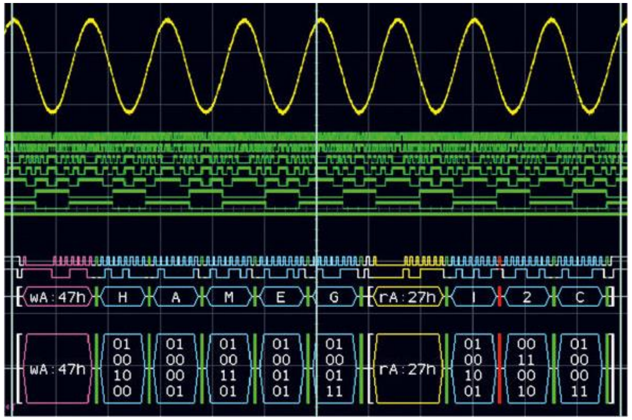
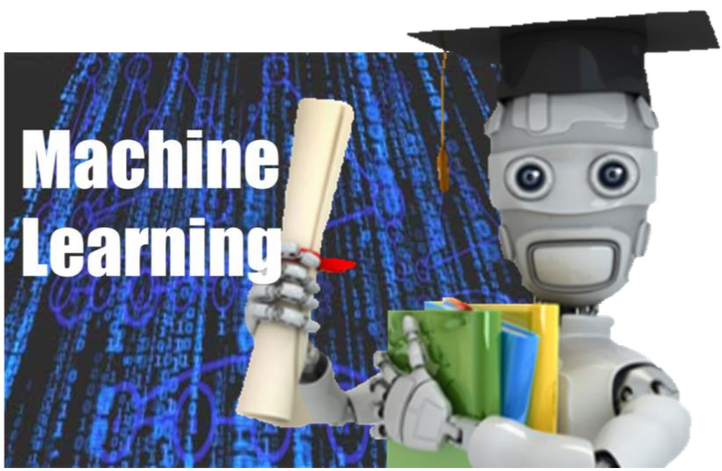
A practical overview of data analysis methodology

Dr Athanasios Tsanas ('Thanasis')

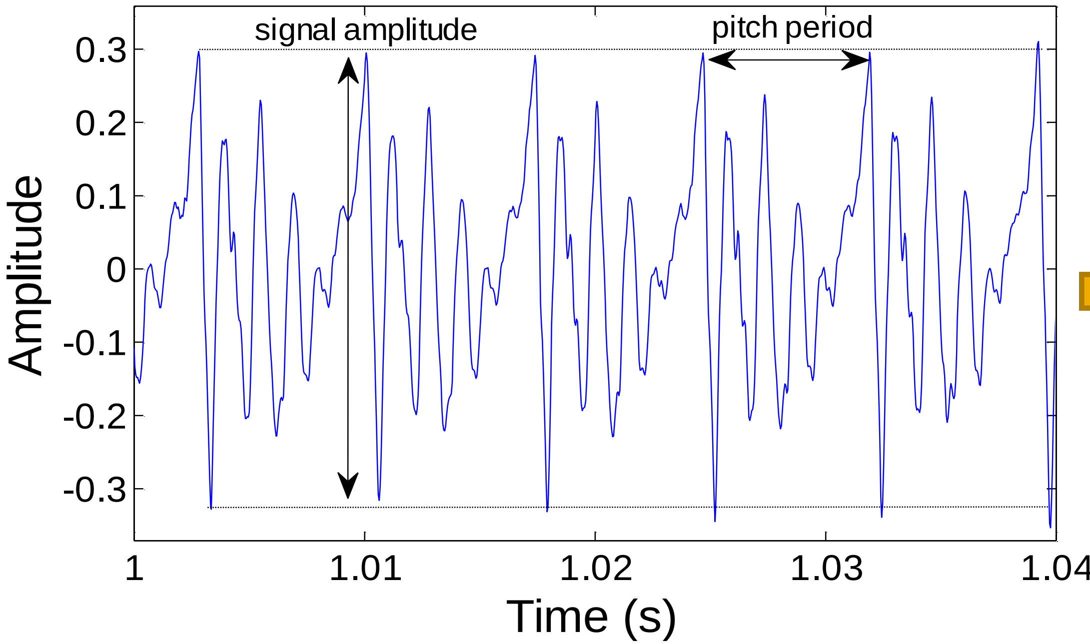
Oxford Centre for Industrial and Applied Mathematics
Institute of Biomedical Engineering, Dept. of Engineering Science
Sleep and Circadian Neuroscience Institute, Dept. of Medicine

My research





The BIG picture



Characterise signal

Feature generation → Feature selection/transformation → Statistical mapping

Supervised learning setting

	X				y
Samples	**feature 1**	**feature 2**	**...**	**feature *M***	**Outcome**
S_1	3.1	1.3		0.9	type 1
S_2	3.7	1.0		1.3	type 2
S_3	2.9	2.6		0.6	type 1
…					…
S_N	1.7	2.0		0.7	type 5

N (samples)

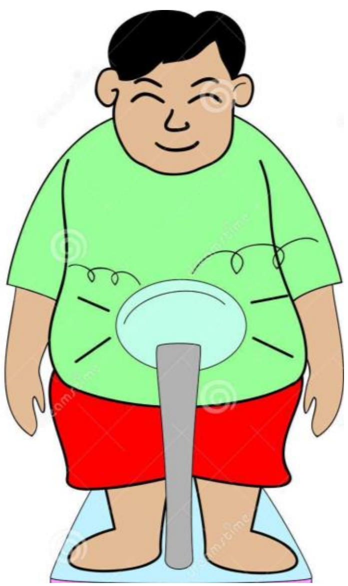
M (features or characteristics)

$$\mathbf{y} = f\,(\mathbf{X})$$

f **: mapping**	**X: Design matrix**	**y: outcome**

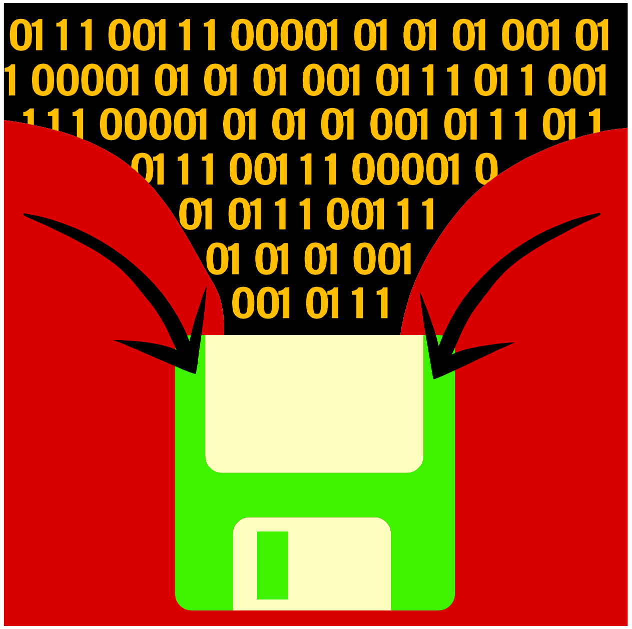
Thin and fat datasets

Samples	f 1	f2	...	feature *M*
S_1	3.1	1.3		0.9
S_2	3.7	1.0		1.3
S_3	2.9	2.6		0.6
…				
S_N	1.7	2.0		0.7



Samples	feature 1	feature 2	...	feature *M*
S_1	3.1	1.3		0.9
S_2	3.7	1.0		1.3
S_3	2.9	2.6		0.6
…				
S_N	1.7	2.0		0.7

Curse of dimensionality



- Many features *M* ☞ **Curse of dimensionality**

Solution to the problem

- Reduce the initial feature space *M*

  into *m* (or *m*<<*M*)

- **Feature selection**
- **Feature transformation**

Feature transformation

- Manifold embedding (e.g. PCA)

- **Not easily interpretable**

Feature selection

- Minimal feature subset with maximal predictive power
- **Interpretable**

Reminder: supervised learning

X **y**

N (samples)

Samples	feature 1	feature 2	...	feature *M*	Outcome
S_1	3.1	1.3		0.9	type 1
S_2	3.7	1.0		1.3	type 2
S_3	2.9	2.6		0.6	type 1
…					…
S_N	1.7	2.0		0.7	type 5

M (features or characteristics)

$$\mathbf{y} = f\,(\mathbf{X})$$

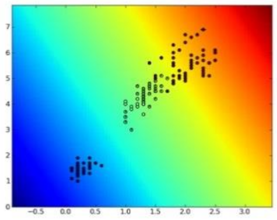
f **: mapping**	**X: Design matrix**	**y: outcome**

Functional mapping

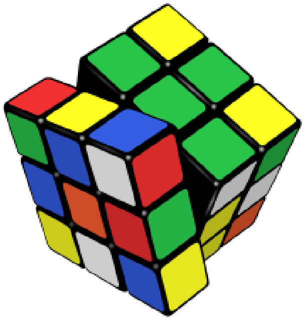
$$\mathbf{y} = f\,(\mathbf{X})$$

- Simple approaches (LDA, kNN…)

- Support Vector Machines (SVM)



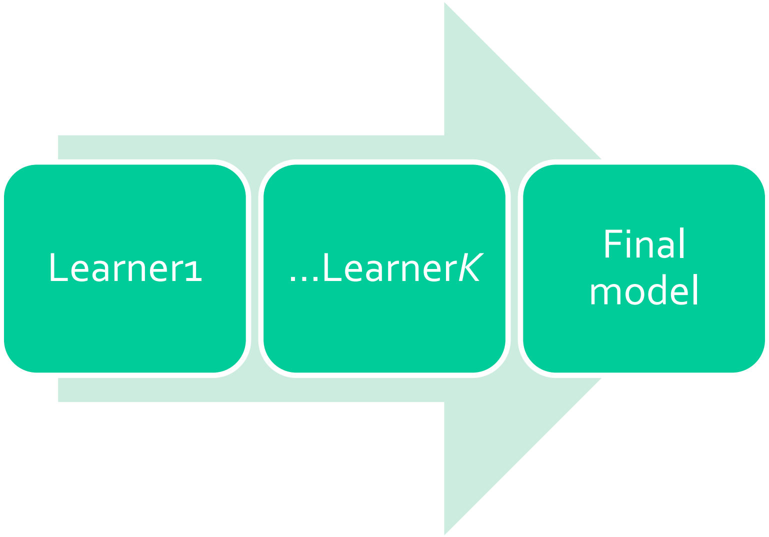
- **Ensembles**

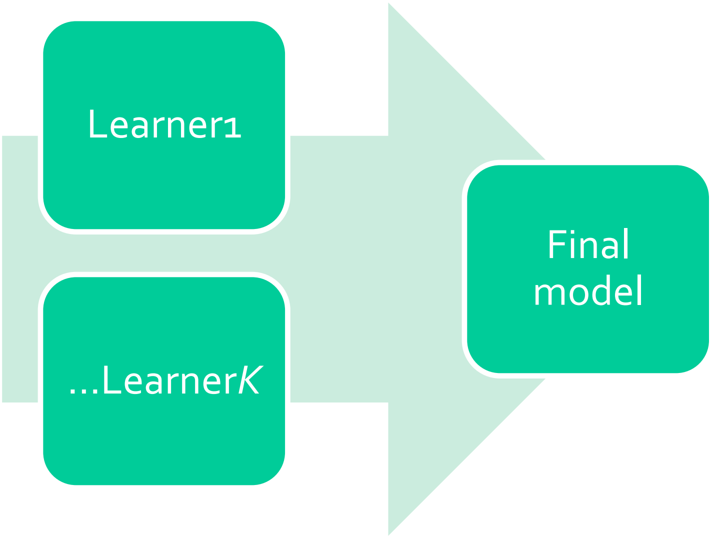


Ensembles

- Ensemble = combination of learners

- **Sequential** VS **Parallel**





Ensembles founding question

“Can a set of **weak learners** create a single **strong learner**?”

A brief history of ensembles

Bootstrapping

Resampling data

Bagging

Boosting

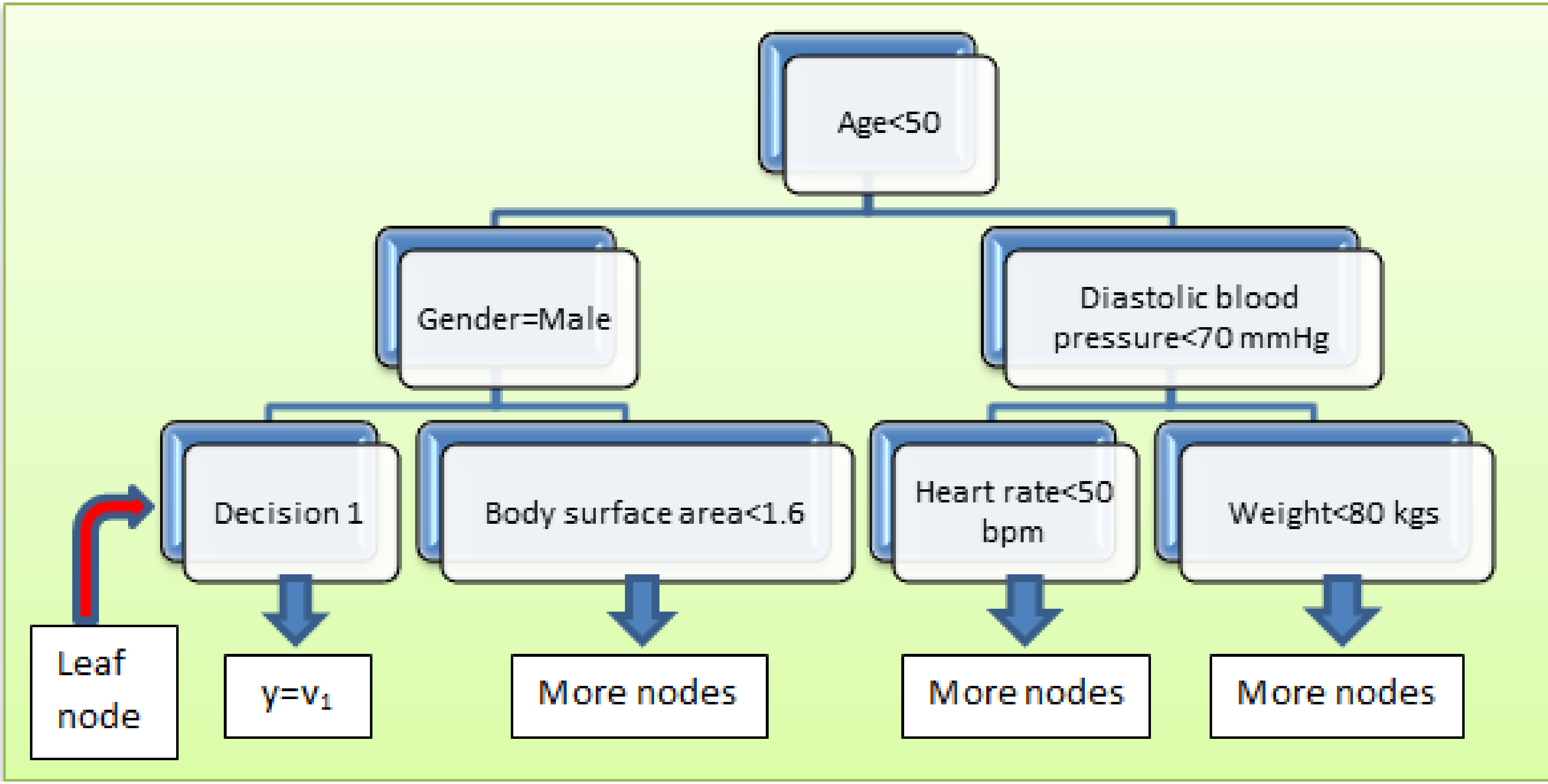
Adaptive boosting (Adaboost)
(Schapire and Freund 1997)

Classifier design

Decision trees

- Powerful, conceptually simple learners

Decision trees: an example



Tree growing process I

- Find best split & partitioning data into two sub-regions (*nodes*)

- Exhaustive search

- Determine the pairs of half-planes $\{R_1(j,s), R_2(j,s)\}$:

$$\begin{cases} R_1(j,s) = \{\mathbf{X} \,|\, \mathbf{f}_j \le s\} \\ R_2(j,s) = \{\mathbf{X} \,|\, \mathbf{f}_j > s\} \end{cases}$$

- Stop when data in the node is below some threshold (min. node size)

Tree growing process II

- Optimal feature $\mathbf{f}_j$ and splitting point s according to loss function:

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\,\epsilon\,R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\,\epsilon\,R_2(j,s)}(y_i-c_2)^2\right]$$

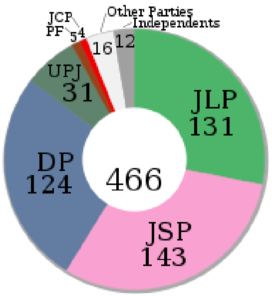
  where c_1, c_2 are the mean values of the y_i in the node when using the sum of squares as the loss function

$$\begin{cases} c_1 = mean(y_i|\mathbf{x}_i\,\epsilon\,R_1(j,s)) \\ c_2 = mean(y_i|\mathbf{x}_i\,\epsilon\,R_2(j,s)) \end{cases}$$

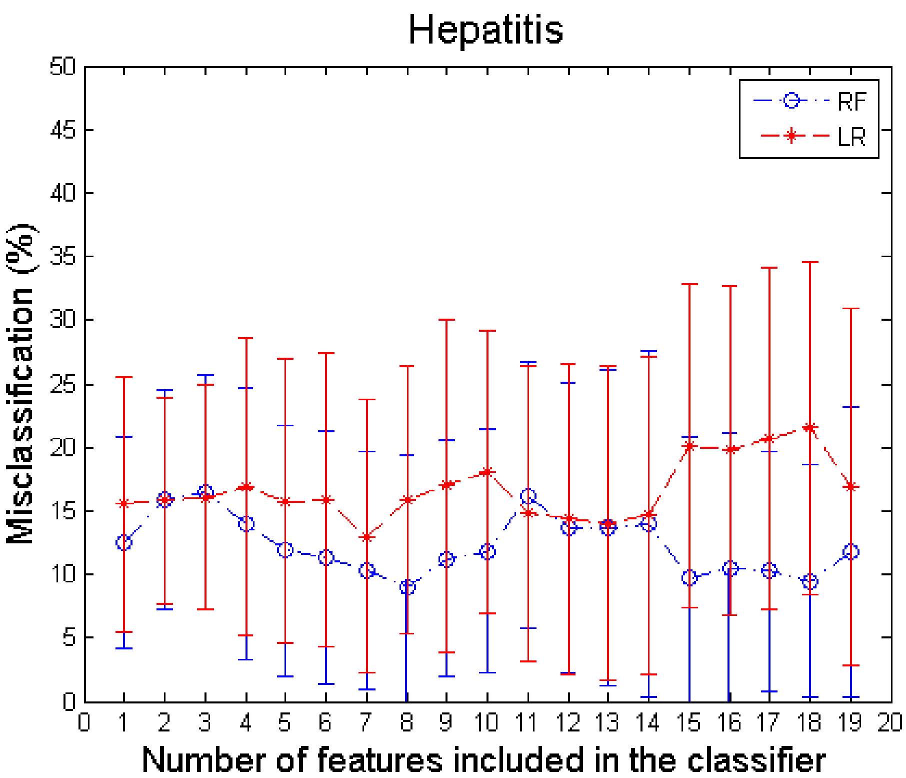
- Equations differ depending on **the loss function** to be optimized

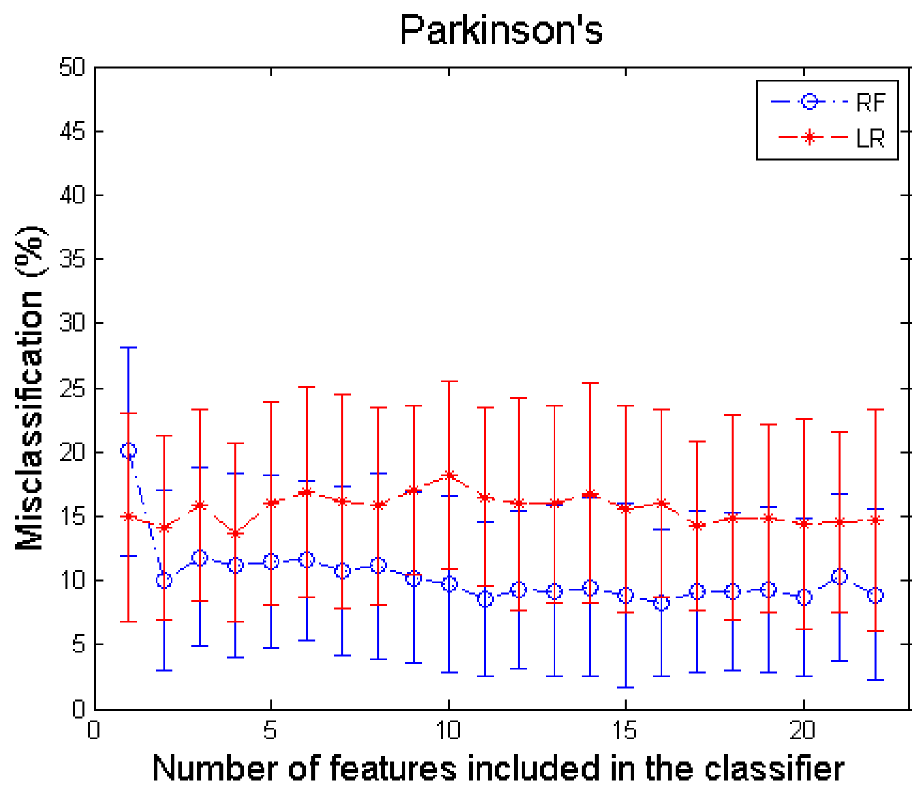
Random Forests (RF)

- Parallel combination of decision trees
- Use many decision trees (typically 500)
- **Bagging**
- Each tree, each node accesses ($\sqrt{M}$) features
- Majority voting



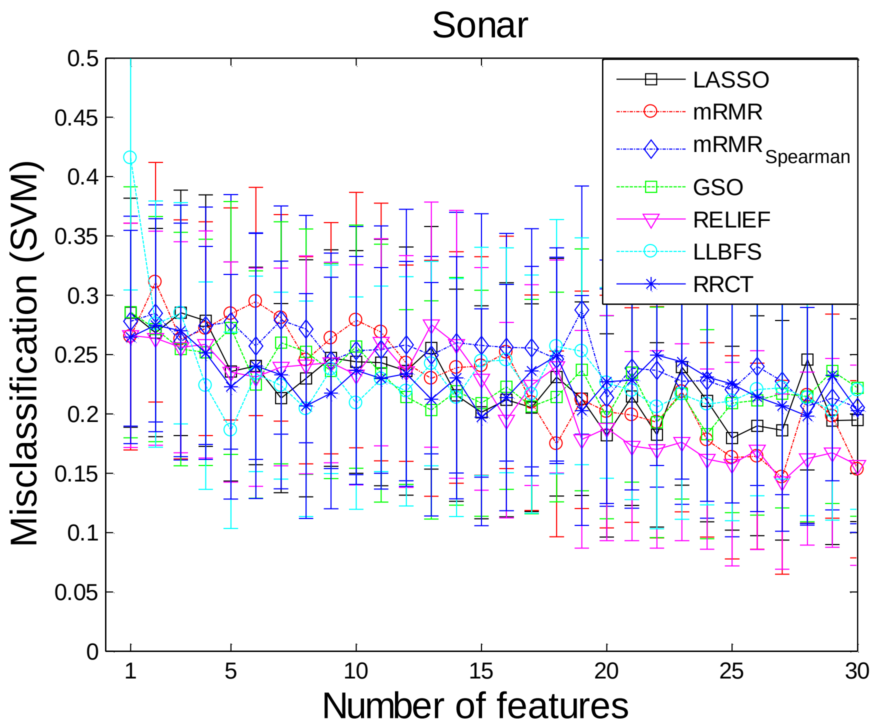
Some results

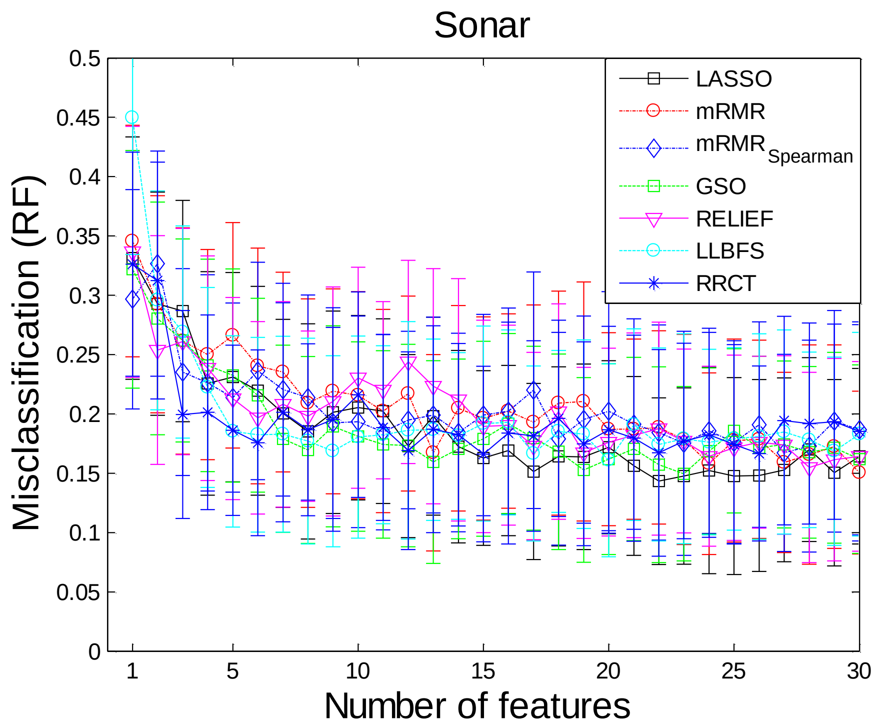




Comparing Logistic Regression (LR) with Random Forests (RF)

Some results





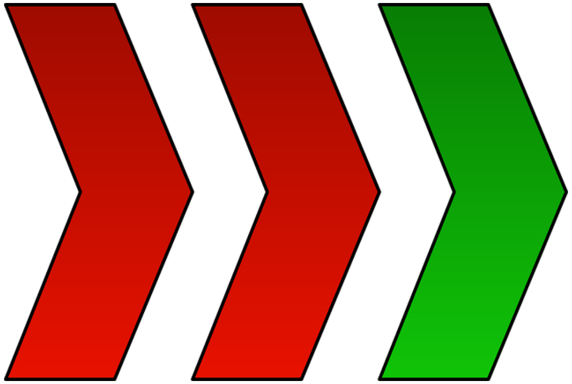
- Comparing Support Vector Machines with Random Forests on one dataset

Final remarks on Random Forests

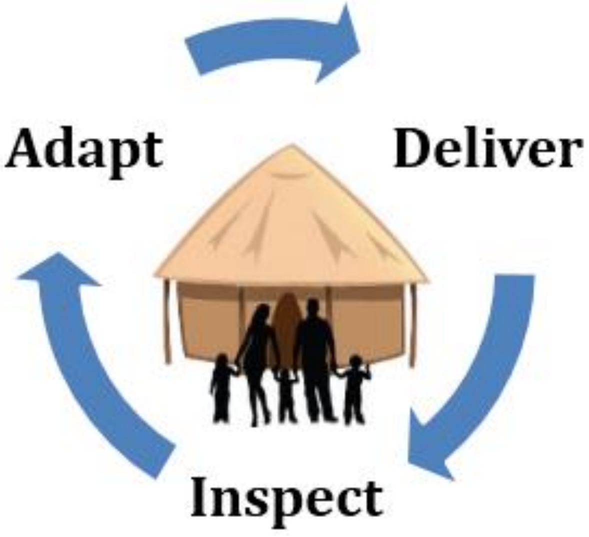
- State of the art learner
- Very robust to hyper-parameter selection (just use it off-the-shelf!)
- Bonus: provides feature importance scores
- Weakness: cannot capture well linear relationships (works in steps internally)
- Powerful in multi-class classification settings

Boosting and adaptive boosting

- **Boosting**: reduce bias by combining sequential learners



- **Adaptive boosting**: adjust the weights of the samples



Sequential ensemble: Adaboost

- Train sequential learners
- Introduce weights on misclassified samples
- Sensitive to noise and outliers in the data
- No overfitting claims; late results ➔ can overfit data
- Can be used with different types of base learners
- Convergence if all learners are better than chance

Adaboost

- Given $\{\mathbf{x}_i, \mathrm{y}_i\}_{i=1\ldots N}$, $\mathbf{x}_i \in \Re^M$ and $\mathrm{y}_i \in \{+1, -1\}$
- Initialize the weights $\mathbf{D}_1$, for each of the samples: $\mathbf{D_1}(i) = \frac{1}{N}$
- For $t = 1 \ldots T$ sequential weak learners
- Find the learner $f(\mathbf{D}_{\boldsymbol{t}})$: $\min\left(\{L(\mathrm{y}_i, \hat{y}_i)\}_{i \in \mathbf{D}_{\boldsymbol{t}}}\right)$
- Ensure that the weak learner is better than chance (0.5)
- Update weights $\mathbf{D}_{\boldsymbol{t}+\mathbf{1}}(i) = f(\mathbf{D}_{\boldsymbol{t}}(i), \hat{y}_i \neq \mathrm{y}_i)$
- Free parameter: influence of misclassified samples

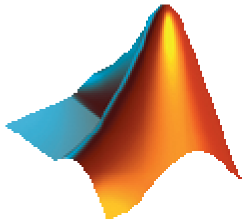
Summary of Adaboost

- Easy to implement

- Generic framework – can work with any type of learner

- Competitive with Random Forests – no clear winner

- Variants for robustness (see Matlab’s native function “fitensemble”)



Adaboost versus RF

- Instead of bootstrapping (RF), uses sample reweighting
- Instead of majority voting (RF), uses weighted voting
- Explicitly revisits samples misclassified by previous weak learners

Choosing algorithms

- What kind of data do you have? Labeled data? Data growing daily?
- Often the biggest practical challenge is creating or obtaining enough training data
- For many problems and algorithms, hundreds or thousands of examples from each class are required to produce a high performance classifier
- Number of samples, features, correlations, interactions… use plots.
- **No free lunch theorem**: no universally best learner!

Thanasis’ rules of thumb

- Get to **know your data before any processing!**
  - (1) plot densities and scatter plots – univariate analysis and intuitive “feel”, perhaps these suggest simple data transformation
  - (2) compute correlations and correlation matrix, identify interactions (e.g. via partial correlations and conditional mutual information)
  - (3) Feature selection – a simple guide: if there are low correlations use LASSO, if there are few (low) interactions use mRMR
  - (4) In my experience, SVMs may work better for binary-class classification problems, and RF may work better for multi-class classification problems

I would strongly advise testing both SVM and RF (and possibly other statistical mapping algorithms) in the problems you study

Appendix: food for thought

- **A Unifying Review of Linear Gaussian Models**

  http://www.cs.nyu.edu/~roweis/papers/NC110201.pdf

  an excellent paper by Sam Roweis and Zoubin Ghahramani unifying linear models

- **Statistical modelling: the two cultures**

  http://faculty.smu.edu/tfomby/eco5385/lecture/Breiman%27s%20Two%20Cultures%20paper.pdf

  Great introduction: first principles versus statistical modelling by Leo Breiman

- **Relations Between Machine Learning Problems**

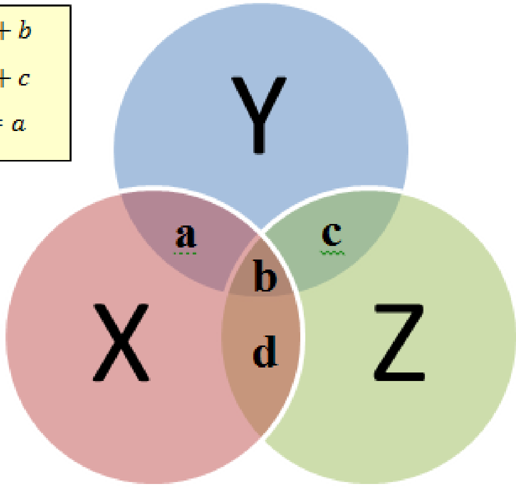
  http://videolectures.net/nipsworkshops2011_williamson_machine/

  Check out these lectures!

Additional Slides

Feature selection concepts

$$r(X,Y) = a + b$$
$$r(Z,Y) = b + c$$
$$r_p(X,Y|Z) = a$$



Relevance

Redundancy

Complementarity

LASSO (L1 regularization)

- Start with classical ordinary least squares regression
- L1 penalty: sparsity promoting, some coefficients become 0

$$\hat{\mathbf{b}}_{LASSO} = \arg\min_{b} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{M} x_{ij} b_j \right)^2 + \lambda \sum_{j=1}^{M} |b_j|$$

where λ is the regularization parameter (increasing λ causes more coefficients to become 0)

- Possible to introduce additional penalties, e.g. L2-norm
- L2 penalty: shrinkage in the regression coefficients

RELIEF

- Concept: work with nearest neighbours
- **Nearest hit** (NH) and **nearest miss** (NM)
- Great for datasets with interactions but does not account for information redundancy

$$W(\mathbf{f}_j) \stackrel{\text{def}}{=} \frac{1}{q}\sum_{i=1}^{q}\left\{\begin{array}{c} \underbrace{-\frac{1}{|\mathrm{NH}(\mathbf{x}_i)|}\cdot\sum_{\mathbf{x}_n \in \mathrm{NH}(\mathbf{x}_i)}\left\|x_{i,j}-x_{n,j}\right\| +}_{\mathit{Nearest\ hit\ term\ distance}} \\ \sum_{y_l \neq y_i}\frac{1}{|\mathrm{NM}(\mathbf{x}_i)|}\cdot\frac{\mathrm{P}(y=y_l)}{1-\mathrm{P}(y=y_i)}\cdot\sum_{\mathbf{x}_n \in \mathrm{NM}(\mathbf{x}_i)}\left\|x_{i,j}-x_{n,j}\right\| \end{array}\right\}$$

mRMR

- minimum Redundancy Maximum Relevance (mRMR)
- **Generally works very well**

$$\text{mRMR} = \max_{i \in Q-S}\left[\underbrace{I(\mathbf{f}_i;\mathbf{y})}_{relevance} - \underbrace{\frac{1}{\|S\|}\sum_{s \in S} I(\mathbf{f}_i;\mathbf{f}_s)}_{redundancy}\right]$$

Where $\|S\|$ refers to the *cardinality* of the selected subset (number of selected features until that step)

My new algorithm RRCT

The best result will come when everyone in the group doing what is best for himself… and the group.

https://www.youtube.com/watch?v=LJS7Igvk6ZM

$$\mathrm{RRCT} \stackrel{\mathrm{def}}{=} \max_{i \in Q-S} \left[\underbrace{r_{\mathrm{IT}}(\mathbf{f}_i;\mathbf{y})}_{relevance} - \underbrace{\frac{1}{|S|}\sum_{s \in S} r_{\mathrm{IT}}(\mathbf{f}_i;\mathbf{f}_s)}_{redundancy} + \underbrace{\left[sign\left(r_{\mathrm{p}}(\mathbf{f}_i;\mathbf{y}|S)\right)\cdot sign\left(r_{\mathrm{p}}(\mathbf{f}_i;\mathbf{y}|S) - r(\mathbf{f}_i;\mathbf{y})\right)\right]\cdot r_{\mathrm{p,IT}}}_{complementarity} \right]$$

My new algorithm RRCT

Relevance & Redundancy trade-off

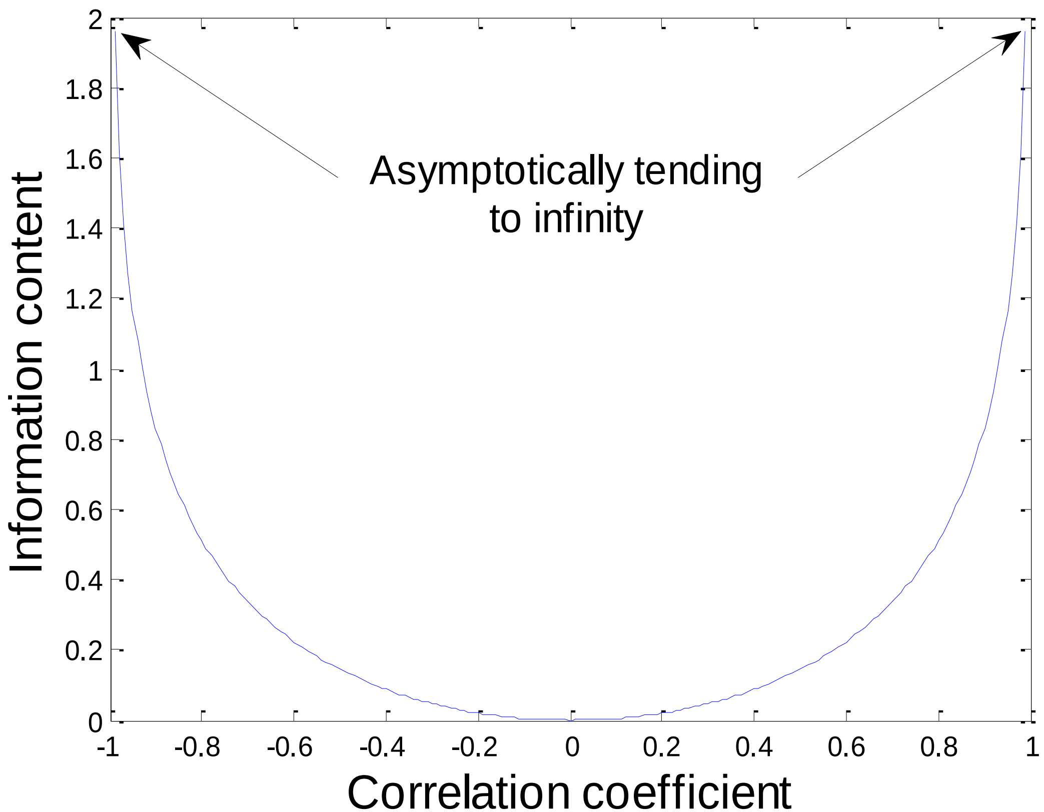
$$\mathbf{D} = -0.5 \cdot \log \begin{bmatrix} 1 - r_1^2 & 1 - \rho_{12}^2 & \dots & 1 - \rho_{1M}^2 \\ 1 - \rho_{12}^2 & 1 - r_2^2 & \cdots & 1 - \rho_{2M}^2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 - \rho_{1M}^2 & 1 - \rho_{2M}^2 & \dots & 1 - r_M^2 \end{bmatrix}$$

Complementarity

$$r_{\mathrm{p}}(X, Y|Z) = \frac{r(X,Y) - r(X,Z) \cdot r(Y,Z)}{\sqrt{r^2(X,Z)} \cdot \sqrt{r^2(Y,Z)}}$$

My new algorithm RRCT

- Normalizing pdfs of variables
- Information theoretic transformation $\boxed{r_{IT} = -0.5 \cdot \log(1 - r^2)}$



Validation setting

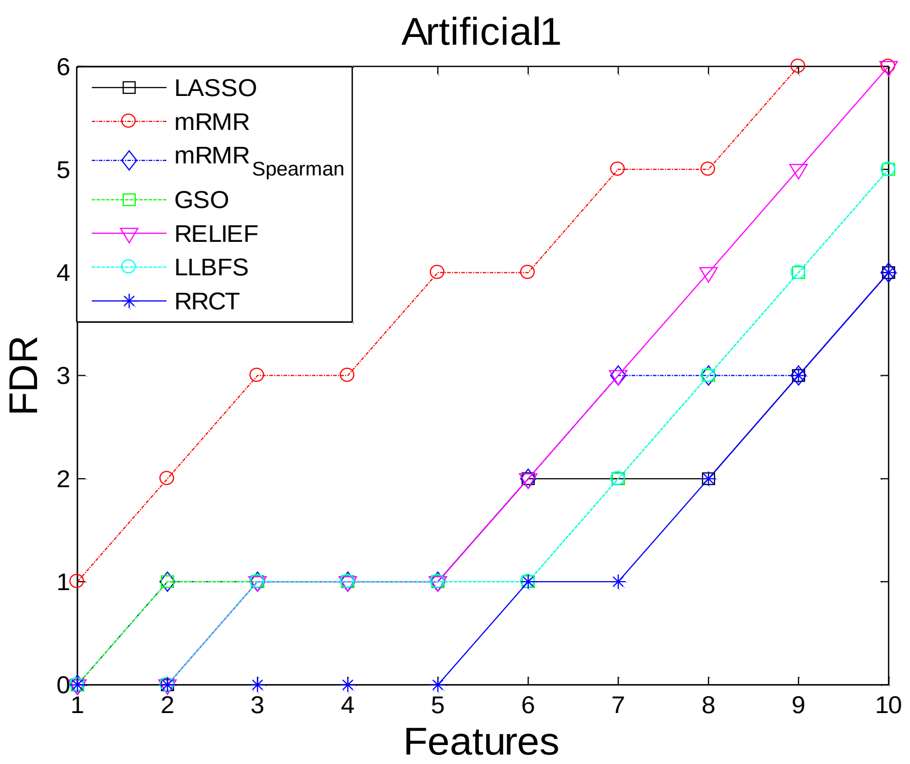
- Matching 'true' feature subset
  - Possible only for artificial datasets
- Maximize the **out of sample prediction performance**
  - adds an additional 'layer': the learner
  - potential problem: **feature exportability**
  - BUT… in practice this is really what is of most interest!

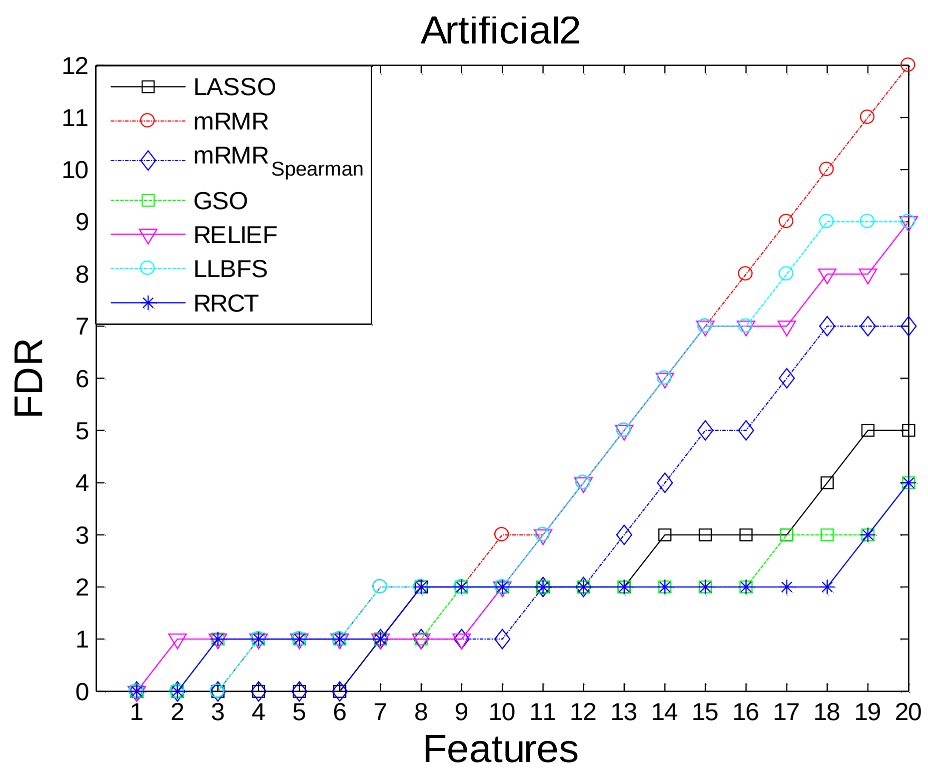
Dataset info

Dataset	Design matrix	Associated task	Type
MONK1[35]	124×6	Classification (2 classes)	D (6)
Artificial 1	500×150	Classification (2 classes)	C(150)
Artificial 2	1000×100	Classification (10 classes)	C(100)
Hepatitis	155×19	Classification (2 classes)	C (17), D (2)
Parkinson's[35]	195×22	Classification (2 classes)	C (22)
Sonar[35]	208×60	Classification (2 classes)	C (60)
Wine[35]	178×13	Classification (3 classes)	C (13)
Image segmentation[35]	2310×19	Classification (7 classes)	C (16), D (3)
Cardiotocography[35]	2129×21	Classification (10 classes)	C (14), D (7)
Ovarian cancer	72×592	Classification (2 classes)	C (592)
SRBCT	88×2308	Classification (4 classes)	C (2308)

False Discovery Rate (FDR)

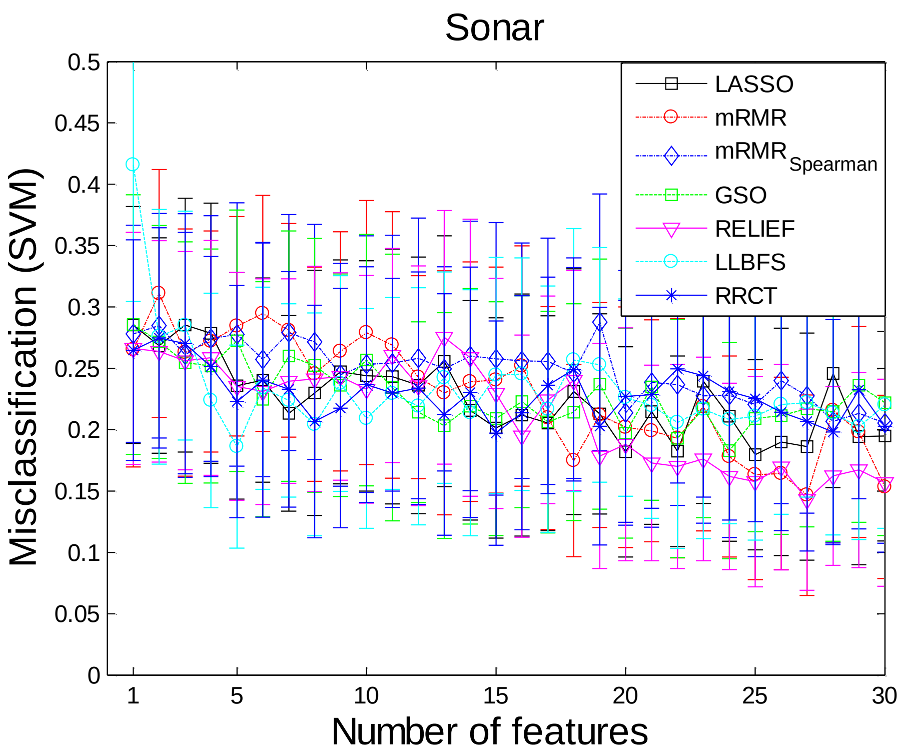
Artificial datasets with known ground truth

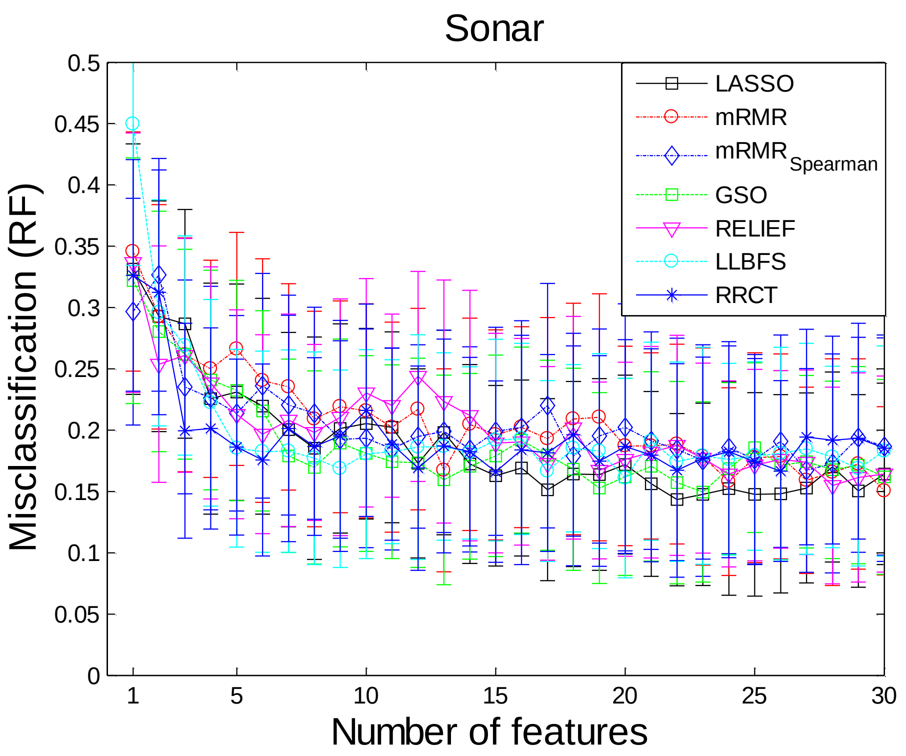




Indicative performance

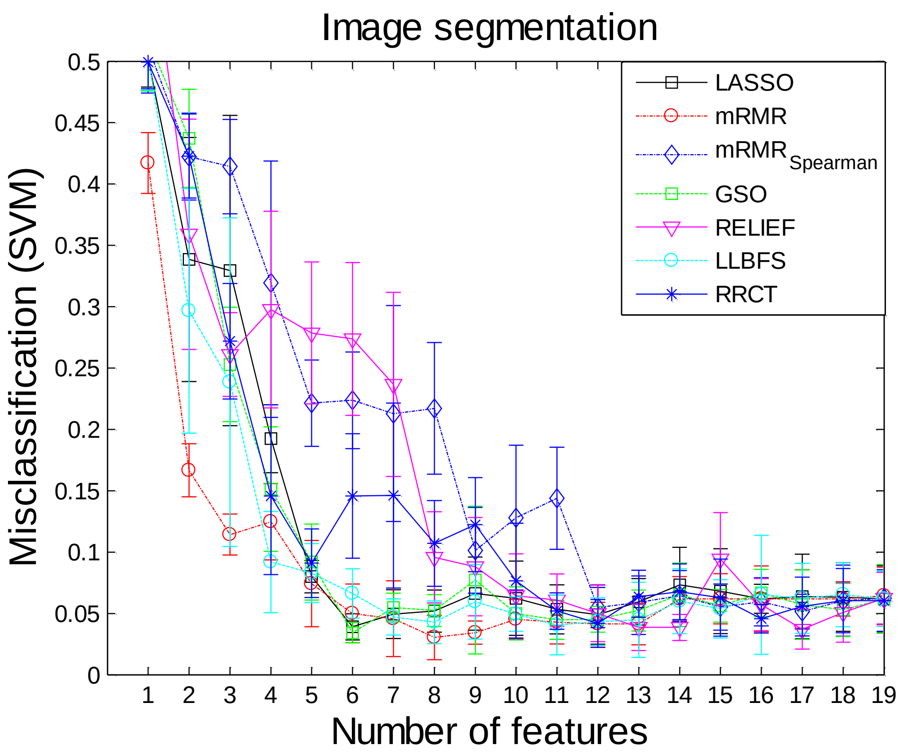
Classifier accuracy as proxy for FS accuracy

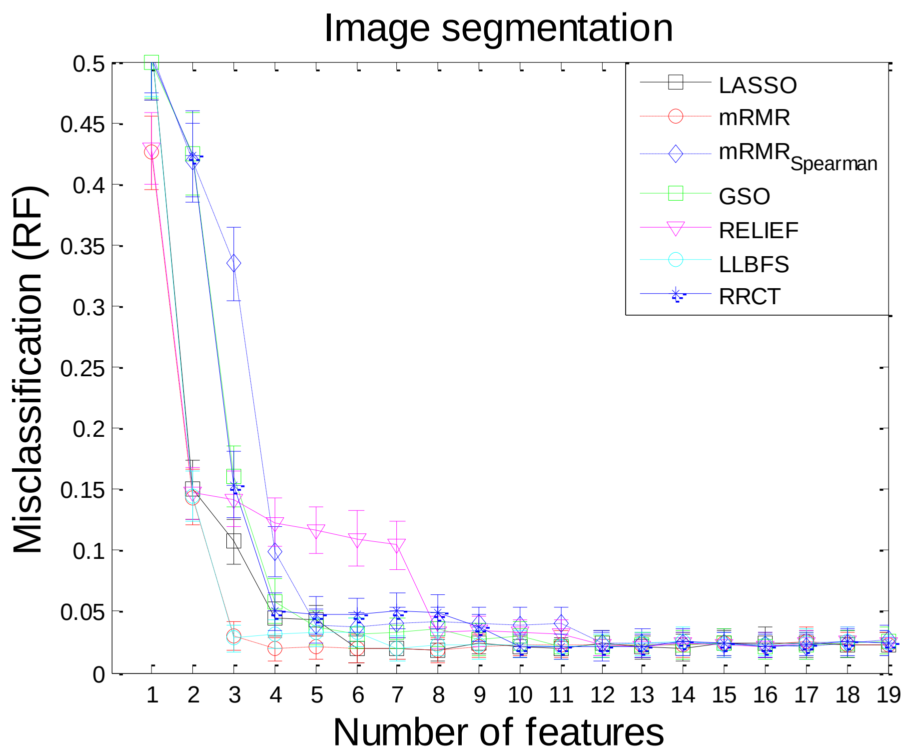




Indicative performance

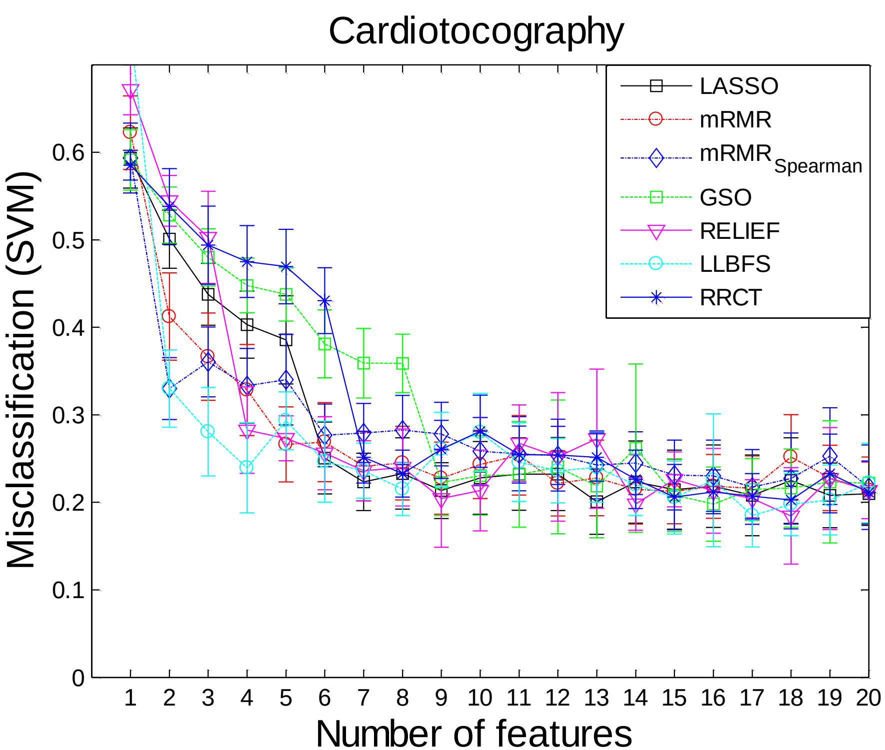
Classifier accuracy as proxy for FS accuracy

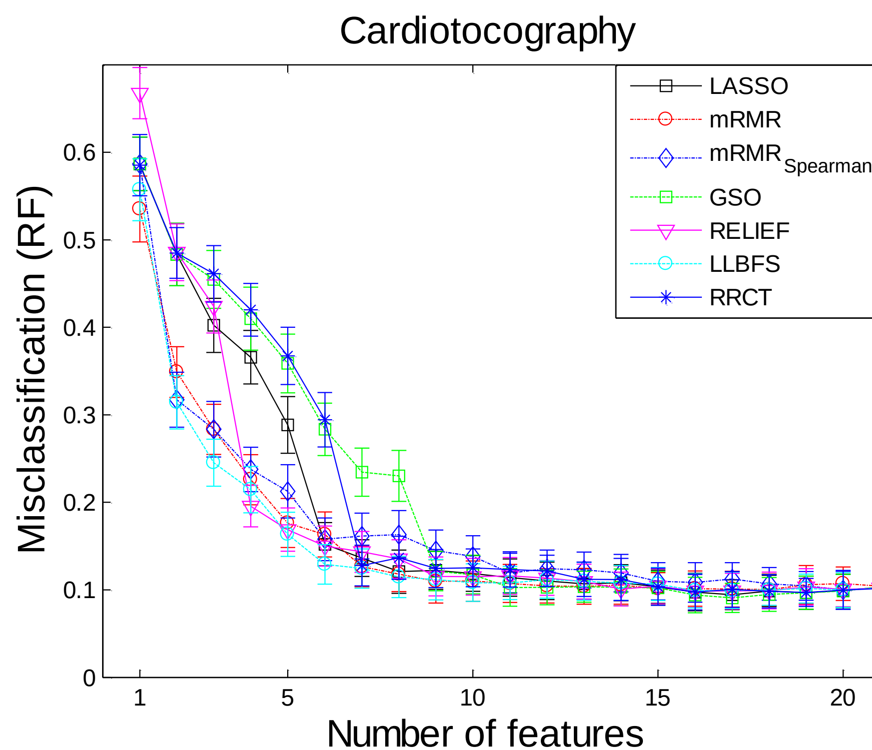




Indicative performance

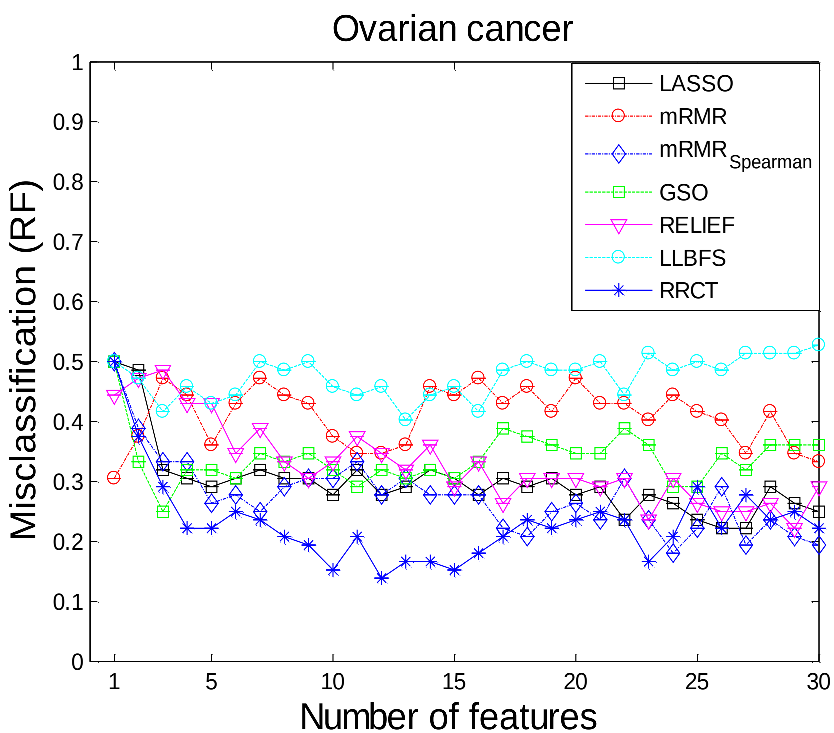
Classifier accuracy as proxy for FS accuracy

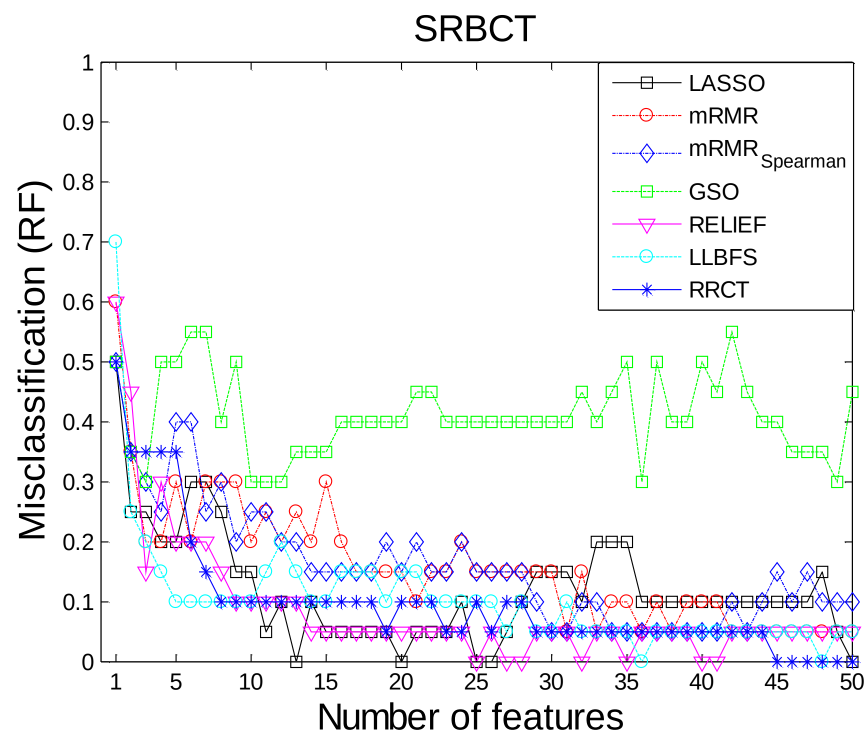




Performance in fat datasets

Very challenging setting for many algorithms





Supervised learning setting

Samples	f 1	f2	...	feature *M*
S_1	3.1	1.3		0.9
S_2	3.7	1.0		1.3
S_3	2.9	2.6		0.6
…				
S_N	1.7	2.0		0.7

Samples	feature 1	feature 2	...	feature *M*
S_1	3.1	1.3		0.9
S_2	3.7	1.0		1.3
S_3	2.9	2.6		0.6
…				
S_N	1.7	2.0		0.7

f : mapping	X: Design matrix	y: outcome