F20DL / F21DL Data Mining and Machine Learning
Lecture 9
Attribute (feature) selection

Diana Bental
Ekaterina Komendantskaya
(with material from David Corne)

Attribute selection

- You should be able to
  - Decide when to apply attribute selection
  - Choose suitable methods
  - Evaluate the results
- Coursework 1

Attribute (feature) selection – *what*?

- You have some data, and you want to use it to build a classifier so that you can predict something (e.g. likelihood of cancer)
- The data has **10 000** attributes (features)
- You need to cut it down to **1 000** fields before you try machine learning. Which 1,000?
- The process of choosing the 1,000 fields to use is called **Attribute Selection** (or **Feature Selection)**

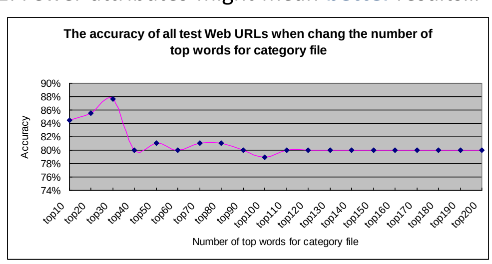
Attribute selection – *why*?

Attribute selection – *why*?

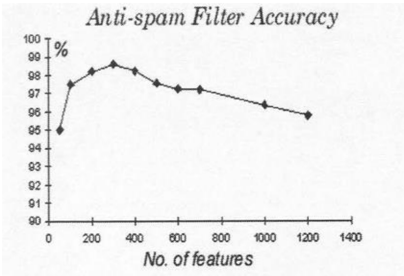
1. Datasets with many attributes
   - Gene expression datasets (~10,000 attributes)
     - http://www.ncbi.nlm.nih.gov/sites/entrez?db=gds
   - Proteomics (proteins and peptides) datasets (~20,000 attributes)
     - http://www.ebi.ac.uk/pride/
   - Satellite data
   - Weather data …. etc
- Computationally expensive

Attribute selection – *why*?

2. Fewer attributes might mean **better** results…



Attribute selection - *why?*



From http://elpub.scix.net/data/works/att/02-28.content.pdf

Attribute selection - *why?*

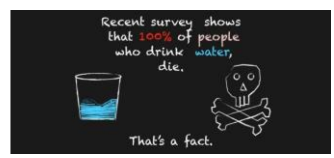
- Quite easy to find lots more cases from papers, where experiments show that accuracy **reduces** when you use more attributes

Attribute selection - *why?*

- Why does accuracy reduce with more attributes?
- Suppose the best attribute set has 20 attributes. If you add another 5 attributes, typically the accuracy of machine learning may reduce. But you still have the original 20 attributes!! Why does this happen???

Noise / Spurious Correlations / Explosion

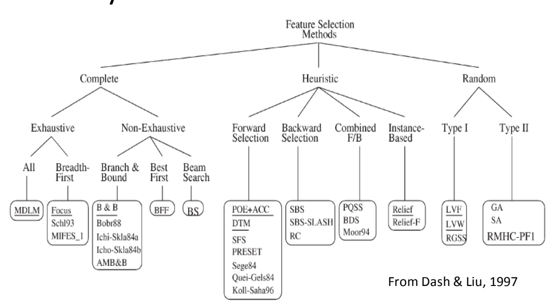
- The additional features typically add **noise**.
- Machine learning will pick up on **spurious correlations**, that might be true in the training set, but not in the test set.
- For some ML methods, more attributes means **more parameters to learn** (more NN weights, more decision tree nodes, etc...) – the increased space of possibilities is more difficult to search.



Attribute selection - *why?*

- So: Aim to remove attributes that are redundant, irrelevant, only weakly relevant

Many attribute selection methods....



From Dash & Liu, 1997

Method types: Filter Methods and Wrapper Methods

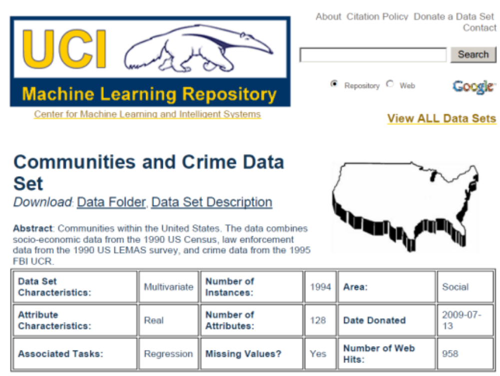
- Filter methods
  - Use correlation, or information gain, or some other method, to pick some attributes
  - Then run the machine learning method on just those attributes
- Wrapper methods
  - Choose some attributes
  - Run the machine learning method
  - Is it good enough?
  - If so then stop.
  - If not then change the set of attributes
  - And repeat.

Filter Method: Correlation based ranking

- It is used often by practitioners
- It is fine for certain datasets
- (Not considered in Dash and Liu's survey at all)

e.g. An interesting data set…

- The Communities and Crime dataset (C&C)



-- **state**: US state (by number) -
-- **county**: numeric code for county
-- **community**: numeric code for community -
-- **communityname**: community name –
-- **fold**: fold number for non-random 10 fold cross validation,
-- **population**: population for community: (numeric - decimal)
-- **householdsize**: mean people per household (numeric - decimal)
-- **agePct12t21**: percentage of population that is 12-21 in age (numeric - decimal)
-- **agePct12t29**: percentage of population that is 12-29 in age (numeric - decimal)
-- **agePct16t24**: percentage of population that is 16-24 in age (numeric - decimal)
-- **agePct65up**: percentage of population that is 65 and over in age (numeric - decimal)
-- **numbUrban**: number of people living in areas classified as urban (numeric - decimal)
-- **pctUrban**: percentage of people living in areas classified as urban (numeric - decimal)
-- **medIncome**: median household income (numeric - decimal) –
-- **pctWWage**: percentage of households with wage or salary income in 1989 (numeric - decimal)
-- **pctWFarmSelf**: percentage of households with farm or self employment income in 1989
[etc etc etc --- 128 fields altogether]

-- **ViolentCrimesPerPop**: total number of violent crimes per 100K population (numeric - decimal) class attribute (to be predicted)

.. About 2000 instances

Recap: Correlation

- Remember how to calculate r
- If we have pairs of values (x, y) Pearson's *r* is

$$\frac{1}{(n-1)} \cdot \sum_{(x,y)\in Sample} \frac{(x-\mu_x)}{std_x} \cdot \frac{(y-\mu_y)}{std_y}$$

- *r* between -1 and 1, 0 = no correlation

The names file in the C&C dataset has correlation values (with the class attribute) for each attribute

	min	Max	mean	std	correlation	median	mode
Population	0	1	0.06	0.13	0.37	0.02	0.01
Householdsize	0	1	0.46	0.16	-0.03	0.44	0.41
agePct12t21	0	1	0.42	0.16	0.06	0.4	0.38
agePct12t29	0	1	0.49	0.14	0.15	0.48	0.49
agePct16t24	0	1	0.34	0.17	0.1	0.29	0.29
agePct65up	0	1	0.42	0.18	0.07	0.42	0.47
numbUrban	0	1	0.06	0.13	0.36	0.03	0
pctUrban	0	1	0.7	0.44	0.08	1	1
medIncome	0	1	0.36	0.21	-0.42	0.32	0.23
pctWWage	0	1	0.56	0.18	-0.31	0.56	0.58
pctWFarmSelf	0	1	0.29	0.2	-0.15	0.23	0.16
pctWInvInc	0	1	0.5	0.18	-0.58	0.48	0.41
pctWSocSec	0	1	0.47	0.17	0.12	0.475	0.56
pctWPubAsst	0	1	0.32	0.22	0.57	0.26	0.1
pctWRetire	0	1	0.48	0.17	-0.1	0.47	0.44
medFamInc	0	1	0.38	0.2	-0.44	0.33	0.25
perCapInc	0	1	0.35	0.19	-0.35	0.3	0.23
NumUnderPov	0	1	0.06	0.13	0.45	0.02	0.01
PctPopUnderPov	0	1	0.3	0.23	0.52	0.25	0.08
PctLess9thGrade	0	1	0.32	0.21	0.41	0.27	0.19

here …

	min	Max	mean	std	correlation	median	mode
Population	0	1	0.06	0.13	0.37	0.02	0.01
Householdsize	0	1	0.46	0.16	-0.03	0.44	0.41
agePct12t21	0	1	0.42	0.16	0.06	0.4	0.38
agePct12t29	0	1	0.49	0.14	0.15	0.48	0.49
agePct16t24	0	1	0.34	0.17	0.1	0.29	0.29
agePct65up	0	1	0.42	0.18	0.07	0.42	0.47
numbUrban	0	1	0.06	0.13	0.36	0.03	0
pctUrban	0	1	0.7	0.44	0.08	1	1
medIncome	0	1	0.36	0.21	-0.42	0.32	0.23
pctWWage	0	1	0.56	0.18	-0.31	0.56	0.58
pctWFarmSelf	0	1	0.29	0.2	-0.15	0.23	0.16
pctWInvInc	0	1	0.5	0.18	-0.58	0.48	0.41
pctWSocSec	0	1	0.47	0.17	0.12	0.475	0.56
pctWPubAsst	0	1	0.32	0.22	0.57	0.26	0.1
pctWRetire	0	1	0.48	0.17	-0.1	0.47	0.44
medFamInc	0	1	0.38	0.2	-0.44	0.33	0.25
perCapInc	0	1	0.35	0.19	-0.35	0.3	0.23
NumUnderPov	0	1	0.06	0.13	0.45	0.02	0.01
PctPopUnderPov	0	1	0.3	0.23	0.52	0.25	0.08
PctLess9thGrade	0	1	0.32	0.21	0.41	0.27	0.19

Here are the top 10 (although the first doesn't count) - this hints at how we might use correlation for **attribute selection**

ViolentCrimesPerPop	0	1	0.24	0.23	1	0.15	0.03	0
PctIlleg	0	1	0.25	0.23	0.74	0.17	0.09	0
PctKids2Par	0	1	0.62	0.21	-0.74	0.64	0.72	0
PctFam2Par	0	1	0.61	0.2	-0.71	0.63	0.7	0
PctYoungKids2Par	0	1	0.66	0.22	-0.67	0.7	0.91	0
PctTeen2Par	0	1	0.58	0.19	-0.66	0.61	0.6	0
pctWPubAsst	0	1	0.32	0.22	0.57	0.26	0.1	0
FemalePctDiv	0	1	0.49	0.18	0.56	0.5	0.54	0
TotalPctDiv	0	1	0.49	0.18	0.55	0.5	0.57	0
MalePctDivorce	0	1	0.46	0.18	0.53	0.47	0.56	0

But…

- Can anyone see a potential problem with choosing only (for example) the 10 attributes that correlate best with the target class ?
- So .. Look for attributes which correlate **highly** with the class attribute and do **not** correlate with each other
- Other filter methods:
  - Entropy/Information gain methods.
  - Build a decision tree and reject the unused attributes, then use nearest neighbour
  - Relief method – see end slides if interested

Filter methods

- First choose the attributes, and *then* apply machine learning
- Use a heuristic (clever guessing method) to decide which attributes are *likely* to be best
- Can use different kinds of machine learning methods afterwards
- Fast – not iterative, chooses attributes without repeatedly running the machine learning method
- May overfit to the data
- Tend to select big subsets - may select the full attribute set as the "best" set
- Filter methods generally look at attributes individually

A made-up dataset

f1	f2	f3	f4	…	class
0.4	0.6	0.4	0.6		1
0.2	0.4	1.6	-0.6		1
0.5	0.7	1.8	-0.8		1
0.7	0.8	0.2	0.9		2
0.9	0.8	1.8	-0.7		2
0.5	0.5	0.6	0.5		2

Correlated with the class

f1	f2	f3	f4	…	class
0.4	0.6	0.4	0.6		1
0.2	0.4	1.6	-0.6		1
0.5	0.7	1.8	-0.8		1
0.7	0.8	0.2	0.9		2
0.9	0.8	1.8	-0.7		2
0.5	0.5	0.6	0.5		2

uncorrelated with the class / seemingly random

f1	f2	f3	f4	…	class
0.4	0.6	0.4	0.6		1
0.2	0.4	1.6	-0.6		1
0.5	0.7	1.8	-0.8		1
0.7	0.8	0.2	0.9		2
0.9	0.8	1.8	-0.7		2
0.5	0.5	0.6	0.5		2

So correlation based attribute selection reduces the dataset to this.

f1	f2			…	class
0.4	0.6				1
0.2	0.4				1
0.5	0.7				1
0.7	0.8				2
0.9	0.8				2
0.5	0.5				2

But, col 5 shows us f3 + f4 – which is perfectly correlated with the class!

f1	f2	f3	f4	f5	class
0.4	0.6	0.4	0.6	1	1
0.2	0.4	1.6	-0.6	1	1
0.5	0.7	1.8	-0.8	1	1
0.7	0.8	0.2	0.9	1.1	2
0.9	0.8	1.8	-0.7	1.1	2
0.5	0.5	0.6	0.5	1.1	2

That example is cheating a bit…

- Adding a new attribute based on existing attributes is **feature extraction**
- It would be very hard to guess that we should add those two attributes together
- But feature extraction is a common operation in e.g. image processing
  - Look for edges, surfaces, eyes... Before interpreting the image as a whole

- So… Need to consider how well attributes work *together*

`Complete' methods

- Original dataset has N attributes
- You want to use a subset of *k* attributes
- A *complete* attribute selection method means: try every subset of *k* attributes, and choose the best!
  - the number of subsets is N! / *k*!(N−*k*)!
  - what is this when N is 100 and *k* is 5?
  - 75,287,520 -- almost nothing

`Complete' methods

- Original dataset has N attributes
- You want to use a subset of *k* attributes
- A *complete* attribute selection method means: try every subset of *k* attributes, and choose the best!
  - the number of subsets is N! / *k*!(N−*k*)!
  - what is this when N is 10 000 and *k* is 100?

`Complete' methods

- Original dataset has N attributes
- You want to use a subset of k attributes
- A *complete* attribute selection method means: try every subset of *k* attributes, and choose the best!
  - the number of subsets is N! / *k*!(N−*k*)!
  - what is this when N is 10 000 and *k* is 100?
  - 5,000,000,000,000,000,000,000,000,000,000,

000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,

Continued 2 …..

000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,

Continued 3 …..

000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,

Continued 4 .....

… continued for another 114 slides.

Actually it is around $5 \times 10^{35,101}$

(there are around 10^{80} atoms in the universe)

`Complete' methods

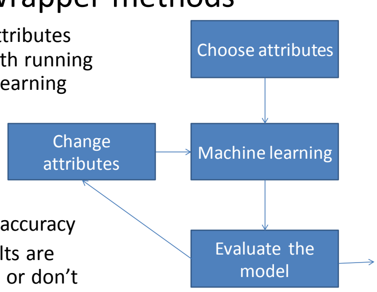
- Can you see a problem with complete methods?

Stochastic filter methods

- One way to try to find a good subset is to run a stochastic search algorithm
  - e.g. Hillclimbing, simulated annealing, genetic algorithm, particle swarm optimisation, …

Wrapper methods

- Choose the attributes alternately with running the machine learning method



- Compare the accuracy
- Until the results are good enough, or don't get better

`Forward' wrapper methods

These methods `grow' a set S of attributes –

1. S starts empty
2. Find the best attribute to add (by checking each attribute in turn to see which one gives best performance on a test set when combined with S).
3. If overall performance has improved, return to step 2; else stop

Forward selection illustrated

F1	F2	F3	F4	F5	F6	F7	F8	F9	Etc…
2	5	65	67	2	2	12	2	234	…
1	2	4	5	13	1	1	43	12	…
4	3	43	2	4	6	2	2	1	…
5	4	2	3	5	5	13	1	2	…
3	5	1	4	7	3	4	6	13	…
2	2	6	5	7	1	5	4	4	…
1	3	4	4	55	4	7	55	43	…

Selected attribute set { }

Run ML with each attribute in turn to find out which works best with current attribute set …

F1	F2	F3	F4	F5	F6	F7	F8	F9	Etc…
2	5	65	67	2	2	12	2	234	…
1	2	4	5	13	1	1	43	12	…
4	3	43	2	4	6	2	2	1	…
5	4	2	3	5	5	13	1	2	…
3	5	1	4	7	3	4	6	13	…
2	2	6	5	7	1	5	4	4	…
1	3	4	4	55	4	7	55	43	…

Selected attribute set { }

Test each attribute in turn to find out which works best with current attribute set …

F1	F2	F3	F4	F5	F6	F7	F8	F9	Etc…
2	5	65	67	2	2	12	2	234	…
1	2	4	5	13	1	1	43	12	…
4	3	43	2	4	6	2	2	1	…
5	4	2	3	5	5	13	1	2	…
3	5	1	4	7	3	4	6	13	…
2	2	6	5	7	1	5	4	4	…
1	3	4	4	55	4	7	55	43	…
65%									

Selected attribute set { }

Test each attribute in turn to find out which works best with current attribute set …

F1	F2	F3	F4	F5	F6	F7	F8	F9	Etc…
2	5	65	67	2	2	12	2	234	…
1	2	4	5	13	1	1	43	12	…
4	3	43	2	4	6	2	2	1	…
5	4	2	3	5	5	13	1	2	…
3	5	1	4	7	3	4	6	13	…
2	2	6	5	7	1	5	4	4	…
1	3	4	4	55	4	7	55	43	…
	58%								

Selected attribute set { }

Test each attribute in turn to find out which works best with current attribute set …

F1	F2	F3	F4	F5	F6	F7	F8	F9	Etc…
2	5	65	67	2	2	12	2	234	…
1	2	4	5	13	1	1	43	12	…
4	3	43	2	4	6	2	2	1	…
5	4	2	3	5	5	13	1	2	…
3	5	1	4	7	3	4	6	13	…
2	2	6	5	7	1	5	4	4	…
1	3	4	4	55	4	7	55	43	…
		54%							

Selected attribute set { }

Test each attribute in turn to find out which works best with current attribute set …

F1	F2	F3	F4	F5	F6	F7	F8	F9	Etc…
2	5	65	67	2	2	12	2	234	…
1	2	4	5	13	1	1	43	12	…
4	3	43	2	4	6	2	2	1	…
5	4	2	3	5	5	13	1	2	…
3	5	1	4	7	3	4	6	13	…
2	2	6	5	7	1	5	4	4	…
1	3	4	4	55	4	7	55	43	…
			72%						

Selected attribute set { }

Test each attribute in turn to find out which works best with current attribute set …

F1	F2	F3	F4	F5	F6	F7	F8	F9	Etc…
2	5	65	67	2	2	12	2	234	…
1	2	4	5	13	1	1	43	12	…
4	3	43	2	4	6	2	2	1	…
5	4	2	3	5	5	13	1	2	…
3	5	1	4	7	3	4	6	13	…
2	2	6	5	7	1	5	4	4	…
1	3	4	4	55	4	7	55	43	…
				64%					

Selected attribute set {}

Etc

F1	F2	F3	F4	F5	F6	F7	F8	F9	Etc…
2	5	65	67	2	2	12	2	234	…
1	2	4	5	13	1	1	43	12	…
4	3	43	2	4	6	2	2	1	…
5	4	2	3	5	5	13	1	2	…
3	5	1	4	7	3	4	6	13	…
2	2	6	5	7	1	5	4	4	…
1	3	4	4	55	4	7	55	43	…
65%	58%	54%	72%	64%	61%	62%	25%	49%	….

Selected attribute set {}

Add the winning attribute to the selected attribute set

F1	F2	F3	F4	F5	F6	F7	F8	F9	Etc…
2	5	65	67	2	2	12	2	234	…
1	2	4	5	13	1	1	43	12	…
4	3	43	2	4	6	2	2	1	…
5	4	2	3	5	5	13	1	2	…
3	5	1	4	7	3	4	6	13	…
2	2	6	5	7	1	5	4	4	…
1	3	4	4	55	4	7	55	43	…
65%	58%	54%	**72%**	64%	61%	62%	25%	49%	….

Selected attribute set {**F4**}

We have completed **one 'round'** of forward selection

F1	F2	F3	F4	F5	F6	F7	F8	F9	Etc…
2	5	65	67	2	2	12	2	234	…
1	2	4	5	13	1	1	43	12	…
4	3	43	2	4	6	2	2	1	…
5	4	2	3	5	5	13	1	2	…
3	5	1	4	7	3	4	6	13	…
2	2	6	5	7	1	5	4	4	…
1	3	4	4	55	4	7	55	43	…
65%	58%	54%	**72%**	64%	61%	62%	25%	49%	….

Selected attribute set {**F4**}

Test each attribute in turn to find out which works best with current attribute set …

F1	F2	F3	F4	F5	F6	F7	F8	F9	Etc…
2	5	65	67	2	2	12	2	234	…
1	2	4	5	13	1	1	43	12	…
4	3	43	2	4	6	2	2	1	…
5	4	2	3	5	5	13	1	2	…
3	5	1	4	7	3	4	6	13	…
2	2	6	5	7	1	5	4	4	…
1	3	4	4	55	4	7	55	43	…

Selected attribute set {**F4**}

Test each attribute in turn to find out which works best with current attribute set …

F1	F2	F3	F4	F5	F6	F7	F8	F9	Etc…
2	5	65	67	2	2	12	2	234	…
1	2	4	5	13	1	1	43	12	…
4	3	43	2	4	6	2	2	1	…
5	4	2	3	5	5	13	1	2	…
3	5	1	4	7	3	4	6	13	…
2	2	6	5	7	1	5	4	4	…
1	3	4	4	55	4	7	55	43	…
61%									

Selected attribute set {**F4**}

Test each attribute in turn to find out which works best with current attribute set …

F1	F2	F3	F4	F5	F6	F7	F8	F9	Etc…
2	5	65	67	2	2	12	2	234	…
1	2	4	5	13	1	1	43	12	…
4	3	43	2	4	6	2	2	1	…
5	4	2	3	5	5	13	1	2	…
3	5	1	4	7	3	4	6	13	…
2	2	6	5	7	1	5	4	4	…
1	3	4	4	55	4	7	55	43	…
	59%								

Selected attribute set {**F4**}

Test each attribute in turn to find out which works best with current attribute set …

F1	F2	F3	F4	F5	F6	F7	F8	F9	Etc…
2	5	65	67	2	2	12	2	234	…
1	2	4	5	13	1	1	43	12	…
4	3	43	2	4	6	2	2	1	…
5	4	2	3	5	5	13	1	2	…
3	5	1	4	7	3	4	6	13	…
2	2	6	5	7	1	5	4	4	…
1	3	4	4	55	4	7	55	43	…
		58%							

Selected attribute set {**F4**}

Test each attribute in turn to find out which works best with current attribute set …

F1	F2	F3	F4	F5	F6	F7	F8	F9	Etc…
2	5	65	67	2	2	12	2	234	…
1	2	4	5	13	1	1	43	12	…
4	3	43	2	4	6	2	2	1	…
5	4	2	3	5	5	13	1	2	…
3	5	1	4	7	3	4	6	13	…
2	2	6	5	7	1	5	4	4	…
1	3	4	4	55	4	7	55	43	…
				66%					

Selected attribute set {**F4**}

Etc

F1	F2	F3	F4	F5	F6	F7	F8	F9	Etc…
2	5	65	67	2	2	12	2	234	…
1	2	4	5	13	1	1	43	12	…
4	3	43	2	4	6	2	2	1	…
5	4	2	3	5	5	13	1	2	…
3	5	1	4	7	3	4	6	13	…
2	2	6	5	7	1	5	4	4	…
1	3	4	4	55	4	7	55	43	…
61%	59%	58%		66%	68%	75%	47%	49%	….

Selected attribute set {**F4**}

Add the winning attribute to the selected attribute set

F1	F2	F3	F4	F5	F6	F7	F8	F9	Etc…
2	5	65	67	2	2	12	2	234	…
1	2	4	5	13	1	1	43	12	…
4	3	43	2	4	6	2	2	1	…
5	4	2	3	5	5	13	1	2	…
3	5	1	4	7	3	4	6	13	…
2	2	6	5	7	1	5	4	4	…
1	3	4	4	55	4	7	55	43	…
61%	59%	58%		66%	68%	75%	47%	49%	….

Selected attribute set {**F4, F7**}

We have completed the second ‘**round’** of forward selection

F1	F2	F3	F4	F5	F6	F7	F8	F9	Etc…
2	5	65	67	2	2	12	2	234	…
1	2	4	5	13	1	1	43	12	…
4	3	43	2	4	6	2	2	1	…
5	4	2	3	5	5	13	1	2	…
3	5	1	4	7	3	4	6	13	…
2	2	6	5	7	1	5	4	4	…
1	3	4	4	55	4	7	55	43	…
61%	59%	58%		66%	68%	75%	47%	49%	….

Selected attribute set {**F4, F7**}

Continue…

adding one attribute after each round,
until overall accuracy starts to reduce

F1	F2	F3	F4	F5	F6	F7	F8	F9	Etc…
2	5	65	67	2	2	12	2	234	…
1	2	4	5	13	1	1	43	12	…
4	3	43	2	4	6	2	2	1	…
5	4	2	3	5	5	13	1	2	…
3	5	1	4	7	3	4	6	13	…
2	2	6	5	7	1	5	4	4	…
1	3	4	4	55	4	7	55	43	…
61%	59%	58%		66%	68%	75%	47%	49%	….

Selected attribute set {**F4, F7**}

`Backward' methods

These methods **remove** attributes one by one.

1. S starts with the **full attribute set**
2. Find the best feature to *remove* (by checking which removal from S gives best performance on a test set).
3. If overall performance has improved, return to step 2; else stop

- Forward and backward are heuristic (clever guess) methods
  - Neither forward nor backward are guaranteed to give the best set of attributes
- When to choose forward instead of backward?

About wrapper methods

- Accurate – give good accuracy with a specific classification method (but may be the wrong set for a different method)
- Avoid over-fitting by running machine learning with cross-validation
- Slow - need to build and test the model for every subset of features

Take away…

- Too many attributes may mean *less* accurate ML
  - Even if the attributes are relevant
- Attribute selection is difficult, with many different methods and algorithms
- Can use machine learning methods to select attributes
  - Can *filter* the attributes beforehand and then apply machine learning
  - Or *wrap* attribute selection in with the learning

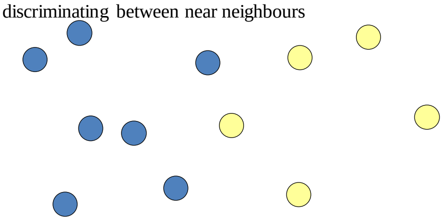
For more…

- Data Mining and Machine Learning
  - Section 7.1 Attribute Selection
- Practical overview
  - https://machinelearningmastery.com/perform-feature-selection-machine-learning-data-weka/
- Papers
  - Kira and Rendell AAAI 1992 "The Feature Selection Problem: Traditional Methods and a New Algorithm"
  - Dash and Liu IDA 1997 "Feature selection for classification"

Relief is the classic example of an instance-based heuristic filter method

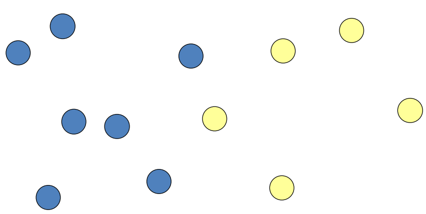
The Relief method

An *instance-based, heuristic* (filter) method – it works out weight values for each feature, based on how important they *seem* to be in discriminating between near neighbours



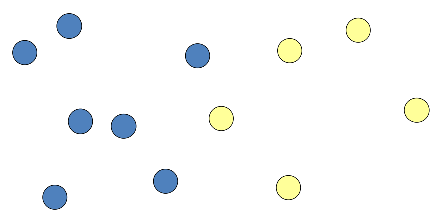
The Relief method

There are two attributes here – the x and the y co-ordinate
Initially they each have zero weight: wx = 0; wy = 0;



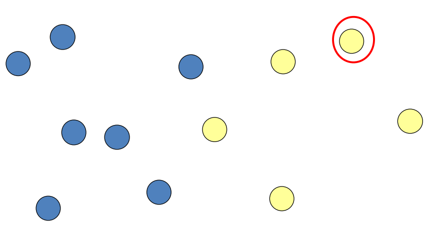
The Relief method

wx = 0; wy = 0; choose an instance at random



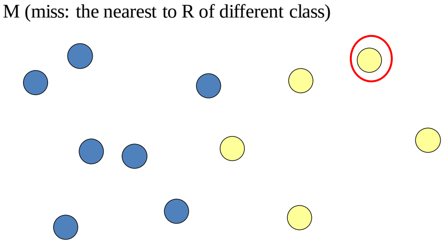
The Relief method

wx = 0; wy = 0; choose an instance at random, call it R



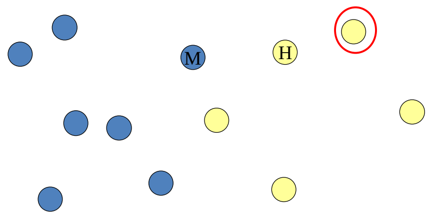
The Relief method

wx = 0; wy = 0; find H (hit: the nearest to R of the same class) and M (miss: the nearest to R of different class)



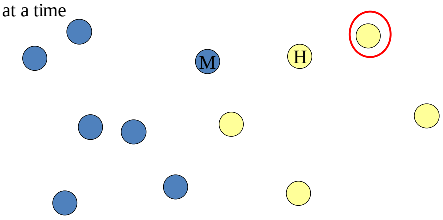
The Relief method

wx = 0; wy = 0; find H (hit: the nearest to R of the same class) and M (miss: the nearest to R of different class)



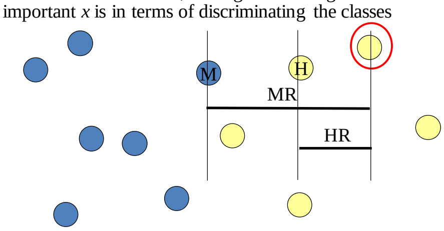
The Relief method

wx = 0; wy = 0; now we update the weights based on the distances between R and H and between R and M. This happens one feature at a time



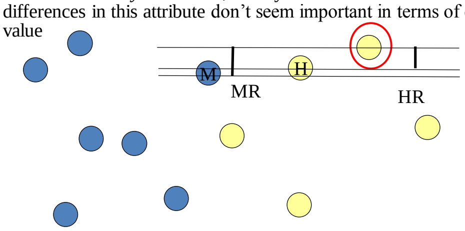
The Relief method

To change wx, we add to it: (MR – HR)/*n* ; so, the further the `miss' in the *x* direction, the higher the weight of *x* – the more important *x* is in terms of discriminating the classes



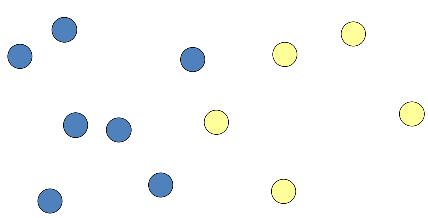
The Relief method

To change wy, we add to it: (MR – HR)/*n* again, but this time calculated in the *y* dimension; clearly the difference is smaller; differences in this attribute don't seem important in terms of class value



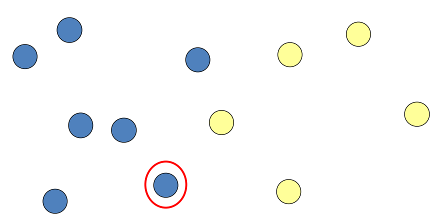
The Relief method

Maybe now we have wx = 0.07, wy = 0.002.



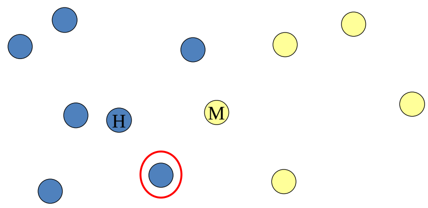
The Relief method

wx = 0.07, wy = 0.002;
Pick another instance at random, and do the same again.



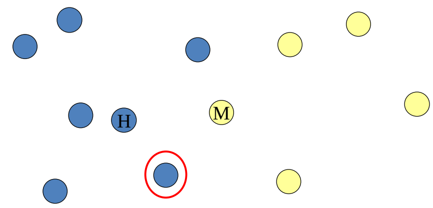
The Relief method

wx = 0.07, wy = 0.002;
Identify H and M



The Relief method

wx = 0.07, wy = 0.002;
Add the HR and MR differences divided by *n*, for each attribute, again …



The Relief method

- In the end, there is a weight value for each attribute. The higher the value, the more relevant the attribute.
- We can use these weights for attribute selection, simply by choosing the attributes with the *S* highest weights (if we want to use *S* attributes)
- Once we have chosen *S* attributes, we can use a classifier with them.
- Notes:
  - Relief works for numeric min-max normalized data with two classes, and for nominal data (using a Hamming distance)
  - Need to extend it for multiple classes… how?
  - Why divide by *n?* Then the weight values can be interpreted as a *difference in probabilities*.
  - Doesn't work well if there isn't enough training data

The Relief method, plucked directly from the original paper (Kira and Rendell 1992)

```
Relief(S, m, τ)
    Separate S into S+ = {positive instances} and
        S- = {negative instances}
    W = (0, 0, . . . , 0)
    For i = 1 to m
        Pick at random an instance X ∈ S
        Pick at random one of the positive instances
            closest to X, Z+ ∈ S+
        Pick at random one of the negative instances
            closest to X, Z- ∈ S-
        if (X is a positive instance)
            then    Near-hit = Z+; Near-miss = Z-
            else    Near-hit = Z-; Near-miss = Z+
        update-weight(W, X, Near-hit, Near-miss)
    Relevance = (1/m)W
    For i = 1 to p
        if (relevance_i ≥ t)
            then    f_i is a relevant feature
            else    f_i is an irrelevant feature

update-weight(W, X, Near-hit, Near-miss)
    For i = 1 to p
        W_i = W_i - diff(x_i, near-hit_i)^2 + diff(x_i, near-miss_i)^2
```