, 0]
  - (50,275,yes)



IB3 example

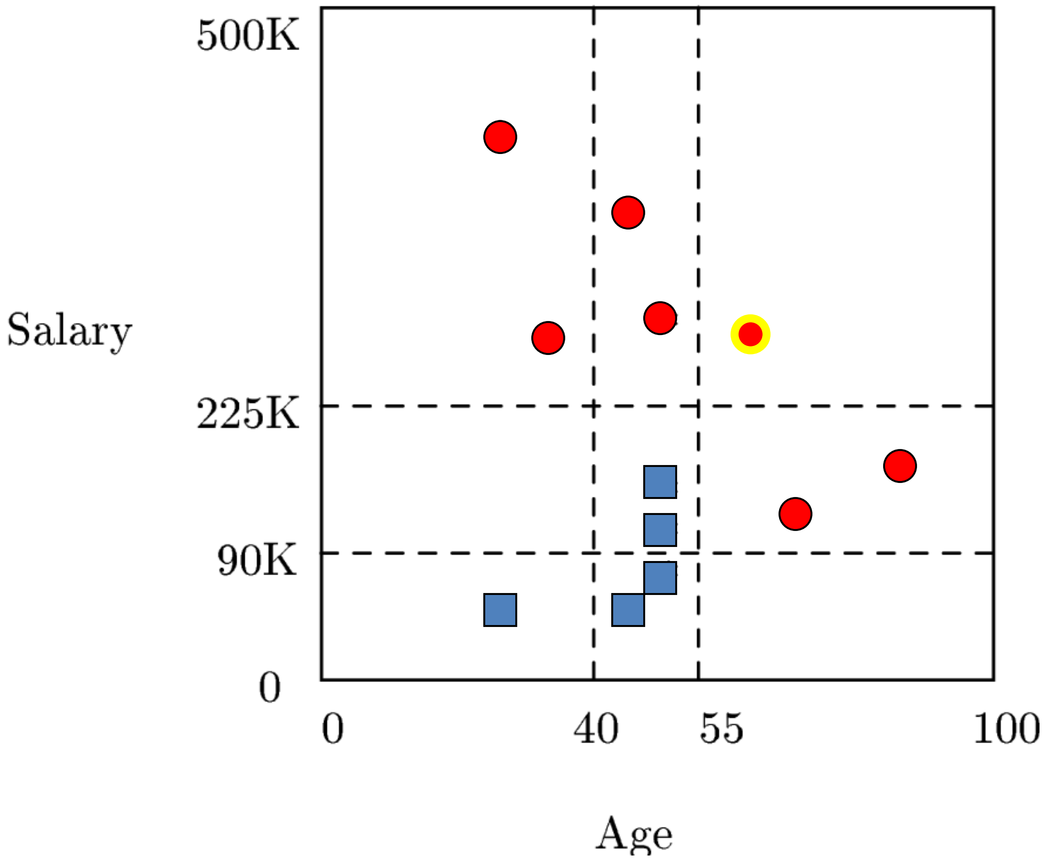
- ε=0, γ=1.
  - (25,60,no) [2, 1]
  - (85,140,yes) [0, 1]
  - (45,60,no) [0, 1]
  - (30,260,yes) [0, 2]
  - (50,75,no) [0, 1]
  - (50,120,no) [0, 1]
  - (70,110,yes) [0, 0]
  - (25,400,yes) [0, 1]
  - (50,100,no) [0, 0]
  - (45,350,yes) [0, 0]
  - (50,275,yes) [0, 1]
  - (60,260,yes)



What do we discard?

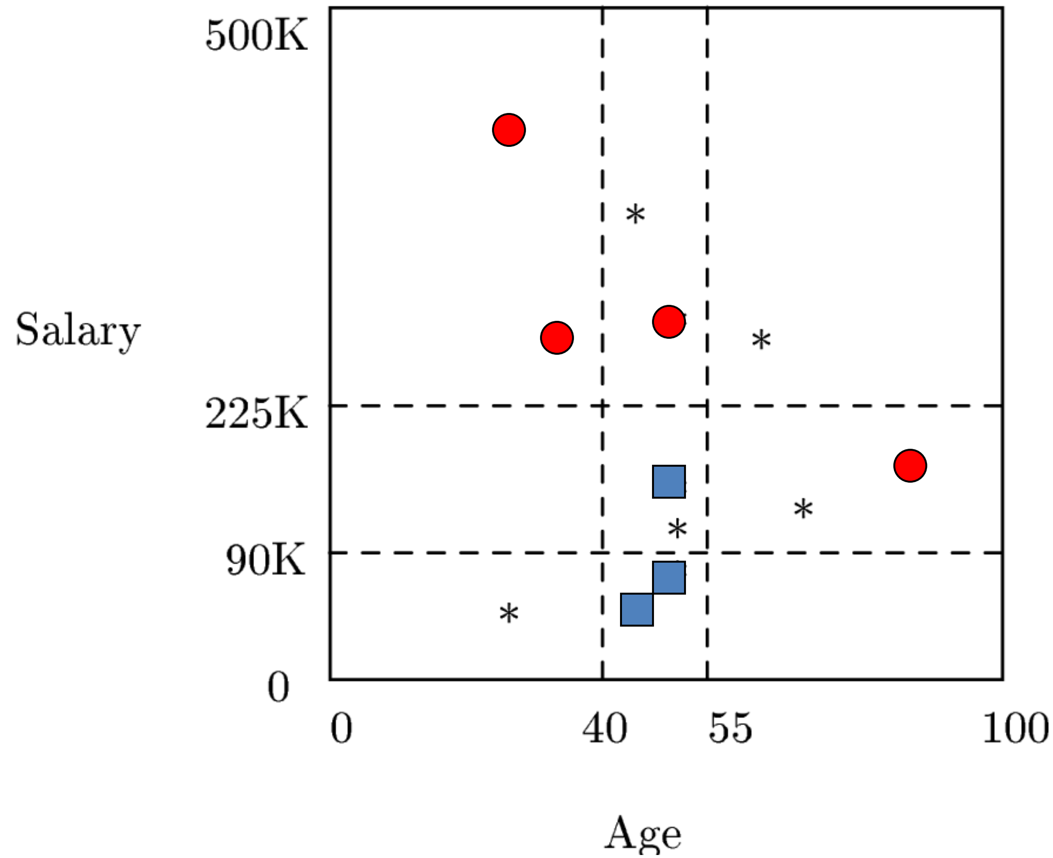
IB3 example

- ε=0, γ=1.
  - ~~(25,60,no) [2, 1]~~
  - (85,140,yes) [0, 1]
  - (45,60,no) [0, 1]
  - (30,260,yes) [0, 2]
  - (50,75,no) [0, 1]
  - (50,120,no) [0, 1]
  - ~~(70,110,yes) [0, 0]~~
  - (25,400,yes) [0, 1]
  - ~~(50,100,no) [0, 0]~~
  - ~~(45,350,yes) [0, 0]~~
  - (50,275,yes) [0, 1]
  - ~~(60,260,yes)~~



IB3 example

- The points that will be kept for classification are:
  - (85,140,yes) [0, 1]
  - (45,60,no) [0, 1]
  - (30,260,yes) [0, 2]
  - (50,75,no) [0, 1]
  - (50,120,no) [0, 1]
  - (25,400,yes) [0, 1]
  - (50,275,yes) [0, 1]



IB3 summary

- Discard instances that don't perform well
- Keep a record of the number of correct and incorrect classification decisions that each exemplar makes.
- After all instances have been added keep only the ones with:
  - The number of incorrect classifications is $\leq \varepsilon$
  - If the number of correct classifications $\geq \gamma$.

A nearest neighbor approach to making recommendations

Knowing that lots of people liked something is not enough. *Who* liked is important.

?

Underworld: Awakening 3D ★★★☆☆

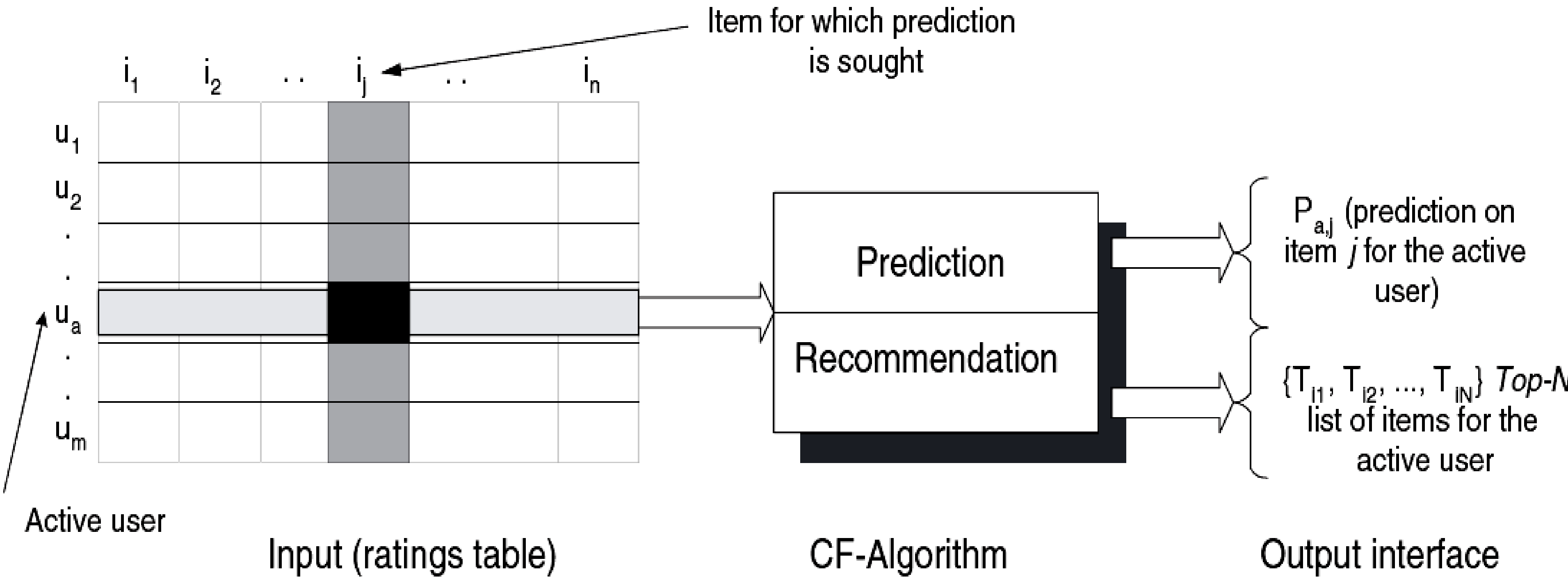
Reviews - Trailer - IMDb

1hr 28min - Rated 18-A - Action/Adventure/Scifi/Fantasy/Horror - English
Director: Mans Marlind - Cast: Kate Beckinsale, Scott Speedman, India Eisley, Charles Dance, Michael Ealy - :

Selene escapes imprisonment to find herself in a world where humans have discovered the existence of both Vampire and Lycan clans, and are conducting an all-out war to eradicate both immortal species.

Collaborative filtering

Recommendation vs. prediction



Automated recommender system

- Build a customer profile
- Compare the new customer’s profile with the profiles of other customers, locate similar “neighbors”
- Predict the rating that the new customer would give to items he has not yet rated by combining ratings of a peer group selected by similar tastes
- Rank predictions and output top-scored ones

Automated recommender system

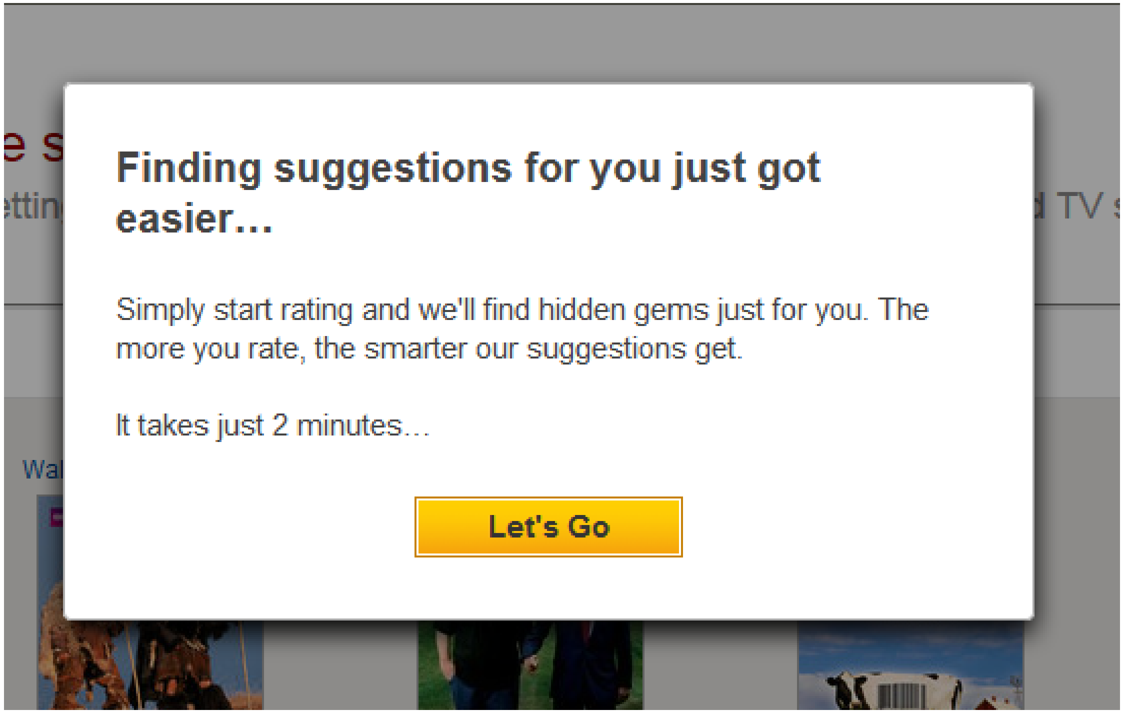
1. Build a customer profile
2. Compare the new customer's profile with the profiles of other customers, locate similar "neighbors"
3. Predict the rating that the new customer would give to items he has not yet rated by combining ratings of a peer group selected by similar tastes
4. Rank predictions and output top-scored ones

Customer profile: sparsity

- Each customer is represented as a vector with one element per each item
  - Rating on a scale 0 – 1
  - Null for ‘no opinion’
- Much more items than ratings (thousands of items at Amazon)
- Profile is sparse: most entries are Null

Creating customer profile

A reasonable approach: have new customers rate a list of 20 or so most frequently rated items and then free them to rate additional items as they please



Netflix.ca

Creating customer profile

At least for a set of predefined items, we have ratings for most of the users – basis for finding similarities

Automated recommender system

- Build a customer profile
- Compare the new customer’s profile with the profiles of other customers, locate similar “neighbors”
- Predict the rating that the new customer would give to items he has not yet rated by combining ratings of a peer group selected by similar tastes
- Rank predictions and output top-scored ones

Sample input: movies ratings

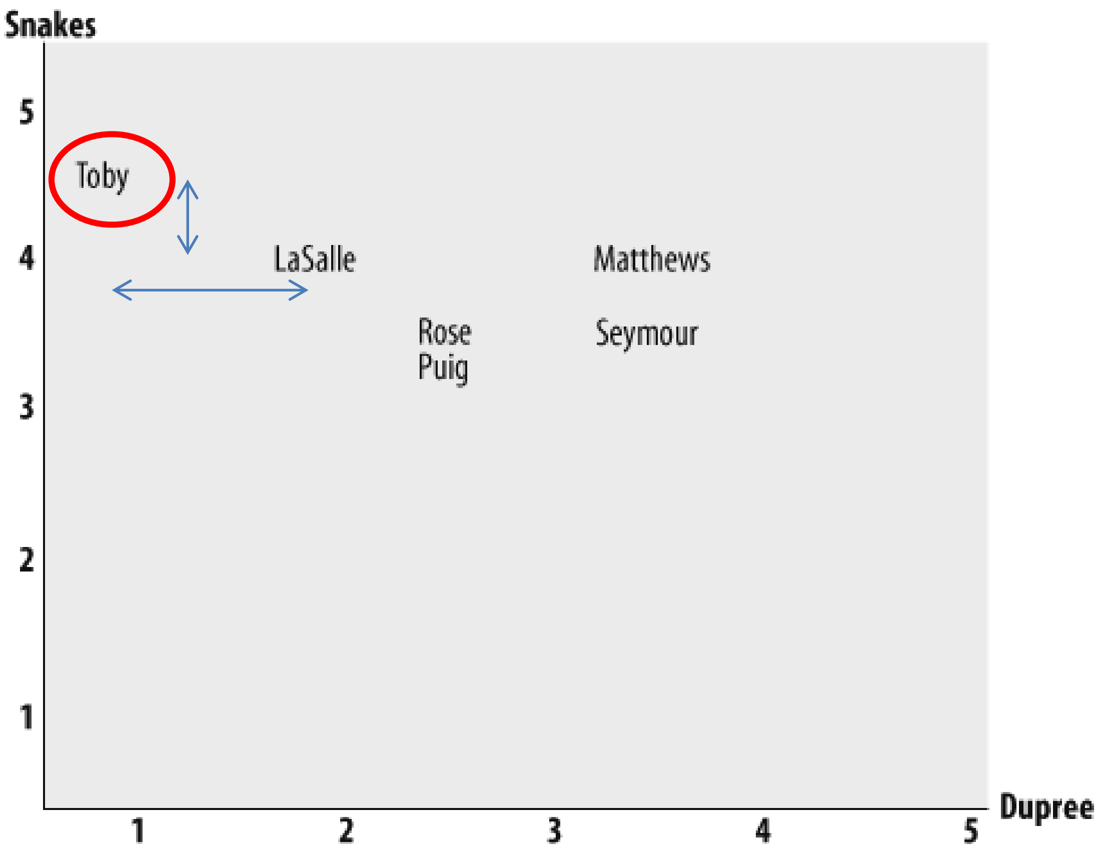
```
critics = {
  'Lisa Rose':       {'Lady in the Water': 2.5,
                      'Snakes on a Plane': 3.5,
                      'Just my Luck': 3.0,
                      'Superman Returns': 3.5,
                      'You, Me and Dupree': 2.5,
                      'The Night Listener': 3.0},

  'Gene Seymour':    {'Lady in the Water': 3.0,
                      'Snakes on a Plane': 3.5,
                      'Just my Luck': 1.5,
                      'Superman Returns': 5.0,
                      'The Night Listener': 3.0,
                      'You, Me and Dupree': 3.5},
  'Michael Phillips': {'Lady in the Water': 2.5,
                      'Snakes on a Plane': 3.0,
                      'Superman Returns': 3.5,
                      'The Night Listener': 4.0},
  'Claudia Puig':    {'Snakes on a Plane': 3.5,
                      'Just my Luck': 3.0,
                      'The Night Listener': 4.5,
                      'Superman Returns': 4.0,
                      'You, Me and Dupree': 2.5},
  'Mick LaSalle':    {'Lady in the Water': 3.0,
                      'Snakes on a Plane': 4.0,
                      'Just my Luck': 2.0,
                      'Superman Returns': 3.0,
                      'The Night Listener': 3.0,
                      'You, Me and Dupree': 2.0},
  'Jack Matthews':   {'Lady in the Water': 3.0,
                      'Snakes on a Plane': 4.0,
                      'Superman Returns': 5.0,
                      'The Night Listener': 3.0,
                      'You, Me and Dupree': 3.5},
  'Toby':            {'Snakes on a Plane': 4.5,
                      'Superman Returns': 4.0,
                      'You, Me and Dupree': 1.0}
}
```

Adopted from: “Programming collective intelligence” by Toby Segaran

Finding Similar Users: distance

- Find similar users by calculating Euclidean distance in a space where each movie is a dimension: the smaller the distance, the more similar are the users

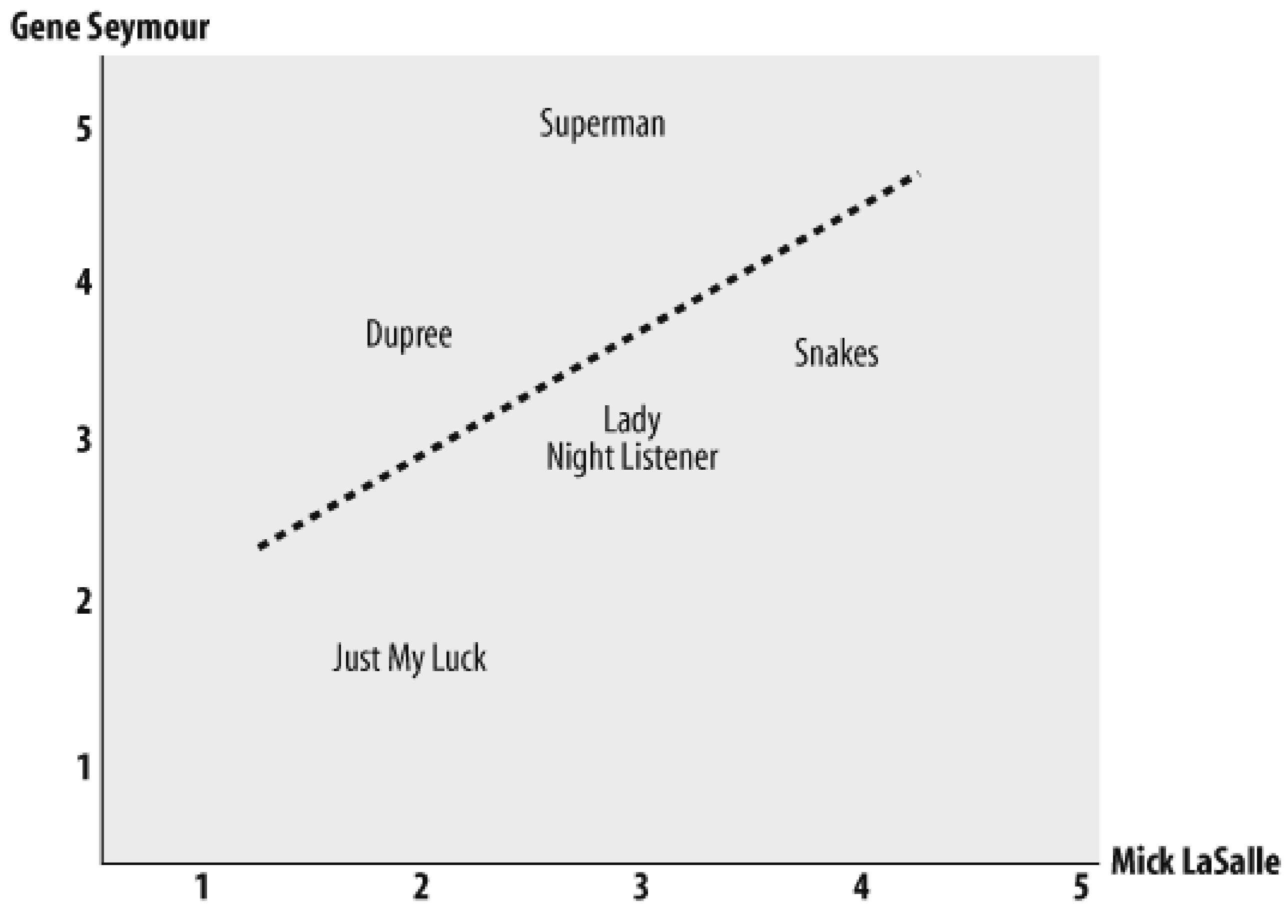


People in preference space

Weigh each neighbor by the distance from an active user

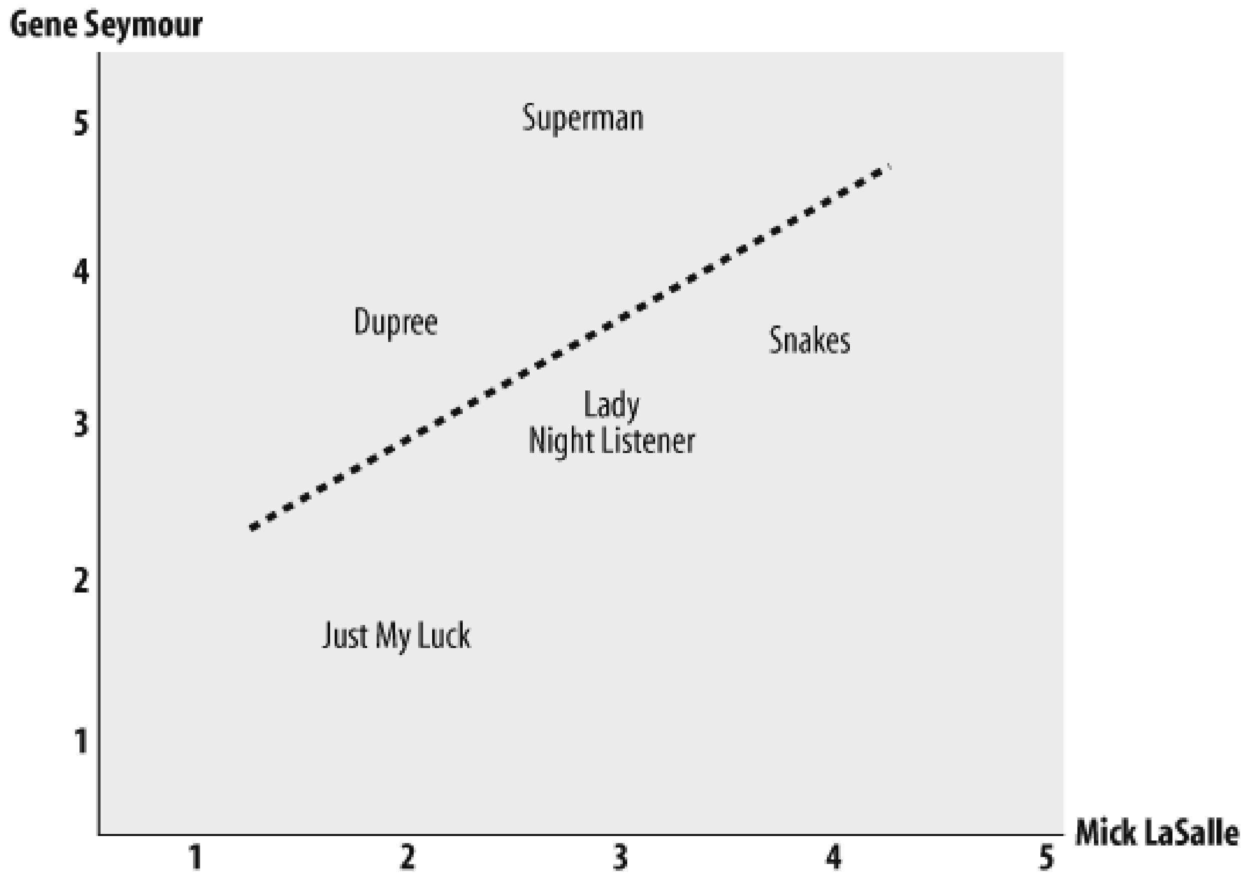
Similar ratings

Both users rank ‘Superman’ higher than ‘Dupree’



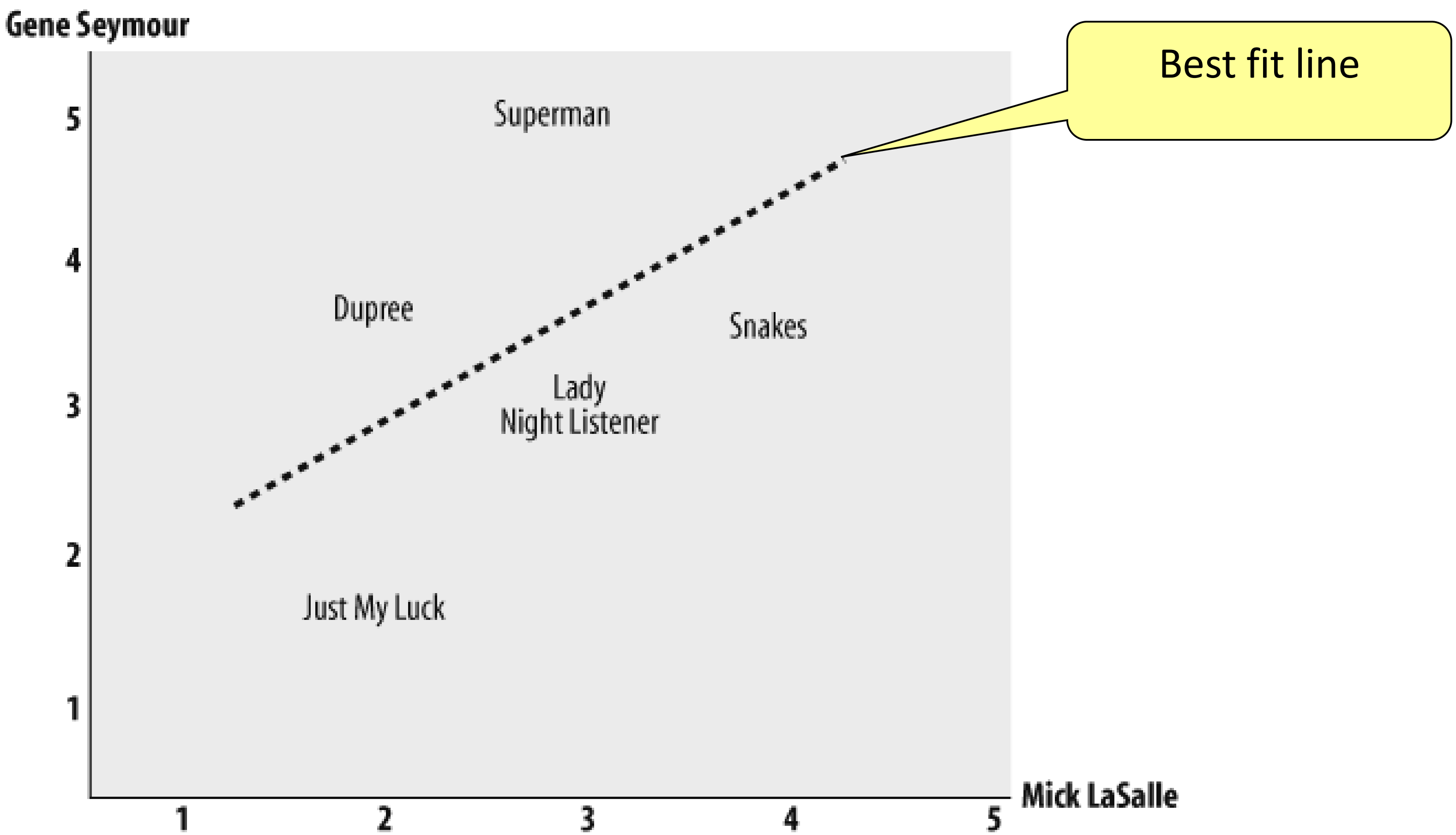
Similar ratings

Large values of ratings for the same movie for Gene tend to be associated with large values for Mick, small values – with small values

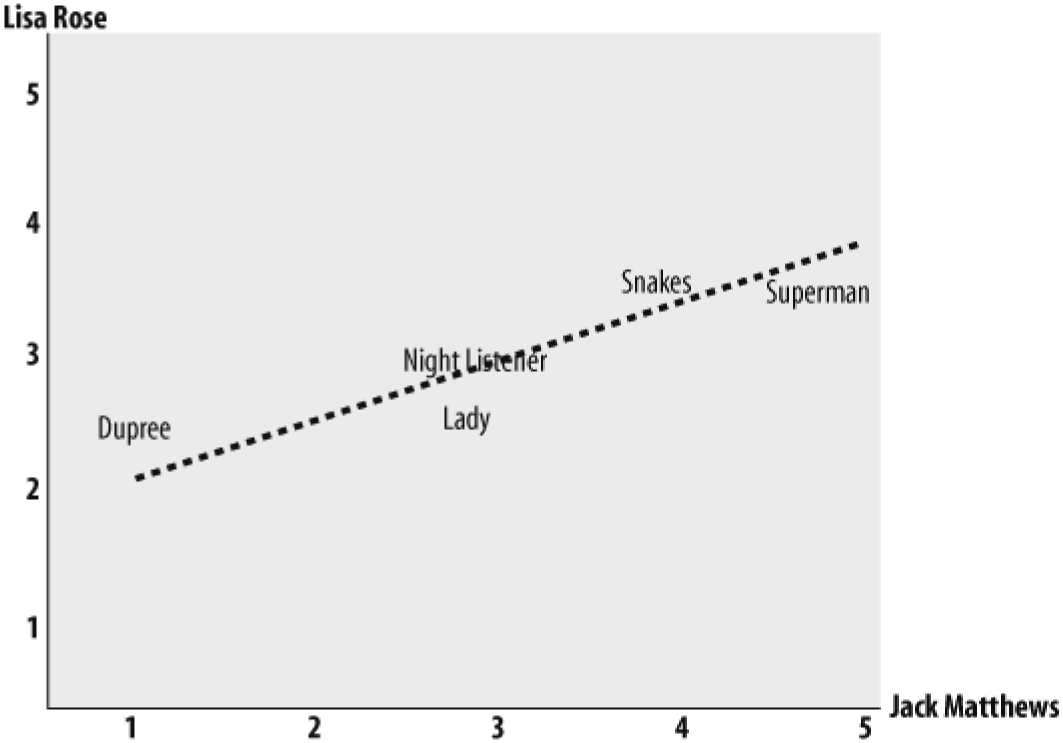


New similarity measure: Pearson Correlation Score

The correlation coefficient is a measure of how well two sets of data fit on a straight line.



Pearson correlation vs. Euclidean distance



Two critics with a high correlation score.

- Corrects for grade inflation.
  - E.g., Jack Matthews tends to give higher scores than Lisa Rose, but the line still fits because they have relatively similar preferences.
- Euclidean distance score will say they are quite dissimilar – far away

Formula:
Pearson coefficient of a *sample*

$$r = \frac{\overbrace{\sum_{i=1}^{N}(A_i - \bar{A})(B_i - \bar{B})}^{\text{Covariance: how both variables vary}}}{\underbrace{\sqrt{\sum_{i=1}^{N}(Ai - \bar{A})^2}}_{\text{Variance of A}}\ \underbrace{\sqrt{\sum_{i=1}^{N}(Bi - \bar{B})^2}}_{\text{Variance of B}}}$$

The correlation ranges from
-1 (perfect negative correlation) to
1 (perfect positive correlation).
0 – no correlation.

Automated recommender system

- Build a customer profile
- Compare the new customer's profile with the profiles of other customers, locate similar "neighbors"
- Predict the rating that the new customer would give to items he has not yet rated by combining ratings of a peer group selected by similar tastes
- Rank predictions and output top-scored ones

Ranking the critics

- Using Pearson similarity score, rank all critics vs. Active User
- The top-K list of similar users is exactly K nearest neighbors
- But instead of predicting a score for a single item, we get the scores for all items which similar users ranked

Top similar users for ‘Toby’, K=5:

[(0.99124070716192991, 'Lisa Rose'),
(0.92447345164190486, 'Mick LaSalle'),
(0.89340514744156474, 'Claudia Puig')]

Toby should trust Lisa Ross, she has a similar taste

Automated recommender system

- Build a customer profile
- Compare the new customer’s profile with the profiles of other customers, locate similar “neighbors”
- Predict the rating that the new customer would give to items he has not yet rated by combining ratings of a peer group selected by similar tastes
- Rank predictions and output top-scored ones

Ranked list of recommendations

Example: recommendations for Toby

5 nearest neighbors with similar tastes

Critic	Similarity	Night	S.xNight	Lady	S.xLady	Luck	S.xLuck
Rose	0.99	3.0	2.97	2.5	2.48	3.0	2.97
Seymour	0.38	3.0	1.14	3.0	1.14	1.5	0.57
Puig	0.89	4.5	4.02			3.0	2.68
LaSalle	0.92	3.0	2.77	3.0	2.77	2.0	1.85
Matthews	0.66	3.0	1.99	3.0	1.99		
Total			12.89		8.38		8.07
Sim.Sum			3.84		2.95		3.18
Total/Sim. Sum			3.35		2.83		2.53

Ranked list of recommendations

Collect all rankings from similar users

Ranks for movies which Toby did not see yet

Critic	Similarity	Night	S.xNight	Lady	S.xLady	Luck	S.xLuck
Rose	0.99	3.0	2.97	2.5	2.48	3.0	2.97
Seymour	0.38	3.0	1.14	3.0	1.14	1.5	0.57
Puig	0.89	4.5	4.02			3.0	2.68
LaSalle	0.92	3.0	2.77	3.0	2.77	2.0	1.85
Matthews	0.66	3.0	1.99	3.0	1.99		
Total			12.89		8.38		8.07
Sim.Sum			3.84		2.95		3.18
Total/Sim. Sum			3.35		2.83		2.53

Ranked list of recommendations

Compute predicted rank for each movie: weighted average

Movie rating multiplied by similarity

Critic	Similarity	Night	S.xNig	Lady	S.xLady	Luck	S.xLuck
Rose	0.99	3.0	2.97	2.5	2.48	3.0	2.97
Seymour	0.38	3.0	1.14	3.0	1.14	1.5	0.57
Puig	0.89	4.5	4.02			3.0	2.68
LaSalle	0.92	3.0	2.77	3.0	2.77	2.0	1.85
Matthews	0.66	3.0	1.99	3.0	1.99		
Total			12.89		8.38		8.07
Sim.Sum			3.84		2.95		3.18
Total/Sim. Sum			3.35		2.83		2.53

Ranked list of recommendations

Compute predicted rank for each movie:
total weighted sum

Critic	Similarity	Night	S.xNight	Lady	S.xLady	Luck	S.xLuck
Rose	0.99	3.0	2.97	2.5	2.48	3.0	2.97
Seymour	0.38	3.0	1.14	3.0	1.14	1.5	0.57
Puig	0.89	4.5	4.02			3.0	2.68
LaSalle	0.92	3.0	2.77	3.0	2.77	2.0	1.85
Matthews	0.66	3.0	1.99	3.0	1.99		
Total			12.89		8.38		8.07
Sim.Sum			3.84		2.95		3.18
Total/Sim. Sum			3.35		2.83		2.53

Ranked list of recommendations

Compute predicted rank for each movie:
total weighted sum

Critic	Similarity	Night	S.xNight	Lady	S.xLady	Luck	S.xLuck
Rose	0.99	3.0	2.97	2.5	2.48	3.0	2.97
Seymour	0.38	3.0	1.14	3.0	1.14	1.5	0.57
Puig	0.89	4.5	4.02			3.0	2.68
LaSalle	0.92	3.0	2.77	3.0	2.77	2.0	1.85
Matthews	0.66	3.0	1.99	3.0	1.99		
Total			12.89		8.38		8.07
Sim.Sum			3.84		2.95		3.18
Total/Sim. Sum			3.35		2.83		2.53

Total weight of critics participated in the score

Ranked list of recommendations

Normalize each ranking

Critic	Similarity	Night	S.xNight	Lady	S.xLady	Luck	S.xLuck
Rose	0.99	3.0	2.97	2.5	2.48	3.0	2.97
Seymour	0.38	3.0	1.14	3.0	1.14	1.5	0.57
Puig	0.89	4.5	4.02			3.0	2.68
LaSalle	0.92	3.0	2.77	3.0	2.77	2.0	1.85
Matthews	0.66	3.0	1.99	3.0	1.99		
Total			12.89		8.38		8.07
Sim.Sum			3.84		2.95		3.18
Total/Sim. Sum			3.35		2.83		2.53

Total normalized by the weight of users participating in this recommendation

Ranked list of recommendations

Sort descending and output rankings with maximum score

Critic	Similarity	Night	S.xNight	Lady	S.xLady	Luck	S.xLuck
Rose	0.99	3.0	2.97	2.5	2.48	3.0	2.97
Seymour	0.38	3.0	1.14	3.0	1.14	1.5	0.57
Puig	0.89	4.5	4.02			3.0	2.68
LaSalle	0.92	3.0	2.77	3.0	2.77	2.0	1.85
Matthews	0.66	3.0	1.99	3.0	1.99		
Total			12.89		8.38		8.07
Sim.Sum			3.84		2.95		3.18
Total/Sim. Sum			3.35		2.83		2.53

Recommendations for Toby