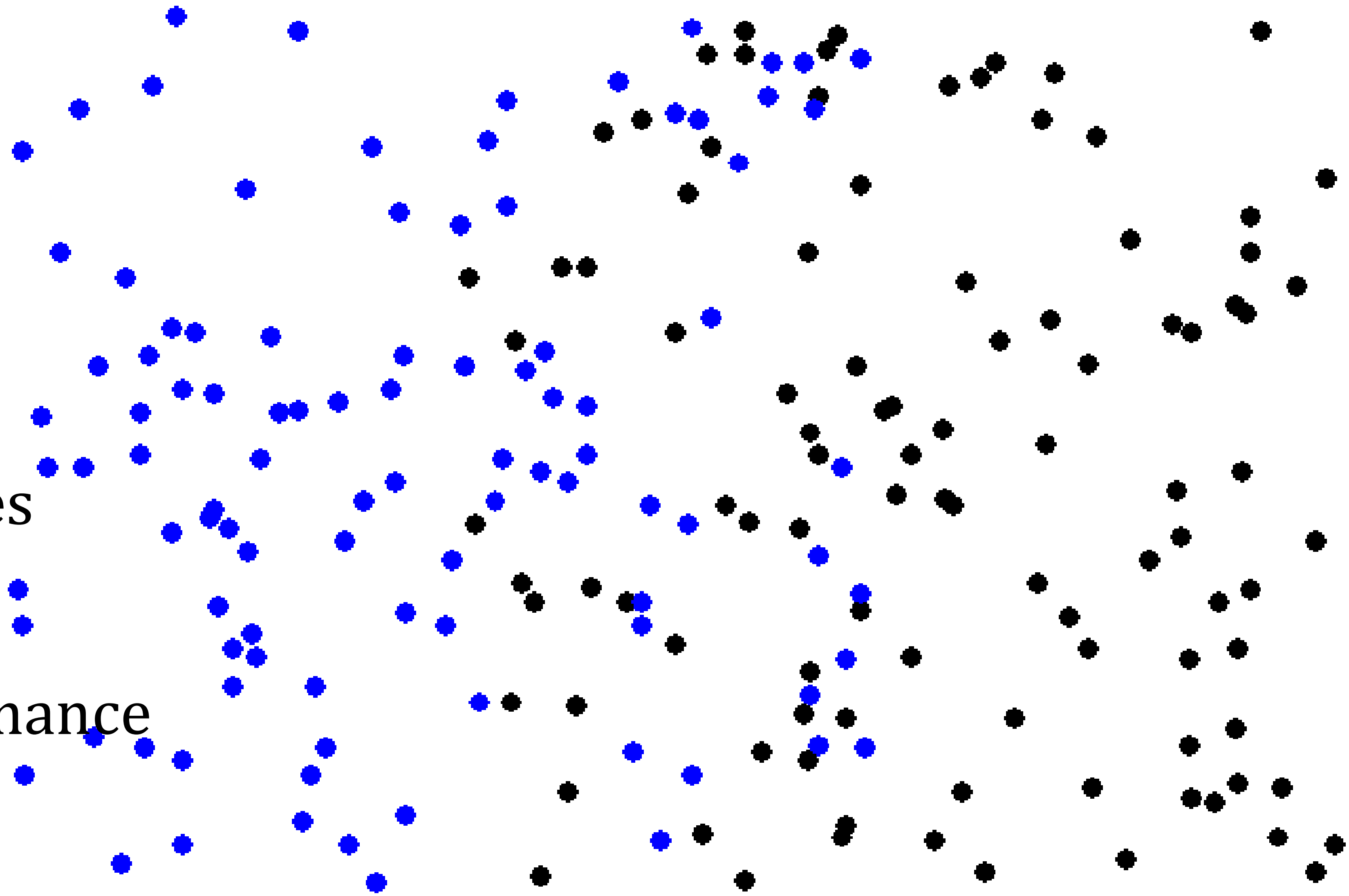
- More Features ⇒ Better Performance ?



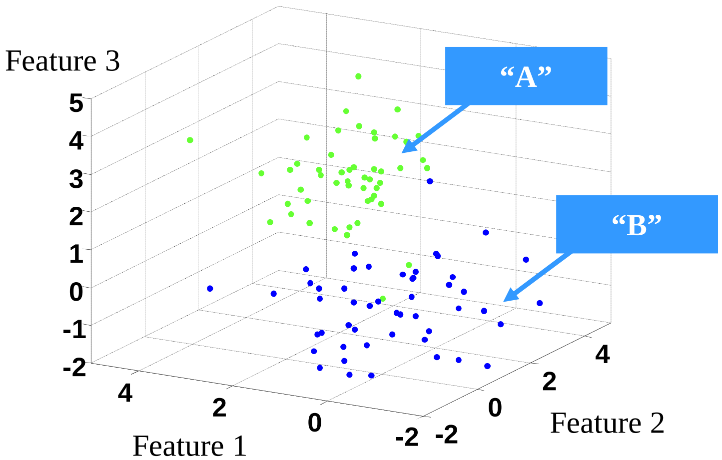
Feature Extraction and Selection

› Marco Loog

Outline

› A bit on high-dimensional spaces
› Some basics of feature extraction
› Some basics of feature selection

Feature Space



› A p-dimensional space, in which each dimension is a feature containing N [labeled] samples [objects]

› Why and how should lower number of features?

Curse of Dimensionality

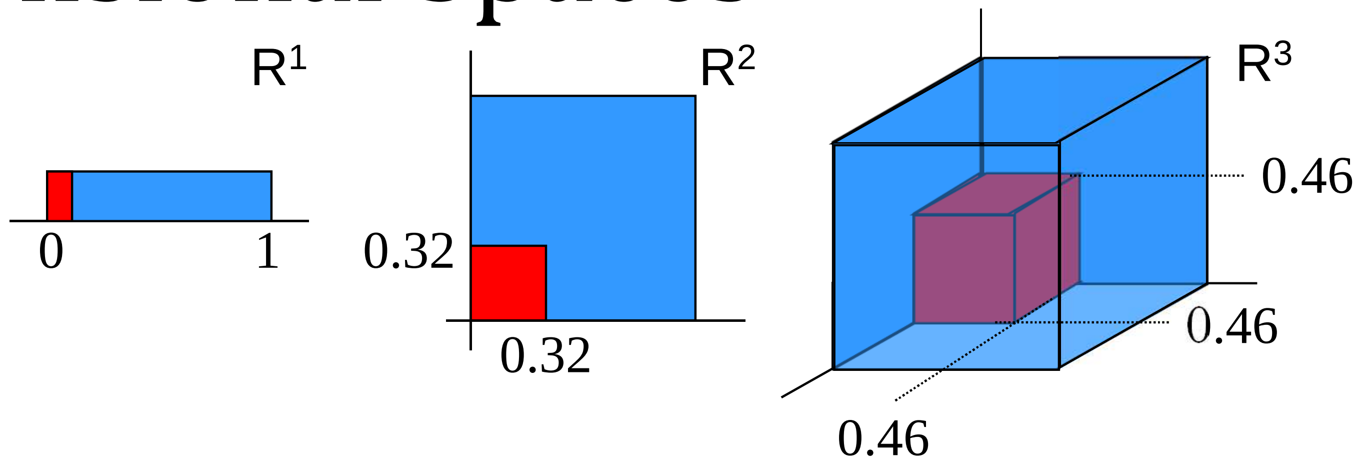
› Problem : too few samples in too many dimensions
[the curse of dimensionality]

Let's discuss histogram-based density estimation
…with increasingly finer binning?

› Anyway : in high-dimensional spaces,
our 2D/3D intuition does not work anymore...

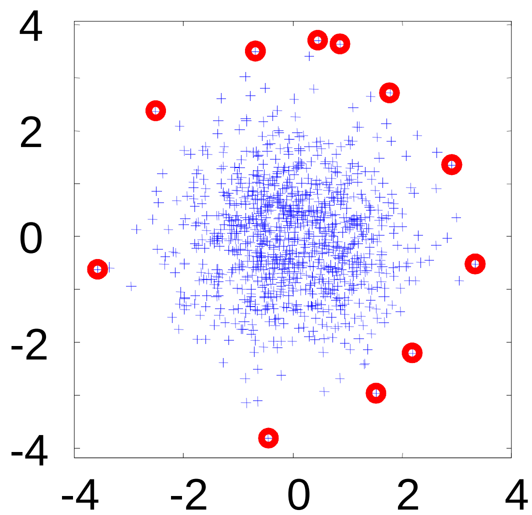
High-Dimensional Spaces

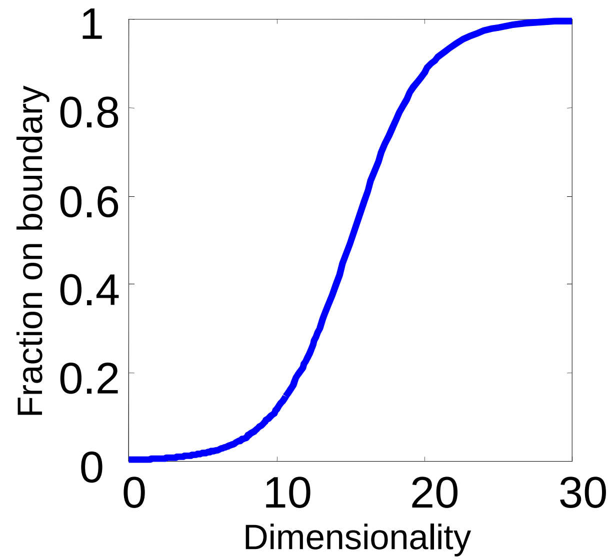
High-Dimensional Spaces



- Example : neighborhood capturing 10% of uniformly distributed data in hypercube
- E.g. in $\mathbb{R}^{20}$ sides of $\sqrt[20]{.1} \approx 0.89$
  - So, not a small block anymore…

High-Dimensional Spaces





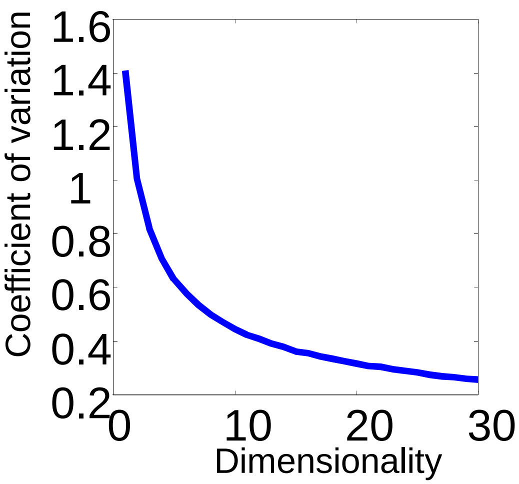
- Example : boundary points
  - 1000 normal samples in 2D then 1% on convex hull
  - 1000 in 20D then 95% on convex hull

High-Dimensional Spaces

› Example : points tend to have equal distances

Consider $\frac{\mathrm{std}(d^2)}{\mathrm{mean}(d^2)}$ for squared distance d^2

For points in $\mathbb{R}^{1000}$ from standard normal, distribution is approximately $N(2000,8000)$



High-Dimensional Spaces

› This means [roughly] for increasing dimensionality

  local, distance-based methods suffer most, e.g. NN-methods

  global, more restricted models suffer less, e.g. linear models

› So…

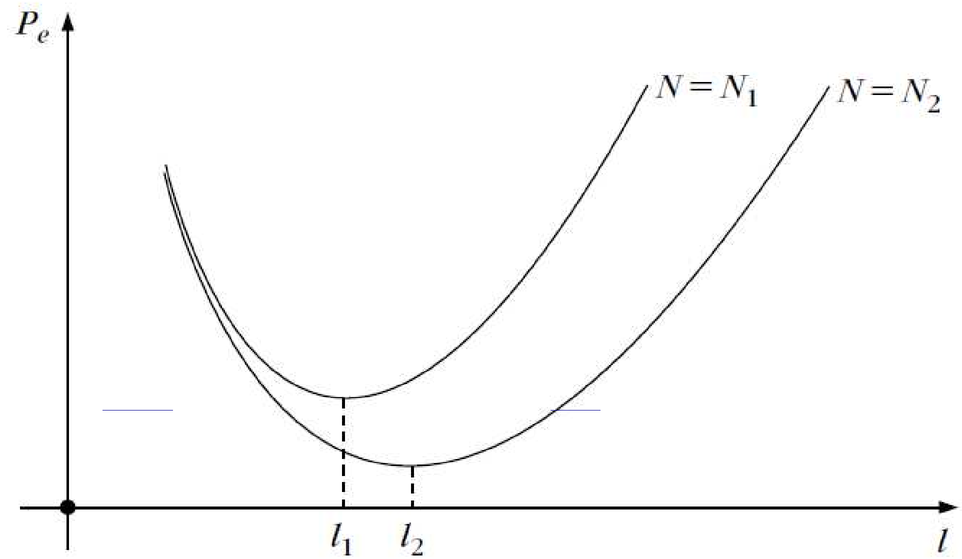
  controlling classifier complexity important

  p should be kept as low as possible : dimensionality reduction

Dimensionality Reduction
by Selection and Extraction

Dimensionality Reduction

- Problem : too few samples in too many dimensions [the curse of dimensionality]
- Solution : drop dimensions / features
  - Feature selection
  - Feature extraction
- Questions :
  - Which dimensions to drop?
  - What feature subset to keep?



Dimensionality Reduction

› Other uses :

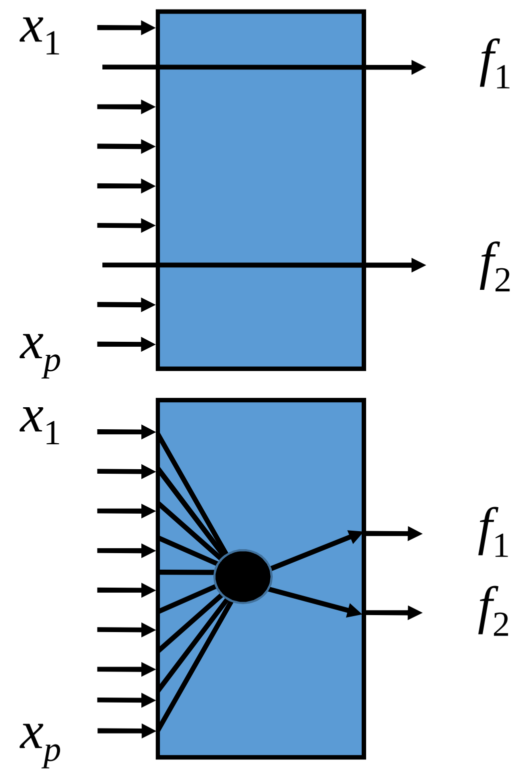
Fewer parameters give faster algorithms and parameters are easier to estimate

Explaining which measurements are useful and which are not [reducing redundancy]

Visualization of data can be a powerful tool when designing pattern recognition systems

Feature Selection vs Extraction

- Feature selection : select d out of p measurements
- Feature extraction : map p measurements to d measurements



Feature Selection vs Extraction

› Think of selection and extraction as finding a mapping

› We need :

Criterion function, e.g. error, class overlap, information loss,…

Optimization or "search" algorithm to find mapping for given criterion

Note on Criteria

- The optimal[?] criterion :
  final performance of the entire system
  - Maybe calculated using cross-validation

- Approximate performance predictors
  - Calculate performance of easy-to-use criterion giving indication of how well a more powerful / realistic criterion may perform

Two
Classical
Linear
Feature Extractors

Linear Feature Extraction

› Unsupervised :

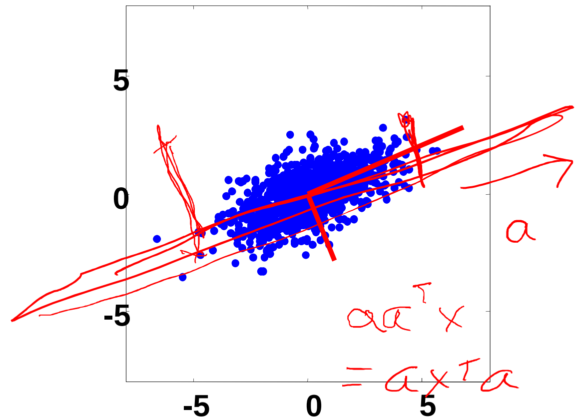
Principal Component Analysis [PCA]

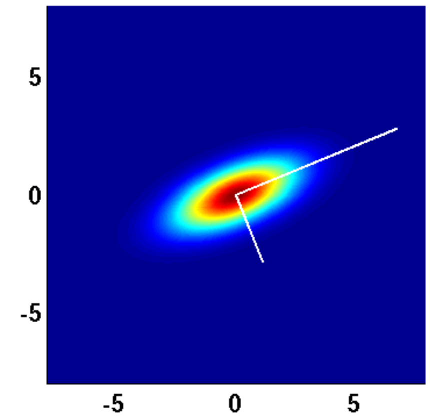
› Supervised :

Linear Discriminant Analysis [LDA]
Fisher mapping [fisherm]

PCA is *the* most widely used feature extraction method
…LDA might be a good second
Similar ideas are at the basis of many “novel” methods

PCA





- Principal component analysis [PCA, 1901] : find directions in data which…
  - Retain as much [total] variance as possible
  - Make projected data uncorrelated
  - Minimize squared reconstruction error

PCA

- Let a be a projection vector that reduces to 1D
- Let's say our data has covariance matrix C

  What value does $a^T Ca$ equal to? $a = \begin{pmatrix} 0 \\ 10 \\ 0 \\ 0 \end{pmatrix}$

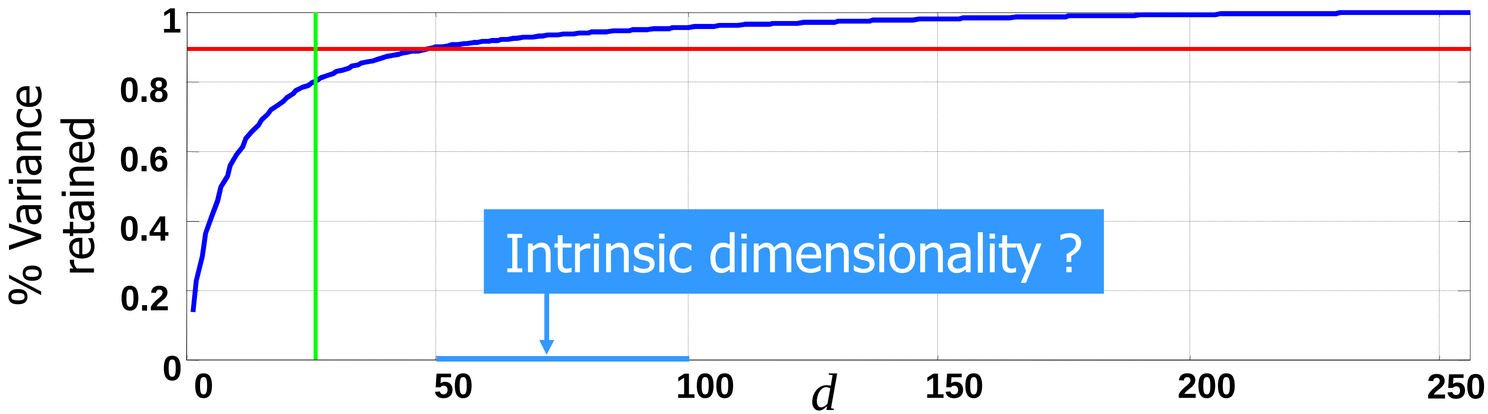
  So, what should we maximize?

- Seems we need an assumption…

  Assume $\|a\| = 1$

  Then solve, for instance, with Lagrangian : $a^T Ca - \lambda(\|a\| - 1)$

PCA Example

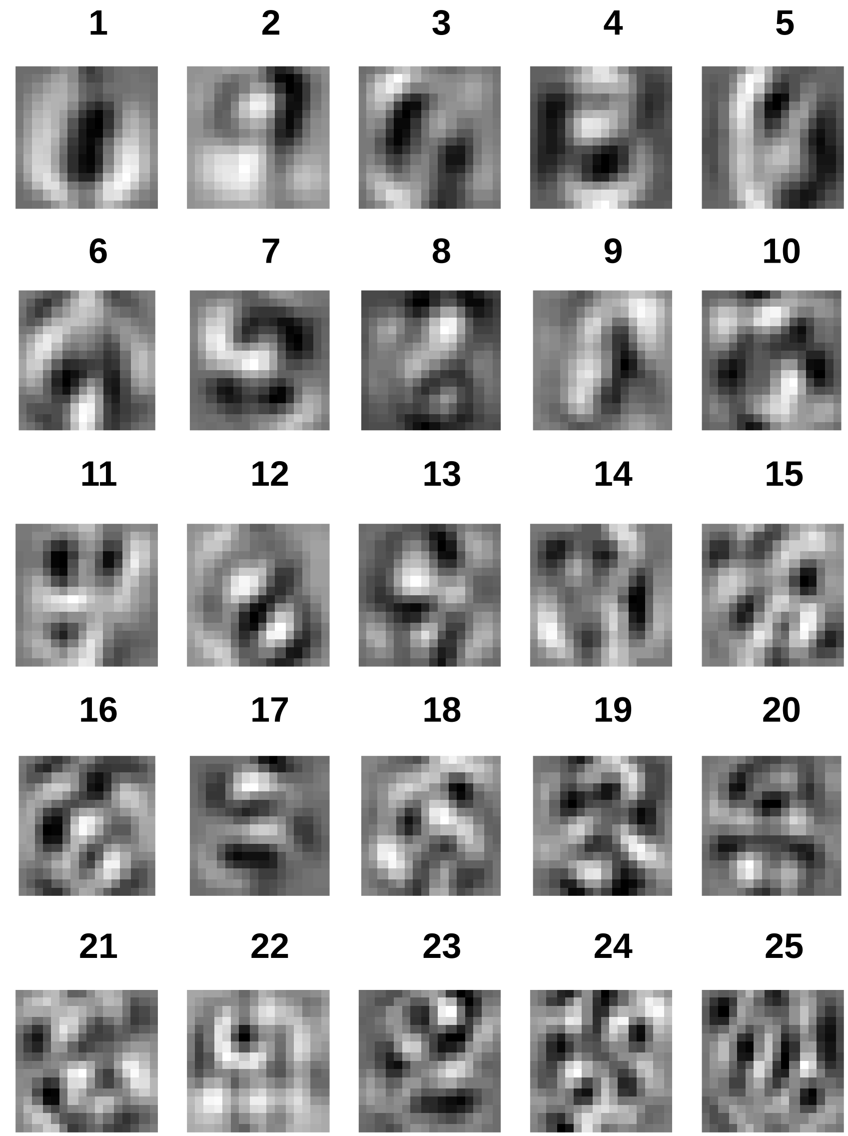


› E.g. NIST digits : 2000 samples, $p = 256$

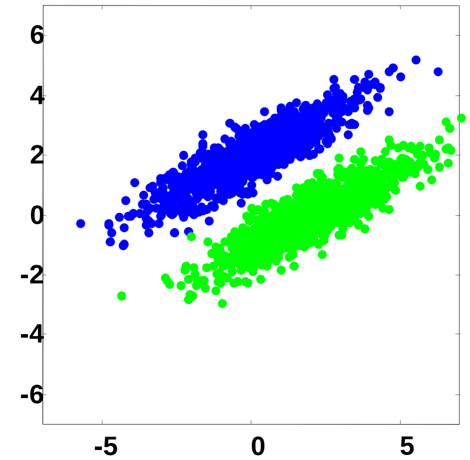
PCA Example

› For image data, principal components might[!] also be interpretable…

› Here : largest occuring variations between digits



Remarks on PCA



› Principal component analysis :

Global and linear

May need considerable amount of data
to estimate covariance [C, Σ, S_T,…] well

› Danger :

Criterion is not necessarily related to the goal

E.g. might discard important directions

[Then again, most classifier also do not optimize error rate directly…]

Intermezzo : Scatter Matrices

- $m, S_T = \Sigma$: mean and covariance of all samples
- m_i, Σ_i : mean and covariance of class i

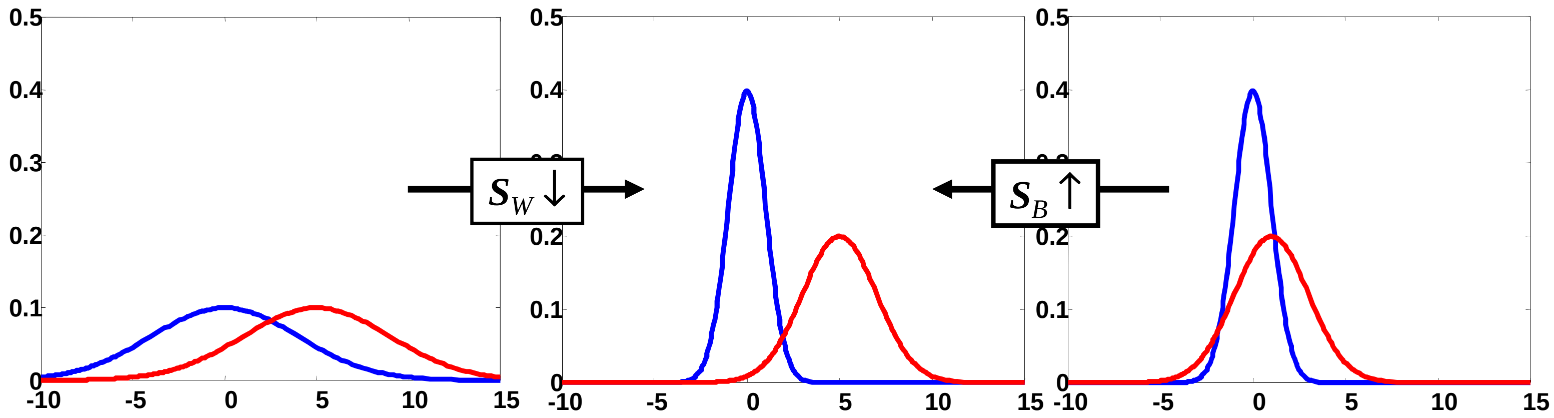
- Total scatter : Σ equals sum of within and between
- Within-scatter : $\boldsymbol{S}_w = \sum_{i=1}^{C} \frac{n_i}{n} \boldsymbol{\Sigma}_i$
- Between-scatter : $\boldsymbol{S}_B = \sum_{i=1}^{C} \frac{n_i}{n} (\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^T$

Intermezzo : Scatter Matrices

› S_T = total scatter, “overall width”

› S_W = “average class width”; the smaller, the better

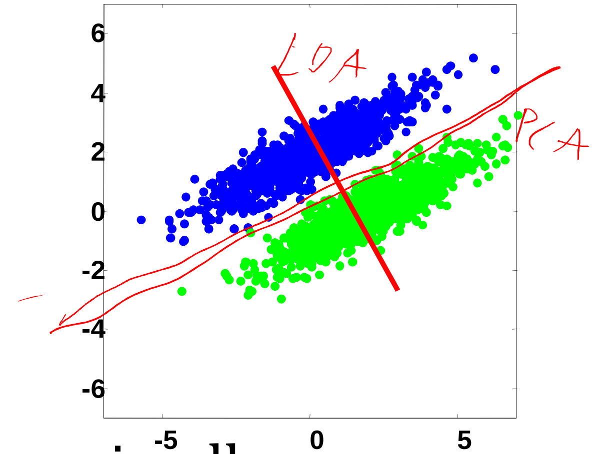
› S_B = “average distance between class means”;
the larger, the better



Supervised Linear Feature Extraction

- If desired output is given, supervised criteria can be used
  - One illustration only : Linear Discriminant Analysis [LDA, or in PRTools terms fisherm]

LDA [or Fisher mapping]



› Reduction to 1D for two classes

Find projection vector a such that classes are maximally separated

Choose a to maximize Fisher criterion :

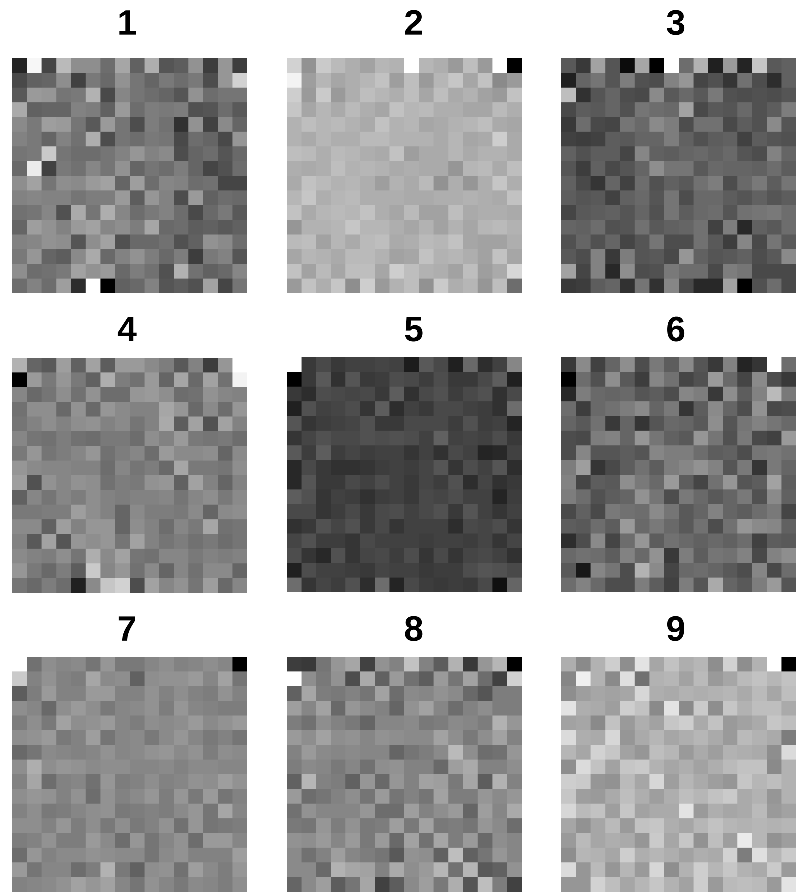
$$J_F(a) = \frac{a^T S_B a}{a^T S_W a}$$

› Solution: Eigenanalysis of $S_W^{-1} S_B$

LDA

› Map down to a maximum of $K - 1$ dimensions

› [Why?]

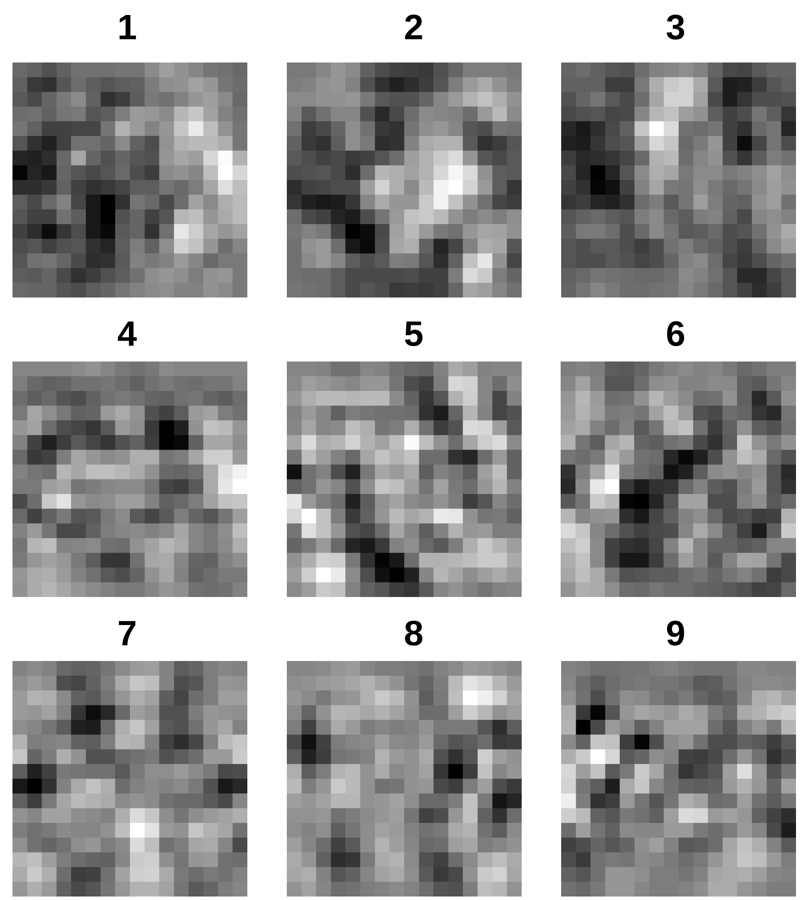


Example : NIST digits

LDA

› To avoid fitting noise, can do PCA first

  If system underdetermined $[n \leq p]$, first doing PCA may even be necessary



Nonlinear Feature Extraction

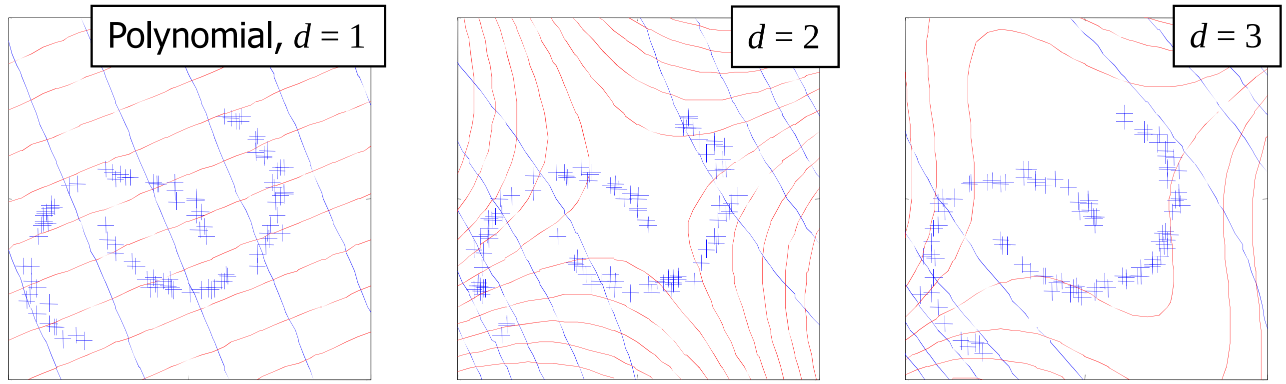
- Large collection of possible mappings…

- Today : only one unsupervised method

  What is *the* way to make linear stuff nonlinear?

- Tomorrow : [the gist of] so-called auto-encoders

KPCA



› Kernelize PCA by relating eigenvectors of $X^T X$ and XX^T [assuming centralized feature vectors]

Summary

› Feature extraction, like selection :

Useful for visualization

Necessary because of curse of dimensionality

› Feature extraction :

Linear vs. nonlinear

Supervised vs. unsupervised

› PCA possibly most important method

On to Feature Selection

Feature Selection

› The general idea is to pick a set of good features from the original/initial set of features

Feature Selection

- We need a criterion function
  - How do we measure how good a feature subset is?
  - E.g. error, class overlap, information loss
- We need a search algorithm
  - How do we go through all possible subsets?
  - E.g. pick best single feature at each time

- Maybe more than for feature extraction : optimality is sacrificed

Feature Selection

› Which approach to feature selection have we seen?

Criteria

› Actual classification performance : “best” possible criterion, but potentially very expensive

› Approximate performance predictors : calculate easy-to-use measure that gives indication of real performance

Probabilistic Criteria

$$D(p\|q) = \int p(x) \log \frac{p(x)}{q(x)} dx$$

- Probabilistic distance
- Expresses distance between two distributions
  - E.g. Kullback-Leibler divergences and variations
- Often needs estimates of class-conditional densities
  - Potentially difficult
  - Potentially expensive

Scatter Matrices Again…

- $m, S_T = \Sigma$: mean and covariance of all samples
- m_i, Σ_i : mean and covariance of class i

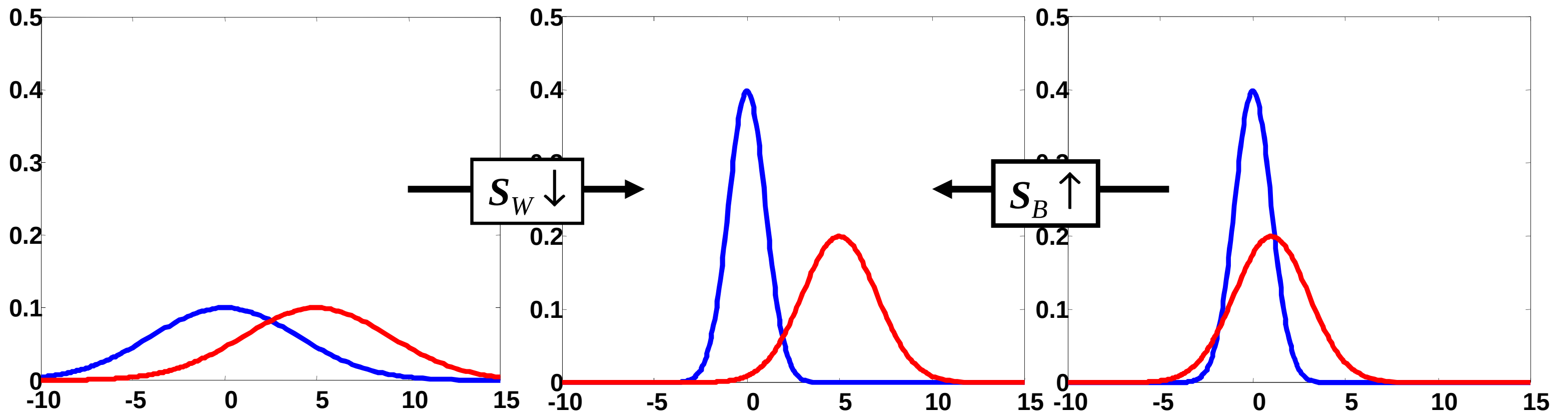
- Total scatter : Σ equals sum of within and between
- Within-scatter : $\boldsymbol{S}_w = \sum_{i=1}^{C} \frac{n_i}{n} \boldsymbol{\Sigma}_i$
- Between-scatter : $\boldsymbol{S}_B = \sum_{i=1}^{C} \frac{n_i}{n} (\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^T$

Scatter Matrices Again…

› S_T = total scatter, "overall width"

› S_W = "average class width"; the smaller, the better

› S_B = "average distance between class means"; the larger, the better

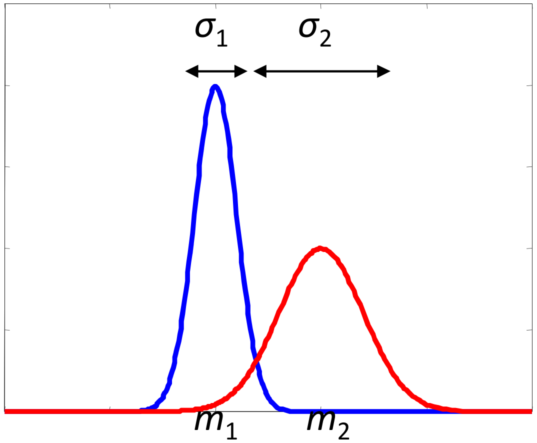


Heuristic Scatter-based Criteria

› Scatter-based performance indicators

$$J_1 = \text{trace}\,(\boldsymbol{S}_W + \boldsymbol{S}_B) = \text{trace}\,(\boldsymbol{\Sigma})$$
$$J_2 = \text{trace}\,(\boldsymbol{S}_B \,/\, \boldsymbol{S}_w)$$
$$J_3 = \det\,(\boldsymbol{\Sigma}\,) \,/\, \det\,(\boldsymbol{S}_w)$$
$$J_3 = \text{trace}\,(\boldsymbol{S}_W\,) \,/\, \text{trace}\,(\boldsymbol{S}_B)$$

Yet Another Criterion



› Mahalanobis distance

$$D_M = (m_1 - m_2)^T C^{-1} (m_1 - m_2)$$

Assumes Gaussian distributions with equal covariance matrix C

In which case, some of the probabilistic distances reduce to this

› Multi-class, e.g. take sum or minimum

Of course, general solution to extend two-class criteria

› 1D case : Fisher criterion $J_F = \frac{(m_1 - m_2)^2}{\sigma_1^2 + \sigma_2^2}$

Sub-optimality of Criteria

$$D_M = (m_1 - m_2)^T C^{-1}(m_1 - m_2)$$

- Give a 2D problem in which Euclidean distance [this assumes $C = I$] picks up the wrong 1D feature…
- Give a 2D problem in which Mahalanobis distance picks up the wrong 1D feature…

Now, the Search Algorithms

- Feature selection : Select a subset of d out of p measurements which optimizes chosen criterion
- Simplest solution : look at all possible subsets

  Any problems there?

- There are $\binom{p}{d} = \frac{p!}{d!(p-d)!}$ subsets

  So, like for the criteria, we settle for approximations…

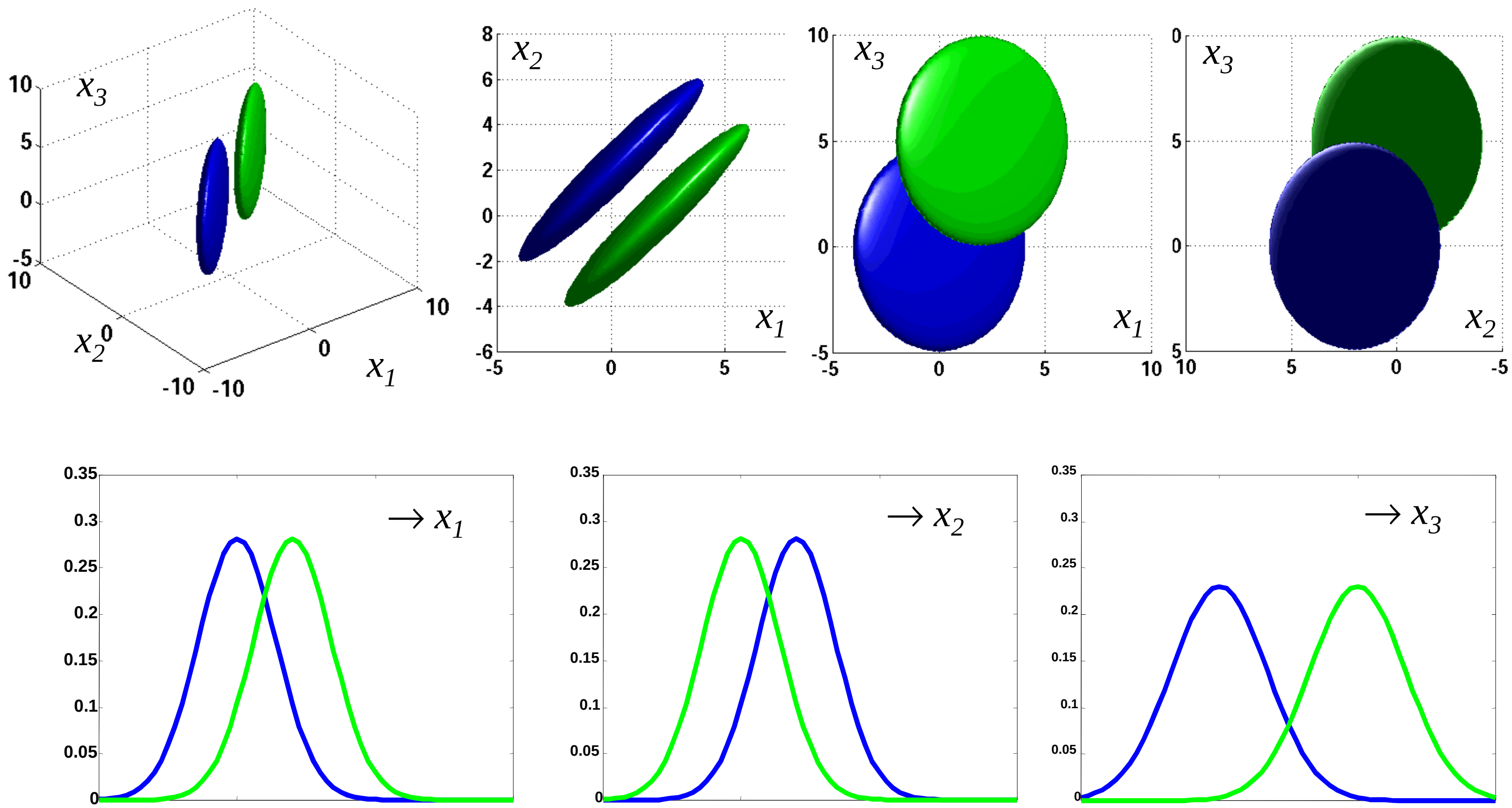
Search Algorithms

- Sub-optimal algorithms : select one feature [or a few features] at a time

- Simplest variation : best individual d

  But these are not necessarily the best d!

d Best or Best d?



More Sub-Optimal Strategies

(1, 3)

(1, 2, 3)
(1, 3, 4)
(1, 3, 5)

› Forward selection

Start with empty feature set

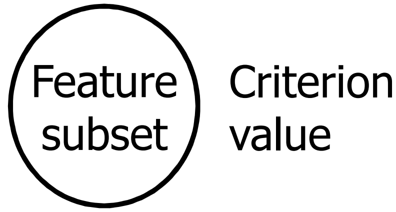
One at a time, keep adding feature that gives best performance considering entire chosen feature set

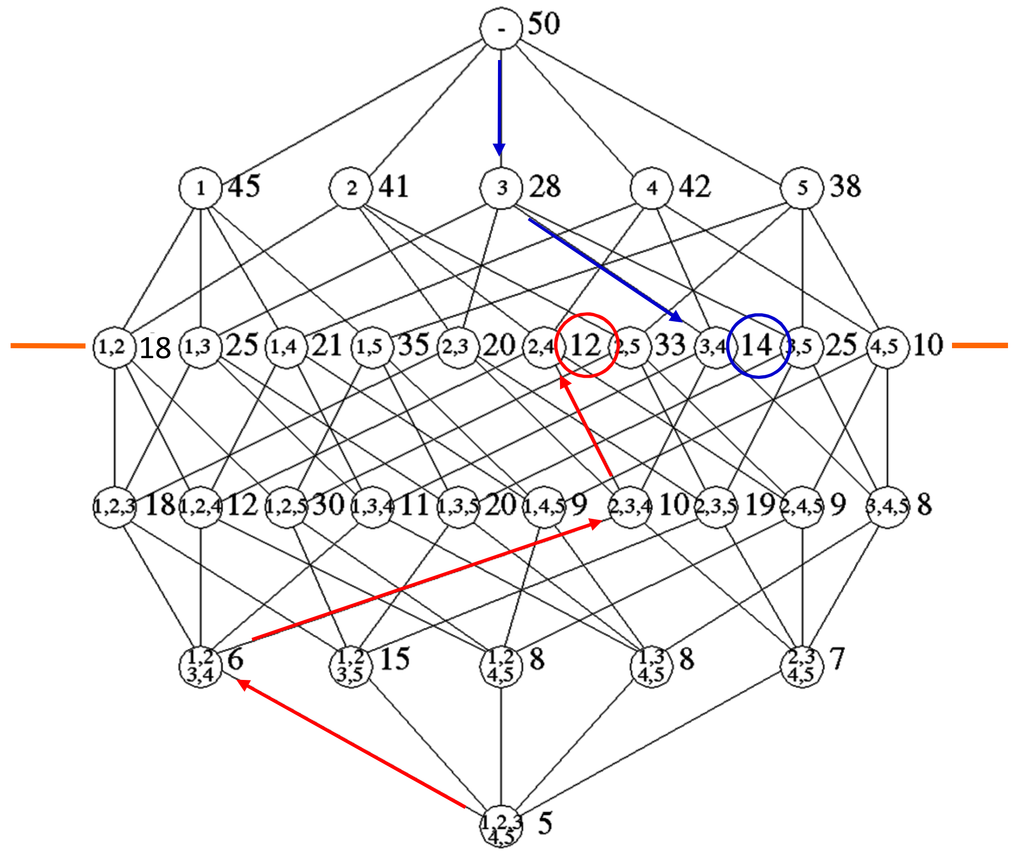
More Sub-Optimal Strategies

› Backward selection

Same as forward selection
...but then the other way ‘round

E.g.





› Select $d = 2$ out of $p = 5$ features

More Sub-Optimal Strategies

› Plus-l-take-away-r

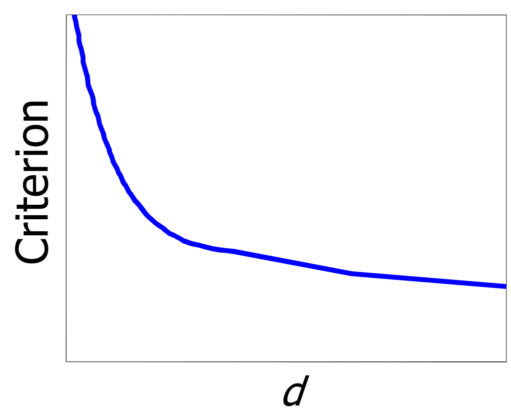
Start with empty set [if $l > r$] or entire set [if $l < r$]

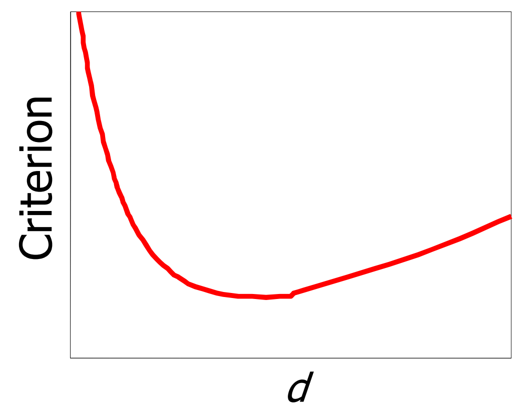
Keep adding best l and removing worst r
[...or vice versa]

Benefit over previous strategies?

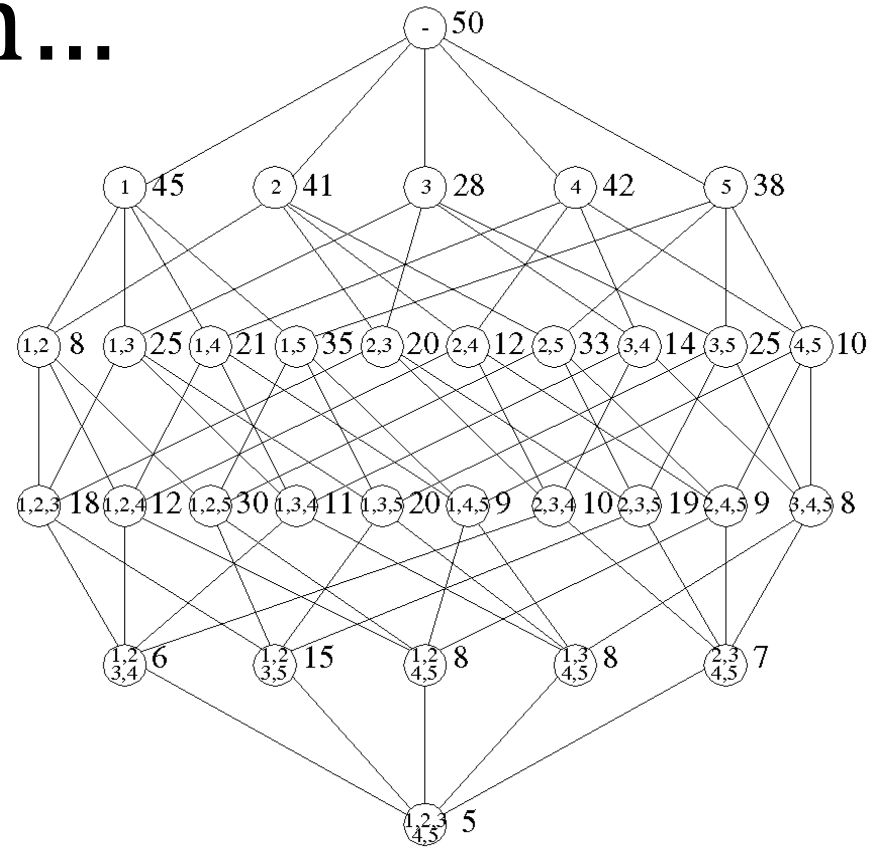
Branch & Bound

- Branch & bound
- Optimal when criterion is monotonic in number of features d



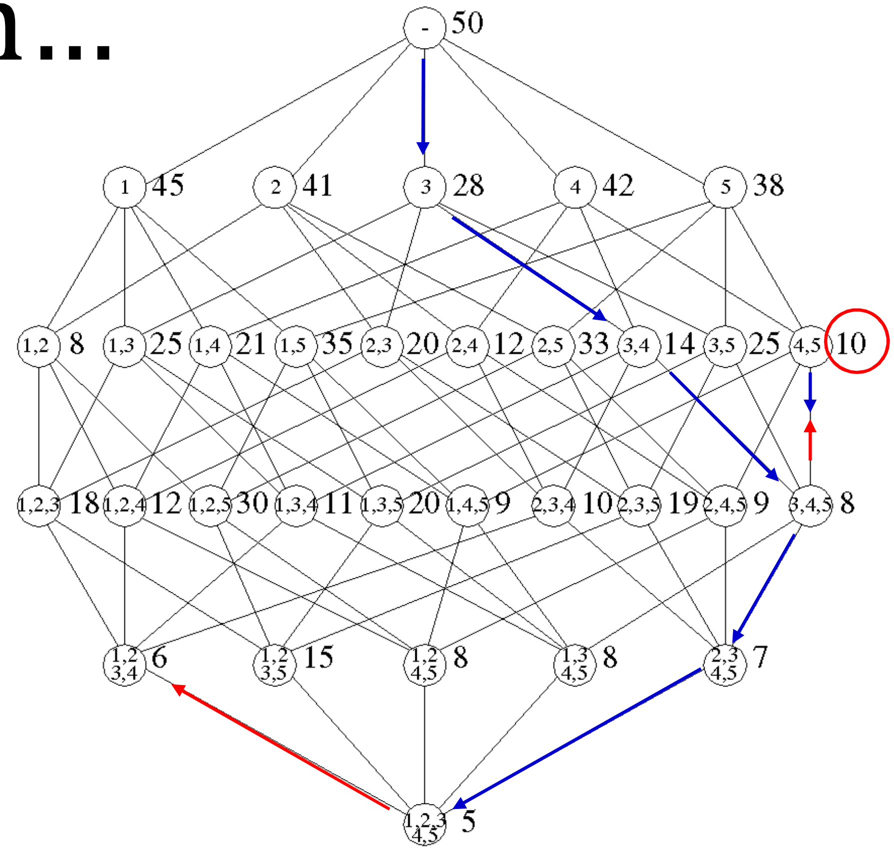


Floating Search…



› Backtrack if that improves criterion for given # features

Floating Search…

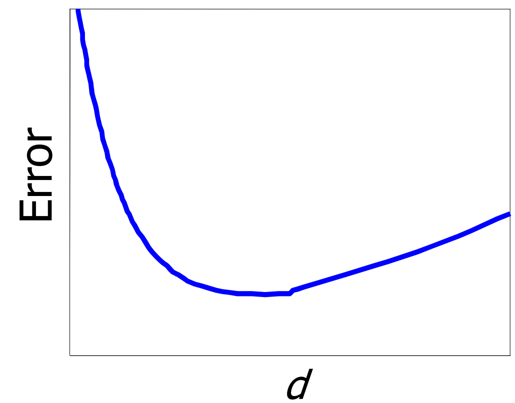


When Should One Stop?

- Due to estimation problems [e.g. covariance matrix], criterion may have an optimum

- Or we could specify desired number of measurements, say, based on data set size

- Or? Use error rate?



Discussion / Conclusion

- Some unexpected behaviors in higher dimensions
- Considered curse of dimensionality
  - Way to counter it and improve performance : lower dimensionality
- Linear approaches to dimensionality reduction
  - Feature selection and feature extraction
  - Feature extraction can be nonlinear…
- Approaches are approximative / suboptimal
  - But that holds for many classifiers to start with…

› Q?