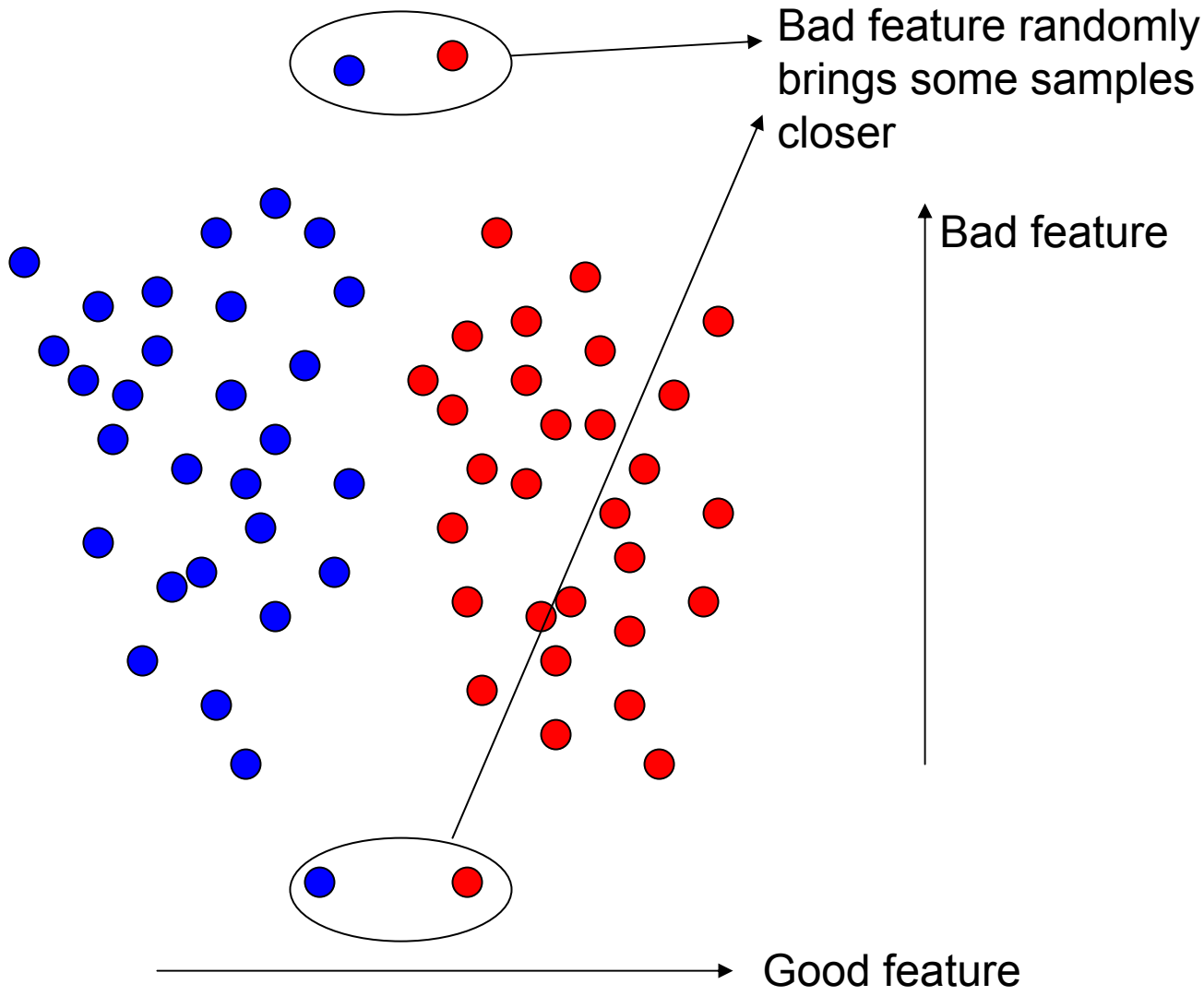
Feature selection

Bram van Ginneken

Feature selection

- Given a set of candidate features, select a subset that performs the best under some classification system

Why bad features hurt



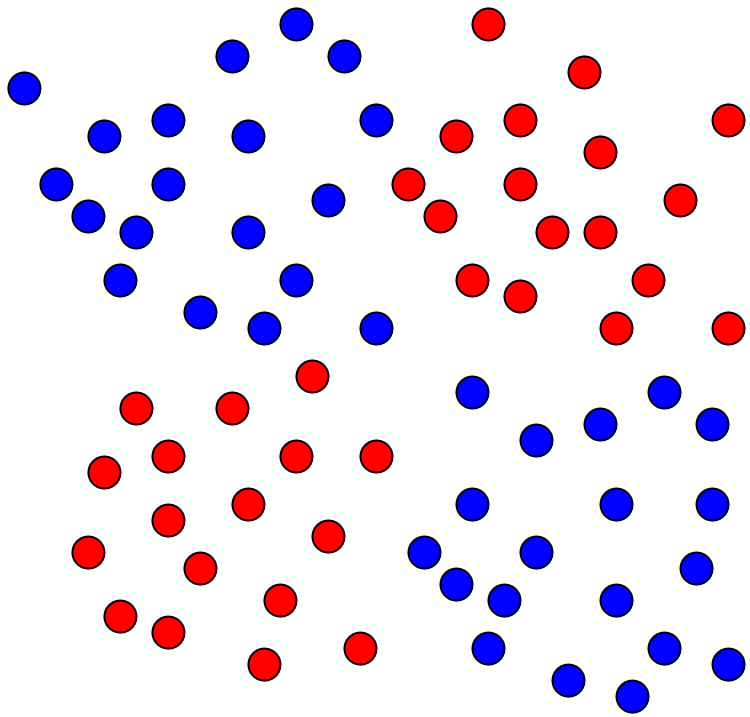
Feature selection vs extraction

- Linear feature extraction is a matrix multiplication $\boldsymbol{y} = \mathbf{A}\,\boldsymbol{x}$
- Feature selection amounts to a matrix with 0s and 1s, and just a single 1 per row → huge restriction
- Still the problem is very hard to optimize: with n features there are 2^n-1 possibilities!

Wrapper versus filter based

- Filter based methods:
  - score features individually
  - highest ranking features are selected
  - typically no combinations are tested
  - independent of classifier
  - cheap
- Wrapper-based methods:
  - test features with the classifier of choice
  - expensive
  - potentially better

Nested features



Together, these two features are valuable. Standalone, they are poor.

Clearly a nonlinear transformation could result in a single strong feature.

Some methods create such features, and use feature selection

Redundant features

- If two features are both good, but highly correlated, adding them both usually does not improve performance a lot
- If a second feature is less good, but not worthless, and independent of the first, adding it may make more sense
- Impossible to evaluate these thing feature by feature, even if pairs are examined, hidden relations can be missed

Filter-based

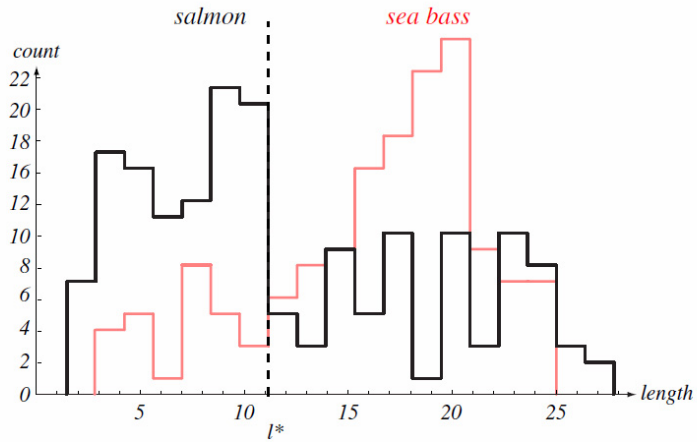


FIGURE 1.2. Histograms for the length feature for the two categories. No single threshold value of the length will serve to unambiguously discriminate between the two categories; using length alone, we will have some errors. The value marked l^* will lead to the smallest number of errors, on average. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

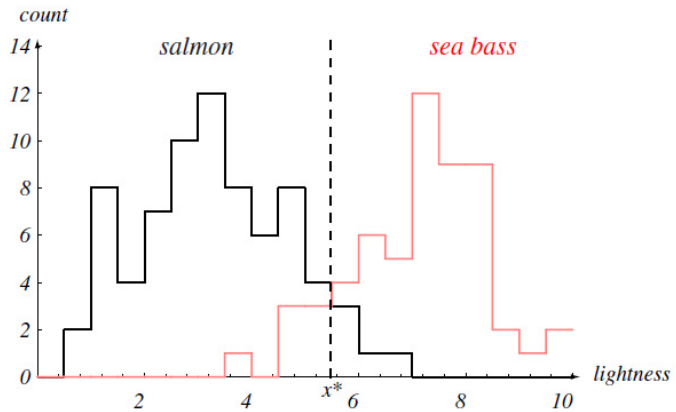
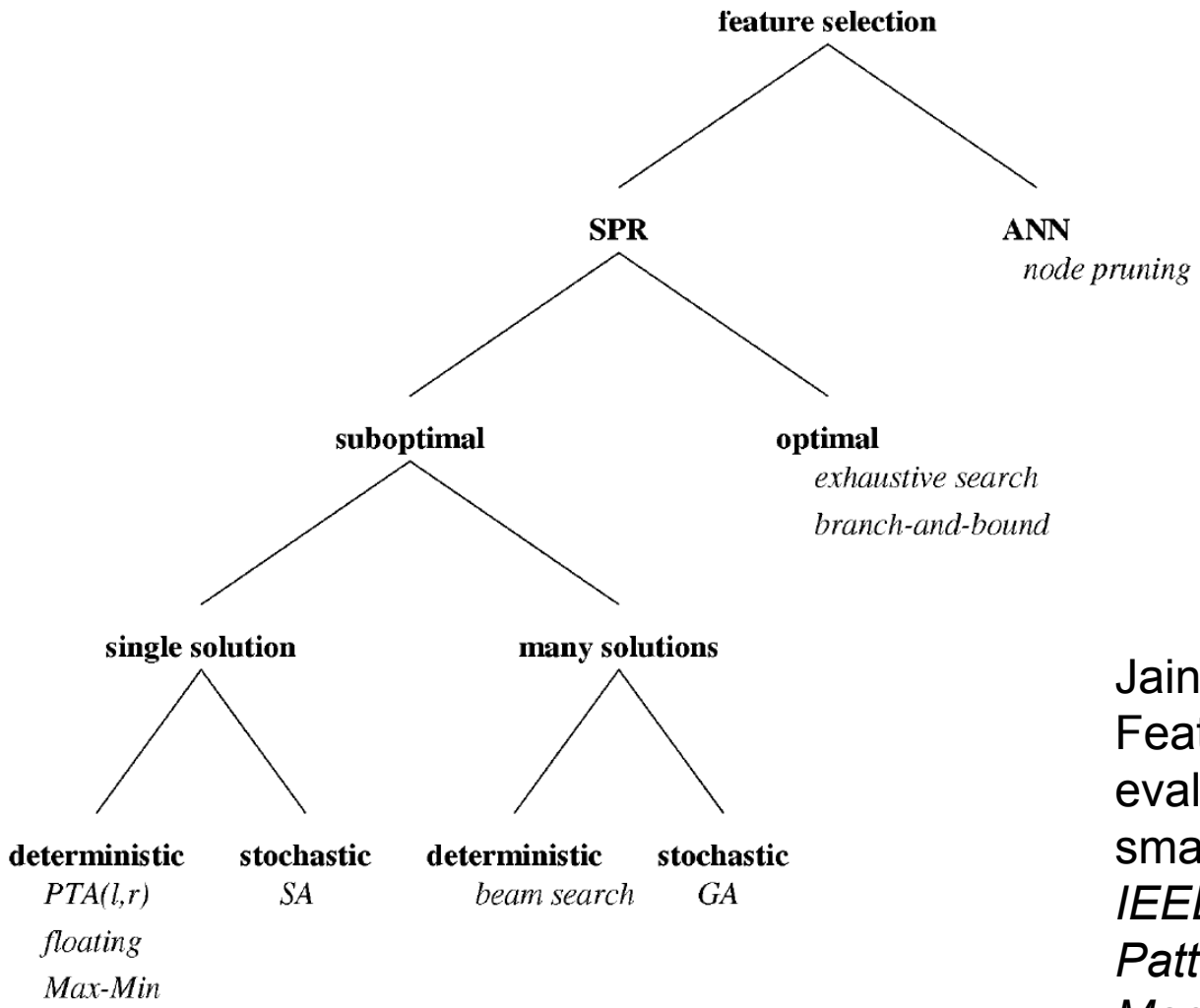


FIGURE 1.3. Histograms for the lightness feature for the two categories. No single threshold value x^* (decision boundary) will serve to unambiguously discriminate between the two categories; using lightness alone, we will have some errors. The value x^* marked will lead to the smallest number of errors, on average. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

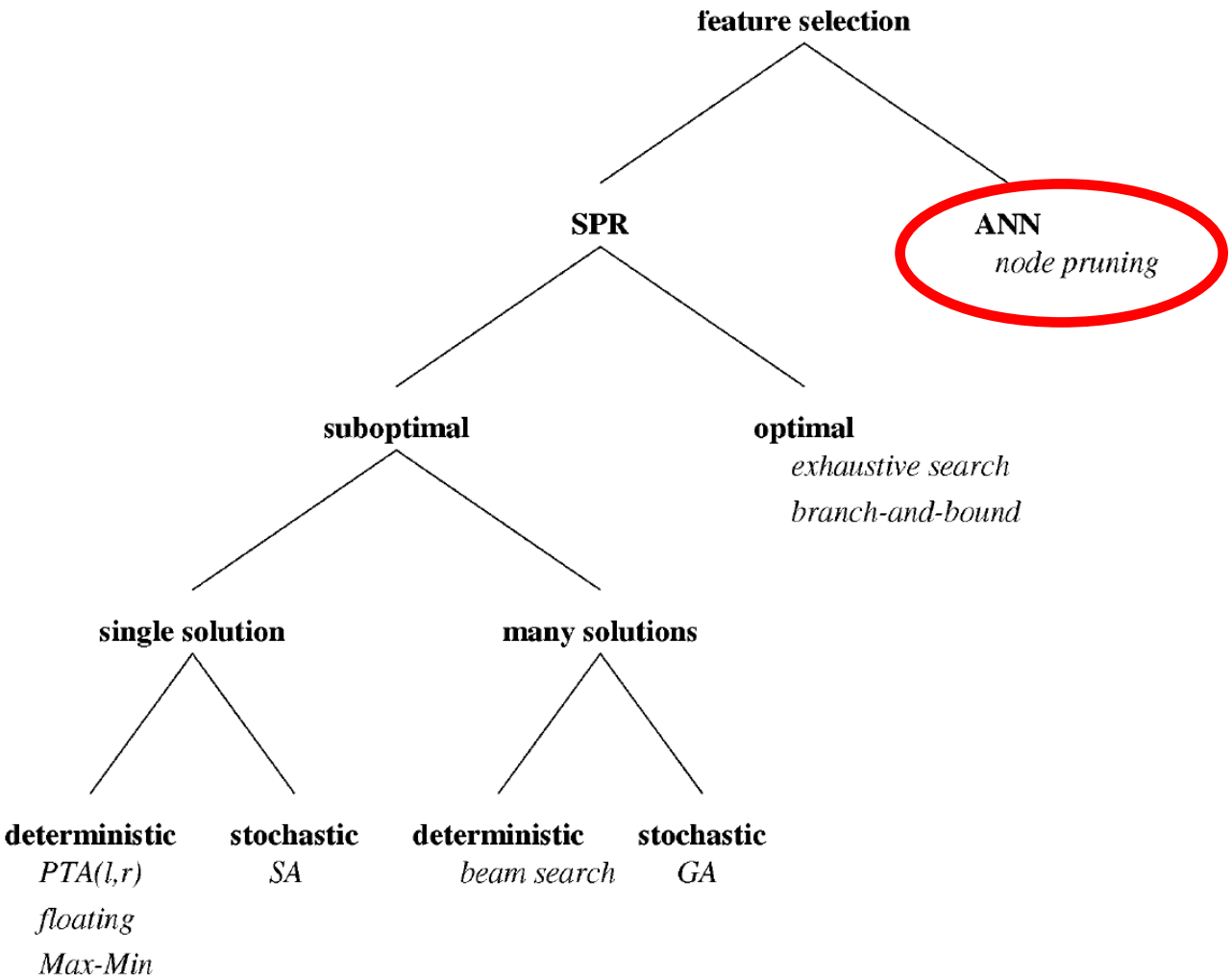
- Measure how well both distributions are separated. E.g. by $(\mu_1-\mu_2)/((\sigma_1+\sigma_2)/2)$

Wrapper methods

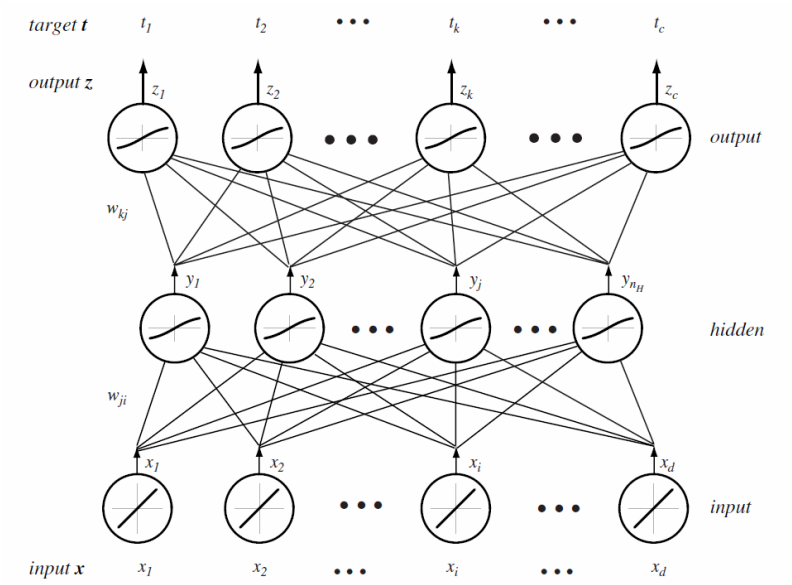


Jain, A. & Zongker, D. Feature selection: evaluation, application and small sample performance *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **1997**, *19*, 153-158

Wrapper methods

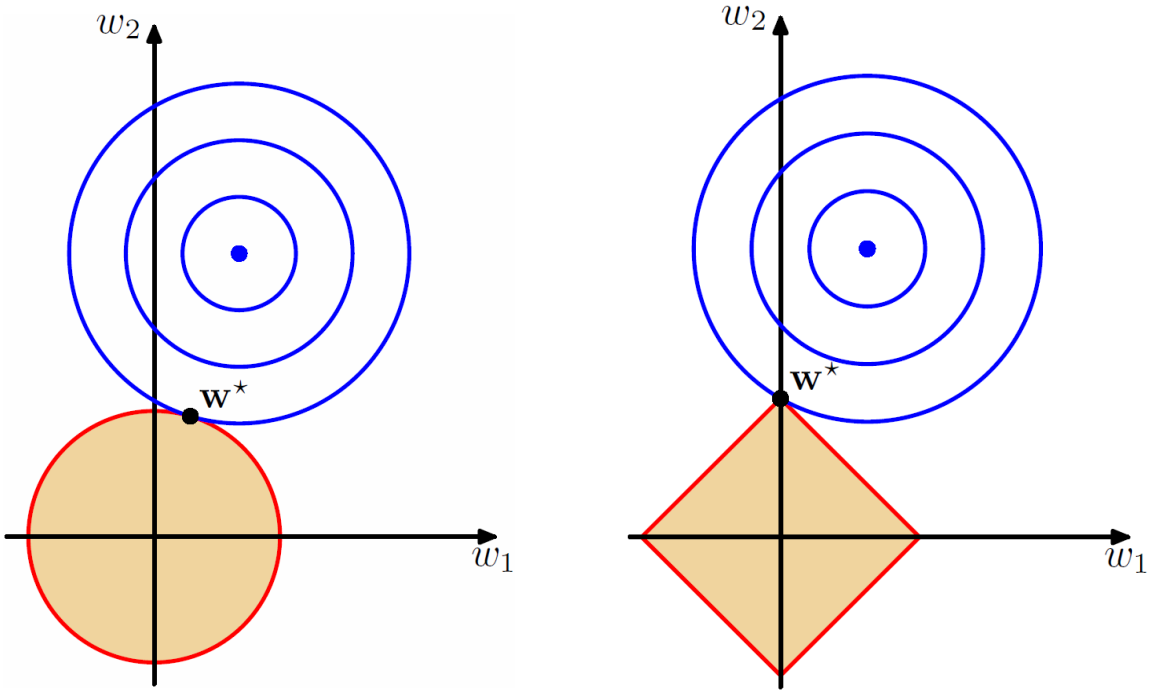


Built-in feature selection



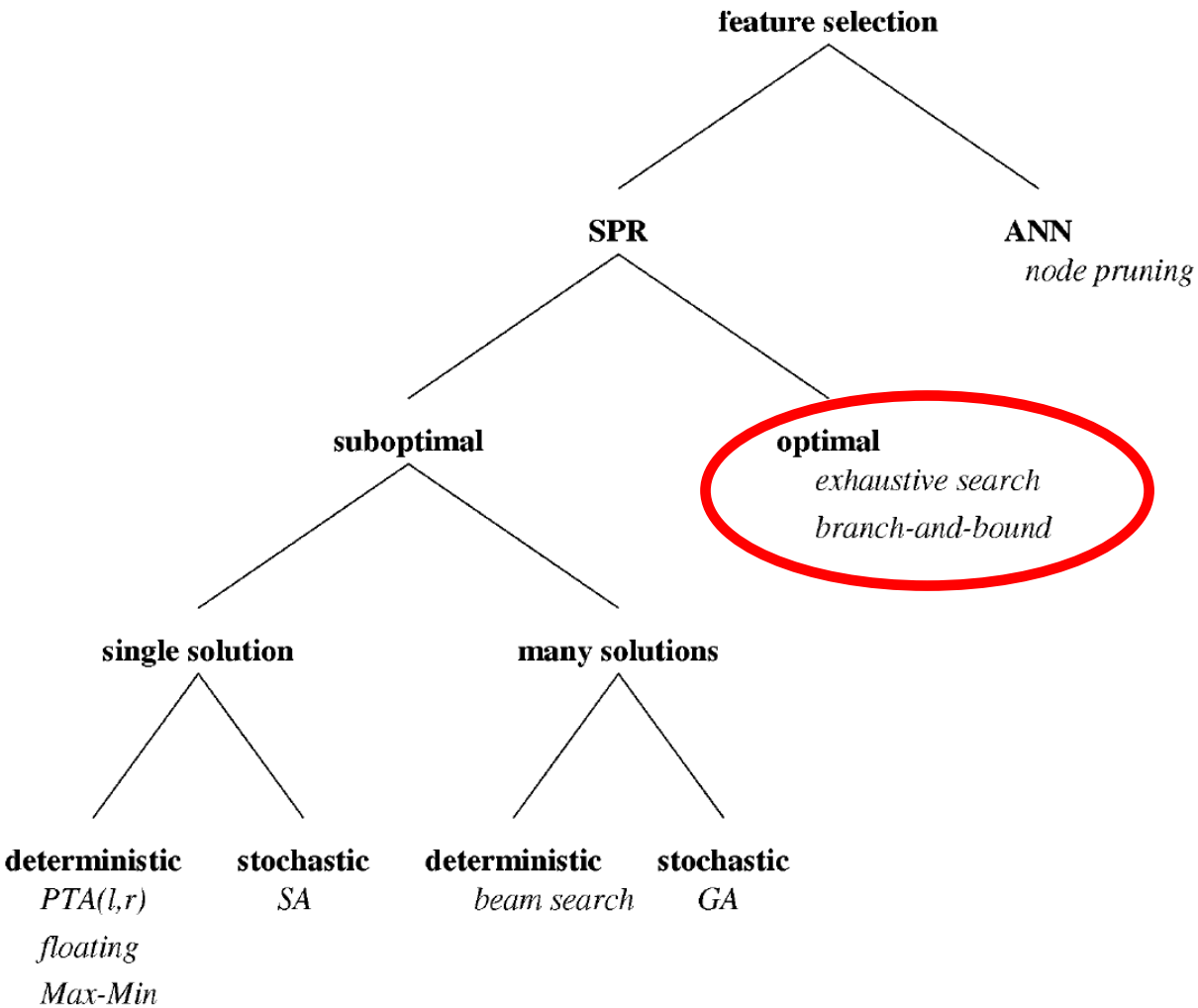
- Features that get low weights may be eliminated during training

Built-in feature selection



- Certain regularization terms force certain weights to zero → feature selection

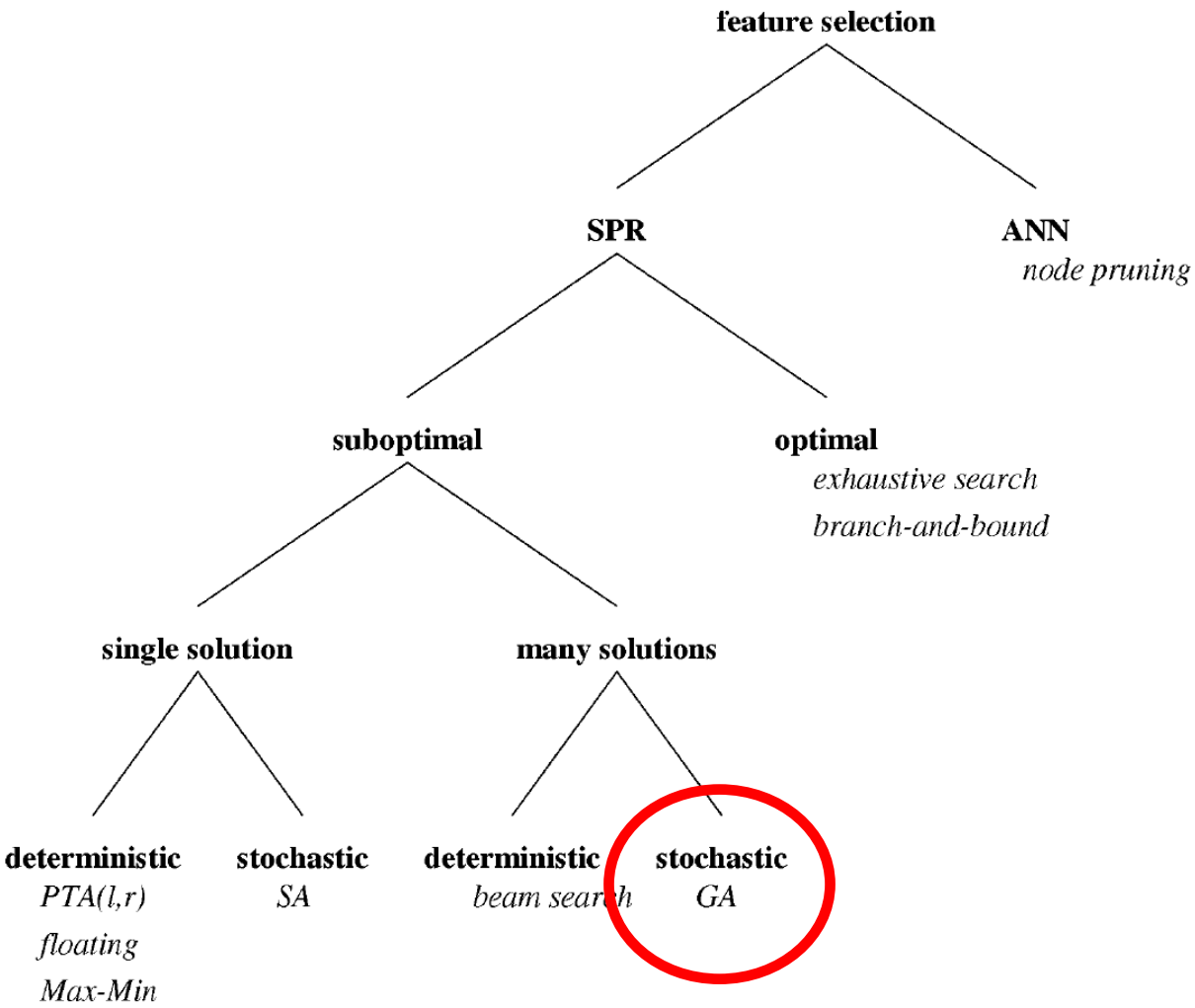
Wrapper methods



Optimal methods

- Exhaustive search is generally infeasible
- If error is guaranteed to decrease with more features, branch-and-bound can be used

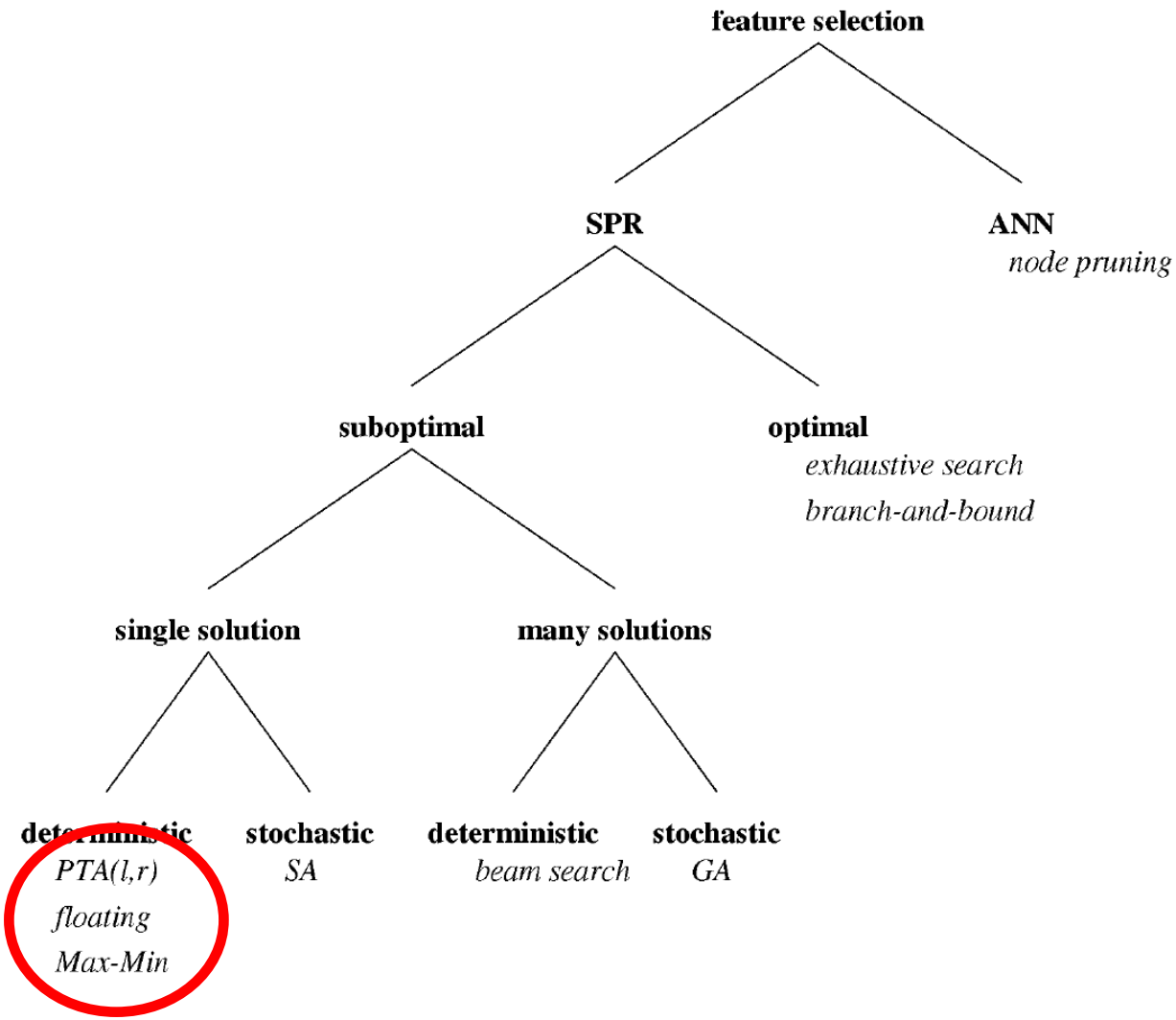
Wrapper methods



Genetic algorithms

- The feature set is represented as a binary string (chromosome): 0100101010011101
- A population of chromosomes is tested and subjected to
  - Mutations (random perturbations)
  - Crossover (combinations of 2 chromosomes)
- Only successful chromosomes survive
- Brute force method
- Many variations in the rules exist
- Can be extended to feature weighting easily

Wrapper methods



Sequential forward selection

1. Features, $f_1,\ldots f_N$. Start with zero feature set S and classifier C and criterion J initialized to 0 (accuracy, Az)
2. For $i = 1,\ldots,N$ add f_i to S (if not already in there), evaluate J_i
3. If best $J_i > J$, add feature f_i to S and goto 2 (unless a maximum number of features has been selected)

Sequential backward selection

- The same procedure, but starting with all features and eliminating the poorest features one by one

Variations

- Consider any combination of *n* features at each step
- Great increase in computation time
- Allows one to discover ‘nested’ features

Sequential floating forward/backward selection (SFFS/SFBS)

- After each forward/backward step, features are removed/added as long as performance improves
- Keep track of evaluated combinations
- Avoid early stopping is important (extra rules needed)
- The number of features in the optimal set ‘floats’
- SFFS has been found to be a very strong technique in several comparisons
- Some even more complex variations have been proposed but these are not used much

Boosted feature selection

Feature selection based on the training set manipulation

Pavel Křížek[a] Josef Kittler[b] Václav Hlaváč[a]
[a]Center for Machine Perception, Czech Technical University, Karlovo nám. 13,
121 35 Prague 2, Czech Republic, {krizekp1,hlavac}@fel.cvut.cz
[b]Centre for Vision, Speech and Signal Processing, University of Surrey,
GU2 7XH Guildford, United Kingdom, J.Kittler@surrey.ac.uk

Motivation

- SFFS is powerful but slow
- New technique examines features one by one, using only a threshold, and selects the best
- Similar to weak learners from boosting
- *By changing the weights of the samples*, ensure that previously selected features are irrelevant, and new good and non-redundant features are selected
- Claim: as good or better than SFFS, but much faster

Let us assign a weight $w_i^t > 0$ to every sample $\boldsymbol{x}_i$ form the training set $\mathcal{S}$ and let $\sum_{i=1}^{m} w_i^t = 1$ and $\boldsymbol{w}^t = (w_1^t, \ldots, w_m^t)$. Here $t \in \mathbb{N}$ indicates iteration step.

For $\theta \in \mathbb{R}$ and $x \in \mathbb{R}$ a simple threshold function

$$h_\theta(x) = \begin{cases} +1 & \text{if } x \geq \theta, \\ -1 & \text{otherwise} \end{cases} \tag{1}$$

is employed to assess discriminatory power of a feature. The parity of threshold (1) is not important, since the decision is symmetric, *i.e.*, $h_\theta(x) = -h_\theta(-x)$.

The quality of features with respect to weights $\boldsymbol{w}^t$ is evaluated by the error function

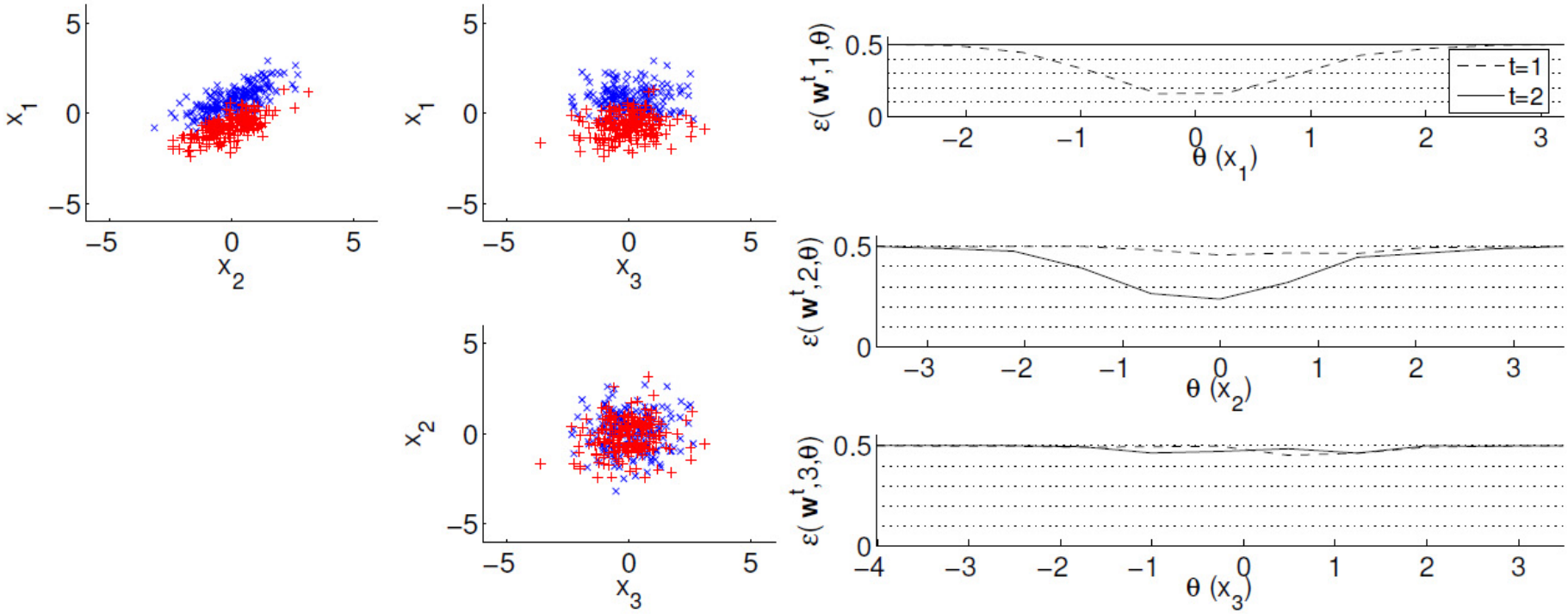
$$\varepsilon(\boldsymbol{w}^t, j, \theta) = \frac{1}{2} - \left| \frac{1}{2} - \sum_{i:\, h_\theta(x_{i,j}) \neq y_i} w_i^t \right| . \tag{2}$$

This is the probability of misclassification defined as a weighted sum of the samples misclassified by threshold θ.

The training set manipulation technique changes weights $\boldsymbol{w}^t$ in such a way that the currently selected feature j appears irrelevant with respect to new weights $\boldsymbol{w}^{t+1}$. Thus, error function (2) has to be $\frac{1}{2}$ for all thresholds θ_k. This is satisfied when the following conditions hold

$$\sum_{i:\ h_{\theta_k}(x_{i,j}) \neq y_i} w_i^{t+1} = \sum_{i:\ h_{\theta_k}(x_{i,j}) = y_i} w_i^{t+1}, \quad \forall k. \qquad (3)$$

We have not found an explicit solution for this problem. Nevertheless, the weights may be determined iteratively using the properties of the re-weighting scheme employed in the real Adaboost algorithm [9]. The method is to run Adaboost in a loop for the currently selected feature j. The loop stops when the conditions in (3) are satisfied.



- x_1 is the best feature, x_2 improves results in combination with x_1 but not alone, x_3 is noise
- x_1 is selected first, after reweighting x_2 is selected, reweighting does not help to make x_3 a useful feature

1. **Initialise:** $\mathcal{F} = \emptyset$, $\mathcal{J} = \{1, \ldots, n\}$, $t = 1$, $w_i^t = \frac{1}{2p}$ or $\frac{1}{2q}$ for i: $y_i = -1$ or $+1$, resp., where p and q are numbers of negative and positive samples, $m = p + q$, $\vartheta_0 = 0.005$, $\vartheta_1 = 0.44$, $\vartheta_2 = 0.06$.
2. if $\mathcal{J} = \emptyset$ then stop
3. **Find the best feature:** $j^* = \arg\min_{j \in \mathcal{J}} \varepsilon(\boldsymbol{w}^t, j)$
4. **Detect noise:** if $\varepsilon(\boldsymbol{w}^t, j^*) > \vartheta_1$ then all remaining features in $\mathcal{J}$ are noise, stop
5. **Look for replicas and relevant features:** $\mathrm{tsm}(j^*, t)$
6. goto 2

function $\mathrm{tsm}(j^*, t)$

- $\mathcal{F} = \mathcal{F} \cup j^*$, $\mathcal{J} = \mathcal{J} \setminus j^*$, if $\mathcal{J} = \emptyset$ then return
- **Modify weights:** $\boldsymbol{w}^{t+1} = \mathrm{modify}(\boldsymbol{w}^t, j^*)$
- **Identify replicas:** find $\boldsymbol{j_r} = \{j \in \mathcal{J}:$
  $|\varepsilon(\boldsymbol{w}^{t+1}, j^*) - \varepsilon(\boldsymbol{w}^{t+1}, j)| < \vartheta_0 \wedge$
  $|\varepsilon(\boldsymbol{w}^t, j^*) - \varepsilon(\boldsymbol{w}^t, j)| < \vartheta_0\}$
  all features in $\boldsymbol{j_r}$ are replicas, $\mathcal{J} = \mathcal{J} \setminus \boldsymbol{j_r}$
- **Identify relevant features:**
  $\boldsymbol{\Delta}_j = \varepsilon(\boldsymbol{w}^t, j) - \varepsilon(\boldsymbol{w}^{t+1}, j)$, where $j \in \mathcal{J}$
  find $\boldsymbol{j_\Delta} = \{j \in \mathcal{J}: |\boldsymbol{\Delta}_j| > \vartheta_2\}$
  for j in $\boldsymbol{j_\Delta}$ do
  if $\boldsymbol{\Delta}_j > 0$ then $\mathrm{tsm}(j, t+1)$ else $\mathrm{tsm}(j, t)$
- return

- Complete algorithm reweights and picks best feature but also checks if feautres are noise (θ_1), if features are too similar (θ_0) and consider only relevant features (θ_2). A bit ad hoc:

The proposed feature selection scheme is built on few facts and several empirical observations.

10 experiments with 10-fold crossvalidation

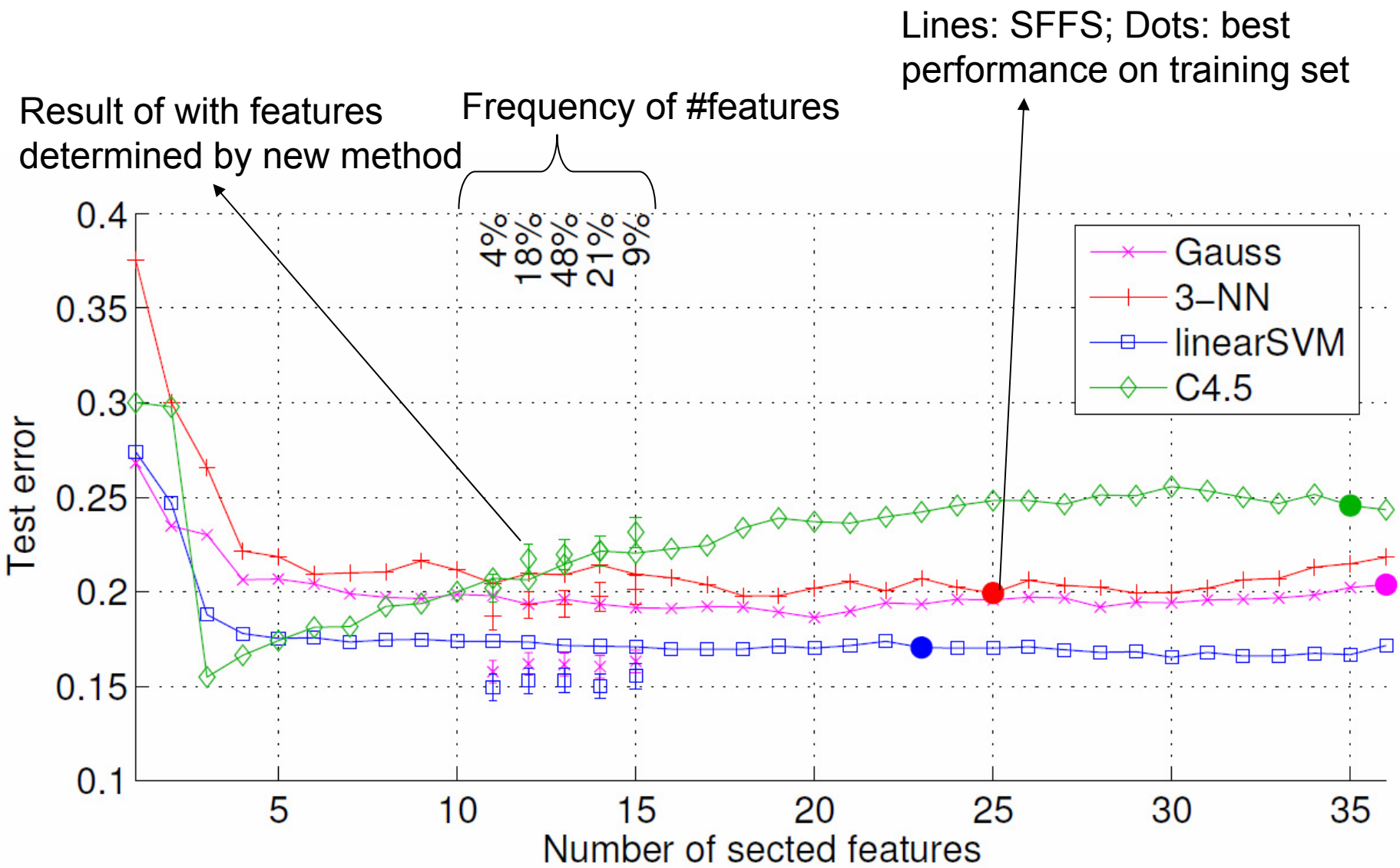


Figure 3. Test error for heart dataset.

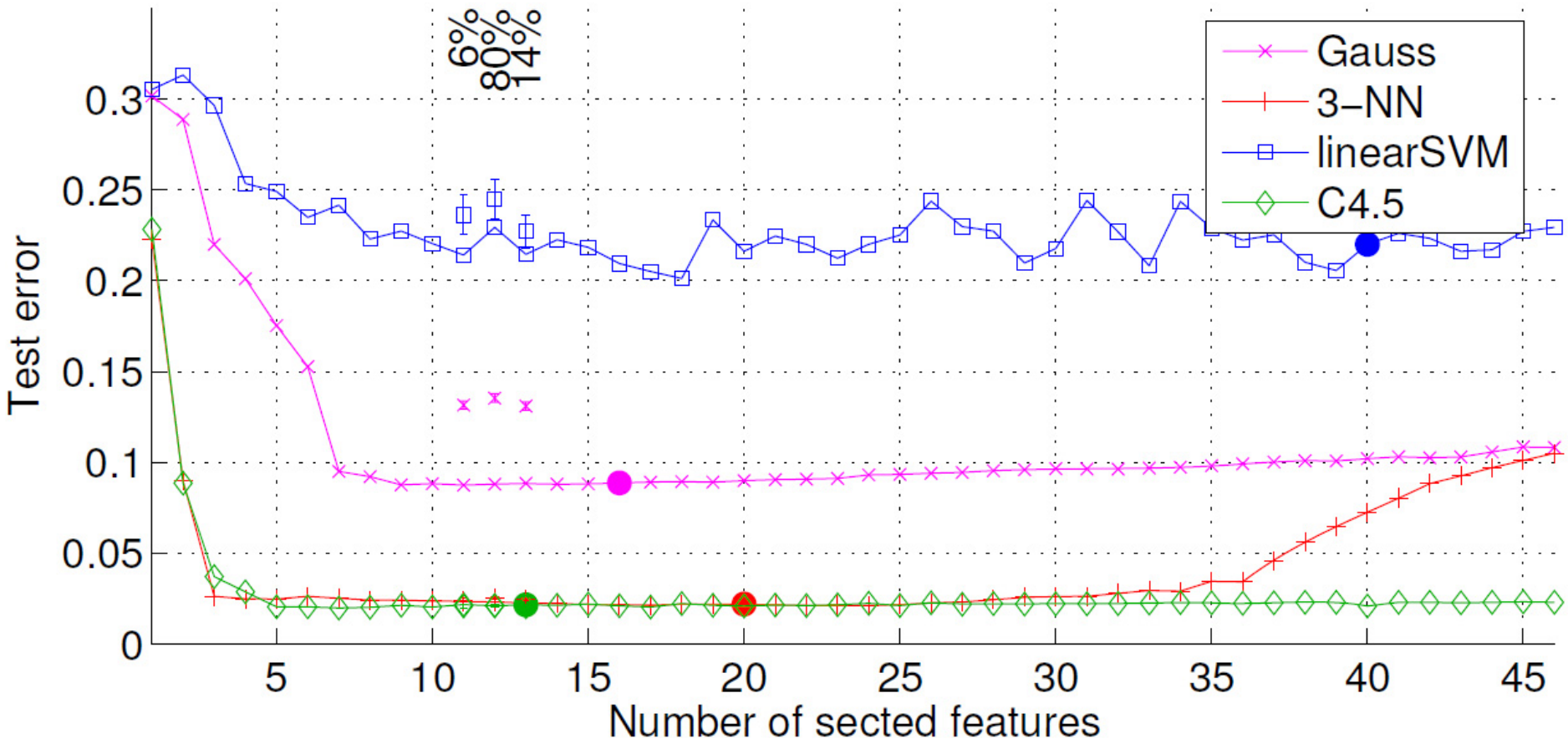


Figure 4. Test error for image dataset.

Summary

- Feature selection important in large real-world PR problems
- Many procedures are somewhat ad hoc
- Computational efficiency often as important as a good result
- Many methods for feature selection exist and give different results on different problems → no single solution