INF 5300 - Feature selection in the context of supervised classification

- «Curse of dimensionality»
- Feature subset selection
  - Objective functions
  - Search strategy

Sections 5.1, 5.2, 5.5 (5.5.1 and 5.5.2 not too detailed), 5.6 in "Pattern Recognition" by S. Theodoridis and K. Koutroumbas.

For the randomized methods: Computational methods of feature selection, Liu, 2007, chapter Randomized Feature Selection, especially sections 6.4 and 6.5.

(see links on the course's web-page for pdfs)

Reminder - Density-based classifiers

Model/estimate $p(\mathbf{x}|c)$

Prior probability for class c

$$P(c|\mathbf{x}_i) = \frac{p(\mathbf{x}_i|c)P(c)}{p(\mathbf{x}_i)}$$

Here this is our discriminant function

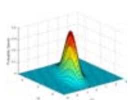
Bayes' rule

Our decision rule

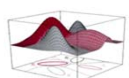
$$f(\mathbf{x_i}) = \underset{c}{\text{argmax}}\ p(c|\mathbf{x_i})$$

$p(\mathbf{x}|c)$

- Gaussian/normal
  - Spherical (nearest mean)
  - Uncorrelated (naive Bayes)
  - Model-restrained
  - Equal-shaped classes
  - Full

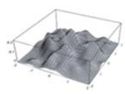


- Mixture models (e.g. mixture of Gaussians)



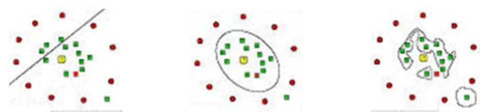
More complex

- Kernel-based (e.g. Parzen-windows)



* Illustrations from Duda et al

Reminder – Classifier complexity



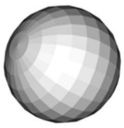
High dimensionality / low sample count

- Even the simple unimodal normal distribution can be too complex

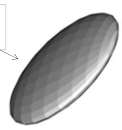
$$g_c(\mathbf{x}) = -\frac{1}{2}\log|\Sigma_c| - \frac{1}{2}(\mathbf{x}-\mu_c)'\Sigma_c^{-1}(\mathbf{x}-\mu_c) + \log \pi_c$$

n elements

n(n+1)/2 elements



3D (n=3) equidensity contours

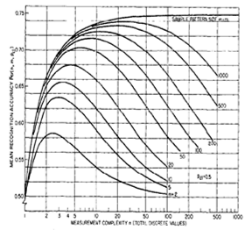


- *Overfits* easily causing poor generalization

Performance on "unseen" data

«Curse of dimensionality» I / II

- Peaking phenomenon
  - Finite number of training samples
  - Adding (even discriminative) features
  - => Eventually worse classification rate
- High dimension -> mostly empty space



Increased dimension of **x** →

* Illustration from Hughes

«Curse of dimensionality» II / II

- Handling the problem
  - Careful feature design to start with!
  - Dimension reduction
    - Feature selection
    - Feature extraction
  - Reduce classifier complexity
    - E.g. assuming diagonal Σ for Guassian-based classifiers
  - Biasing/regularization
    - E.g. diagonal loading for Guassian-based classifiers
  - Sometimes adding unlabeled samples help (semi-supervised classification)

Today's topic

Cf. slide 3; moving upwards

Somewhat of a digression..

Regularized discriminant analysis

Class-specific sample covariance matrix

Common-class sample covariance matrix

$$\hat{\Sigma}_c = (1-\alpha)\tilde{\Sigma}_c + \alpha\big((1-\gamma)\tilde{\Sigma} + \gamma\sigma^2 I\big)$$

Use this as covariance matrix for class *c*

α and γ guides transition between full quadratic classifier, linear classifier and nearest-mean classifier

Scaled identity matrix (think spherical distribution)

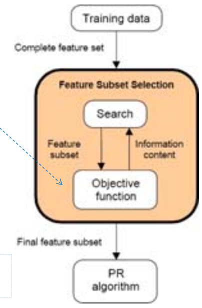
Feature (subset) selection intro

- Why?
  - Enhanced generalization by reducing overfitting
  - Improved model interpretability
  - Computationally tractable dataset
- Three main approaches:
  - «Wrappers»
    - The optimization criterion is based on building and testing actual classifiers
  - «Filters»
    - Criterion is based on a (simplified) class-separability measure
  - «Embedded methods»
    - The classifier itself induces feature selection, e.g. decision threes

Feature selection

- Want the best m out of n feature subset
- Our search needs an objective function
  - "Predict" classifier performance
  - Decides how good a subset is
  - Wrapper-based
    - Often good, often slow
    - Linked to specific classifier
  - Filter-based
    - A proxy measure
    - «Simple» function
    - Fast(er)
    - Might be more general

Remember separate training/vadiation set (cf. cross-validation)



Here non-wrapper-based!

Objective functions I/II

- Want a function that can predict good classifier performance
- E.g. for two classes:
  - Euclidean distance between class means $|\mu_1-\mu_2|$
  - Mahalanobis distance between class means

Note!

«Gaussian + common Σ + divergence»

$$\Delta = (\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2)$$

  - Assume Gaussianity and calculate divergence (eq. 5.14 in Theodoridis)
  - Assume Gaussianity and calculate Bhattacharyya distance (linked to minimum attainable Bayes error rate)

$$B = \frac{1}{8}(\mu_r - \mu_s)^T \left(\frac{\Sigma_r + \Sigma_s}{2}\right)^{-1} (\mu_r - \mu_s) + \frac{1}{2}\ln\frac{\left|\frac{1}{2}(\Sigma_r + \Sigma_s)\right|}{\sqrt{|\Sigma_r||\Sigma_s|}}$$

«Divergence»: Here a distance measure between pdfs

  - More general pdfs → quickly more computationally challenging

Objective functions II/II

- Multiple (>2) classes, e.g.:
  - Average objective-function value of all pairs of classes
  - Smallest value between a pair of classes
  - ...

Often used

Search strategies

- Want the best m out of n feature subset
- Exhaustive search implies $\binom{n}{m}$ evaluations if we fix *m*, and 2^n if we need to search all possible *m* as well
  - Choosing *10* out of *100* will result in $\sim 10^{13}$ queries
- Obviously we need to guide the search / use suboptimal search techniques

In some cases (monotonic cost functions) one can compute the optimal for larger *n* and *m*s (cf. *branch-and-bound* methods).

$$\binom{m}{l} = \frac{m!}{l!(m-l)!}$$

Method 1 - Individual feature selection

- Each feature is treated separately (no correlation/dependence between features is considered)
  - Select a criterion/objective function
  - Calculate the objective function, C(k), for each feature k
  - Select the set of features with the best individual criterion value
- Advantage with individual selection: Computation -- It's fast!
- Disadvantage: Ignores feature dependence/complementary information

If this topic comes up during the exam, we will ask you to illustrate this by a toy example!

Method 2: Sequential forward selection

- Algorithm:
  - Compute the criterion value for each feature. Select the feature with the best value, say x_1.
  - Form all possible combinations of features x_1 (the winner at the previous step) and a new feature, e.g. $[x_1,x_2]$, $[x_1,x_3]$, $[x_1,x_4]$, etc. Compute the criterion and select the best one, say $[x_1,x_3]$.
  - Continue with adding a new feature until desired number reached.
- Number of combinations searched when selecting l out of m: lm-l(l-1)/2.
- Disadvantage: Unable to remove features that become obsolete after including more features

That is, we can end up in a local minimum/maximum

Method 3 - Sequential backward selection

- Example: Want 2 out of 4 features x_1,x_2,x_3,x_4
  - Choose a criterion/objective function C
  - Eliminate one feature at a time by computing C for $[x_1,x_2,x_3]^T$, $[x_1,x_2,x_4]_T$, $[x_1,x_3,x_4]^T$ and $[x_2,x_3,x_4]^T$
  - Select the best (highest C) combination, say $[x_1,x_2,x_3]^T$.
  - From the selected 3-dimensional feature vector eliminate one more feature, and evaluate the criterion for $[x_1,x_2]^T$, $[x_1,x_3]_T$, $[x_2,x_3]^T$ and select the one with the best value.
- Number of combinations searched when selecting l out of m features 1+1/2((m+1)m-l(l+1))
  - Backwards selection is faster if l is closer to m than to 1.
- Disadvantage:
  - Discarded features might have been deemed useful at a later stage
  - High starting dimensionality might put restrictions on objective function

Method 4: Plus-L Minus-R Selection (LRS)

If $L > R$, LRS starts from the empty set and repeatedly adds L features and removes R features
If $L < R$, LRS starts from the full set and repeatedly removes R features followed by L feature additions

Algorithm

1. If $L > R$ then start with the empty set $Y = \emptyset$ else start with the full set $Y = X$ goto step 3
2. Repeat SFS step L times
3. Repeat SBS step R times
4. Goto step 2

LRS attempts to compensate for weaknesses in SFS and SBS by backtracking

How to decide on L and R?

Method 5: Floating search methods I/II

- Similar to plus-L minus-R selection, although with adaptive number of backtrackings
- The dimensionality (number of features) «floats»
- Both forward (SFFS) and backwards (SFBS) versions
- Basic idea for the forward version:
  - Repeat until desired number of features is found:
    - Do a forward step by adding a feature
    - Continue deleting features as long as the results improve (for sets of equal size)
- Provides good results at an «affordable» computational cost

Method 5: Floating search methods II/II

SFFS Algorithm

Input:
$Y = \{y_j \mid j = 1, \ldots, D\}$ //available measurements//
Output:
$X_k = \{x_j \mid j = 1, \ldots, k,\ x_j \in Y\},\ k = 0, 1, \ldots, D$
Initialisation:
$X_0 := \emptyset; \quad k := 0$
(in practice one can begin with $k = 2$ by applying SFS twice)
Termination:
Stop when k equals the number of features required

Step 1 *(Inclusion)*
$x^+ := \arg\max_{x \in Y - X_k} J(X_k + x)$ $\left\{\text{the most significant feature with respect to } X_k\right.$
$X_{k+1} := X_k + x^+; \ k := k + 1$
Step 2 *(Conditional Exclusion)*
$x^- := \arg\max_{x \in X_k} J(X_k - x)$ $\left\{\text{the least significant feature in } X_k\right.$
if $J(X_k - \{x^-\}) > J(X_{k-1})$ **then**
$X_{k-1} := X_k - x^-; \ k := k - 1$
go to Step 2
else
go to Step 1

Randomized methods

- Why? Try to avoid local optima. Emphasis on try; no guarantee!
- Apply previously-mentioned techniques and randomize over; input samples, input features, starting point.
- Popular general-purpose strategies:
  - Simulated annealing
  - Genetic algorithms

Method 6: Simulated annealing I/II

- Named after annealing in metallurgy
  - Heat and then slowly cool to allow atoms to settle into more optimal crystalline structures
- Simulated annealing as optimization strategy
  - High «temperature» → move current solution more readily further away, and more readily accept a «worse» solution
  - Gradually reduce temperature while iterating
- Needs
  - Initial solution, temperature and cooling rate
  - A «getNeighbor(state, temp)» function
  - An (decent) objective function (of course!)

Method 6: Simulated annealing II/II

Given:

Examples $\mathbf{X} = < \mathbf{x}_1, y_1 >, \ldots < \mathbf{x}_m, y_m >$
Annealing schedule, T_0, T_{final} and ΔT with $0 < \Delta T < 1$
Feature subset evaluation function $Eval(\cdot,\cdot)$
Feature subset neighbor function $Neighbor(\cdot,\cdot)$

Algorithm:

$S_{best} \leftarrow$ random feature subset
while $T_i > T_{final}$ **do**
  $S_i \leftarrow Neighbor(S_{best}, T_i)$
  $\Delta E \leftarrow Eval(S_{best}, \mathbf{X}) - Eval(S_i, \mathbf{X})$
  if $\Delta E < 0$ **then** *//if new subset better*
    $S_{best} \leftarrow S_i$
  else *//if new subset worse*
    $S_{best} \leftarrow S_i$ with probability $\exp(\frac{-\Delta E}{T_i})$
  $T_{i+1} \leftarrow \Delta T \times T_i$
return(S_{best})

FIGURE 1.6: A basic simulated annealing algorithm.

(From Computational Methods of Feature Selection, Liu, 2007.)

Method 7: Genetic algorithms

- Mimics the process of natural selection
- Strong similarites to simulated annealing (SA), although parallelized and with the ability to <u>combine</u> good solutions (parents) at each iteration
- Must thus assume there is something to be gained by such a combining
  - How exactly do we combine solutions?
- Computationally heavy

Preprocessing

- Outlier detection
- Missing data
- Features may have different ranges
  - E.g. feature 1 has range $f1_{min}$-$f1_{max}$ while feature n has range fn_{min}-fn_{max}
  - This does seldomly reflect their significance in classification performance!
  - Example: minimum distance classifier uses Euclidean distance
    - Features with large absolute values will dominate the classifier

Feature normalization

- Make all features have similar ranges:
  - Data set with N objects and K features
  - Features x_{ik}, i=1…N, k=1,…K

Zero mean, unit variance:

A simple shift+scaling

$$\bar{x}_k = \frac{1}{N}\sum_{i=i}^{N} x_{ik}$$

$$\sigma_k^2 = \frac{1}{N-1}\sum_{i=i}^{N}(x_{ik} - \bar{x}_k)^2$$

$$\hat{x}_{ik} = \frac{x_{ik} - \bar{x}_k}{\sigma_k}$$

Softmax (non-linear)

Squashes data; often good for heavily-tailed data

$$y = \frac{x_{ik} - \bar{x}_k}{r\sigma_k}$$

$$\hat{x}_{ik} = \frac{1}{1+\exp(-y)}$$

Note: Normalization may change your selected feature subset or the performance of your classifier in general

Summary

- «Curse of dimensionality»
- Objective function
  - Wrapper-based
  - Filter-based
    - Distance measures between pdfs (class-wise densities) / divergence
- Search strategy
  - Exhaustive often not possible
  - Scalar/individual feature selection
  - SFS, SBS, Floating search methods
  - Randomized methods; input shuffling, simulated annealing and genetic algs.