Computational Intelligence: Methods and Applications

Lecture 35

Filters for feature selection and discretization methods.

Włodzisław Duch
SCE, NTU, Singapore
Google: Duch

Filters & Wrappers

Filter approach:

- define your problem, for example assignment of class labels;
- define an index of relevance for each feature $R(X_i)$
- rank features according to their relevance $R(X_{i1}) \geq R(X_{i2}) \geq .. R(X_{id})$
- rank all features with relevance above threshold $R(X_{i1}) \geq t_R$

Wrapper approach:

- select predictor P and performance measure $P(\text{Data})$
- define search scheme: forward, backward or mixed selection
- evaluate starting subset of features $\mathbf{X}_s$, ex. single best or all features
- add/remove feature X_i, accept new set $\mathbf{X}_s \leftarrow \{\mathbf{X}_s + X_i\}$ if
  $P(\text{Datal } \mathbf{X}_s + X_i) > P(\text{Datal } \mathbf{X}_s)$

Filters

Complexity of the forward or backward wrapper approach for *d* features in the worst case is *O*(*dm*), the number of selected features *m* < *d.*

If the evaluation of the error on the training set is costly, or *d* is very large (in some problems it can be 10^4-10^5), then filters are necessary.

Complexity of filters is always O(*d*), and evaluations are less expansive.

Simplest filter for nominal data: majority classifier.
$K=2$ classes, d binary features $X_i=0, 1$, $i=1..d$ and N samples $\mathbf{X}^{(k)}$.
Joint probability $P(\omega_j, X_i)$ is a 2x2 matrix carrying full information.

$$P(\omega, X_j) = \frac{1}{N}\begin{pmatrix} N(\omega_0,0) & N(\omega_0,1) \\ N(\omega_1,0) & N(\omega_1,1)\end{pmatrix} = \begin{pmatrix} P(\omega_0,0) & P(\omega_0,1) \\ P(\omega_1,0) & P(\omega_1,1)\end{pmatrix}$$

Since $P(\omega_0,0)+P(\omega_0,1)=P(\omega_0)$, $P(\omega_1,0)+P(\omega_1,1)=P(\omega_1)=1-P(\omega_0)$, and $P(\omega_0)+P(\omega_1)=1$ there are only 2 free parameters here.

Majority filter

MAP Bayesian classifier makes in the two-class with binary $X_i=0,1$ values, optimal decisions:
IF $P(\omega_0, X_i=0) > P(\omega_1, X_i=0)$ THEN class ω_0

Predicted accuracy: a fraction $P(\omega_0, X_i=0)$ correct, $P(\omega_1, X_i=0)$ errors.
IF $P(\omega_0, X_i=1) > P(\omega_1, X_i=1)$ THEN class ω_0
Predicted accuracy: a fraction $P(\omega_0, X_i=1)$ correct, $P(\omega_1, X_i=1)$ errors.

In general Bayesian “informed majority” classifier *MC* predicts:

$$MC(X_i = x) = \omega_j, \quad j = \arg\max_k P(\omega_k, X_i = x)$$

Accuracy of this classifier using feature X_i requires summing over all values X_i that the feature may take:

$$A(MC, X_i) = \sum_{l=1} \max_k P(\omega_k, X_i = x_l)$$

Example:

$$P(\omega, x) = \begin{pmatrix} 0.1 & 0.4 \\ 0.3 & 0.2 \end{pmatrix}$$

$$A = 0.7$$

MC properties

Since no more information is available two features with the same accuracy $A(MC,X_a) = A(MC,X_b)$ should be ranked as equal.

If for a given value x all samples are from a single class then accuracy of *MC* is 100%, and a single feature is sufficient.

Since optimal decisions are taken at each step is the MAP classifier an optimal solution? For binary features yes, but in other cases:

- Joint probabilities are difficult to estimate, especially for smaller datasets – smoothed probabilities lead to more reliable choices.
- For continuous features results will strongly depend on discretization.
- Information theory weights contributions from each value of X taking into account not only the most probable class, but also distribution of probabilities over other classes, so it may have some advantages, especially for continuous features.

IT filter indices

Mutual information index is frequently used:

$$MI(\boldsymbol{\omega};X_j) = H(\boldsymbol{\omega}) + H(X_j) - H(\boldsymbol{\omega},X_j)$$
$$= \sum_{i=1}^{K}\sum_{k=1}^{M_j} P(\omega_i, X_j = x_k)\,\mathrm{lg}_2 \frac{P(\omega_i, X_j = x_k)}{P(\omega_i)P(X_j = x_k)}$$

To avoid numerical problems with 0/0 for values x_k that are not present in some dataset (ex. in crossvalidation partition) Laplace corrections are used. Their most common form is:

$$P(\omega_i \mid X = x) = \frac{N(\omega_i, X = x) + 1}{N(X = x) + N(\omega_i)}$$ ← number of classes

where the number of all different feature values is $N(X{=}x)$.
Some decision trees evaluate probabilities in this way; instead of 1 and $N(\omega_i)$ other values may be used as long as probabilities sum to 1.

Correlation coefficient

Perhaps the simplest index is based on the Pearson’s correlation coefficient (CC) that calculates expectation values for product of feature values and class values:

$$CC(\boldsymbol{\omega}, X_j) = \frac{E(\omega X_j) - E(\omega)E(X_j)}{\sqrt{\sigma^2(\omega)\sigma^2(X_j)}} \in [-1,+1]$$

For feature values that are linearly dependent correlation coefficient is $+1$ or -1; for complete independence of class and X_j distribution *CC*= 0.

How significant are small correlations? It depends on the number of samples *n*. The answer (see Numerical Recipes www.nr.com) is given by:

$$P(\boldsymbol{\omega} \sim X_j) = \mathrm{erf}\left(\left|\mathrm{CC}(\boldsymbol{\omega}, X_j)\right|\sqrt{n/2}\right)$$

For *n*=1000 even small CC=0.02 gives *P* ~ 0.5,
but for *n*=10 such CC gives only *P* ~ 0.05.

Other relevance indices

Mutual information is based on Kullback-Leibler distance, any distance measure between distributions may also be used, ex. Jeffreys-Matusita

$$D_{JM}(\boldsymbol{\omega}, X_j) = \sum_{i=1}^{K}\sum_{k=1}^{M_j}\left[\sqrt{P(\omega_i, X_j = x_k)} - \sqrt{P(\omega_i)P(X_j = x_k)}\right]^2$$

Bayesian concentration measure is simply:

$$D_{BC}(\boldsymbol{\omega}, X_j) = \sum_{i=1}^{K}\sum_{k=1}^{M_j} P(\omega_i \mid X_j = x_k)^2$$

Many other such dissimilarity measures exist. Which is the best?

In practice they all are similar, although accuracy of calculation of indices is important; relevance indices should be insensitive to noise and unbiased in their treatment of features with many values; IT is fine.

Discretization

All indices of feature relevance require summation over probability distributions. What to do if the feature is continuous?

There are two solutions:

1. Fit some functions to histogram distributions using Parzen windows, ex. fit a sum of several Gaussians, and integrate to get indices, ex:

$$MI\left(\boldsymbol{\omega};X_j\right)=\sum_{i=1}^{K}\int P\left(\omega_i,X_j=x\right)\lg_2\frac{P\left(\omega_i,X_j=x\right)}{P(\omega_i)P\left(X_j=x\right)}dx$$

2. Discretize, put feature values in some bins, and calculate sums.

Histograms with **equal width** of bins (intervals);
Histograms with **equal number of samples** per bin;
Maxdiff histograms: bins starting in the middle $(x_{i+1}-x_i)/2$ of largest gaps
V-optimal: sum of variances in bins should be minimal (good but costly).

Discretized information

With partition of X_j feature values x into r_k bins, joint information is calculated as:

$$H\left(\boldsymbol{\omega},X_j\right)=-\sum_{k=1}^{M_j}\sum_{i=1}^{K}P\left(\omega_i,x\in r_k\right)\lg_2 P\left(\omega_i,x\in r_k\right)$$

and mutual information as:

$$MI\left(\boldsymbol{\omega};X_j\right)=H\left(\boldsymbol{\omega}\right)+H\left(X_j\right)-H\left(\boldsymbol{\omega},X_j\right)=$$

$$-\sum_{i=1}^{K}P(\omega_i)\lg_2 P(\omega_i)-\sum_{k=1}^{M_j}P\left(x\in r_k\right)\lg_2 P\left(x\in r_k\right)-H\left(\boldsymbol{\omega},X_j\right)$$

Tree (entropy-based) discretization

V-opt histograms are good, but difficult to create (dynamic programming techniques should be used).

Simple approach: use decision trees with a single feature, or a small subset of features, to find good splits – this avoids local discretization.

Example:

C4.5 decision tree discretization maximizing information gain,
or SSV tree based separability criterion, vs. constant width bins.

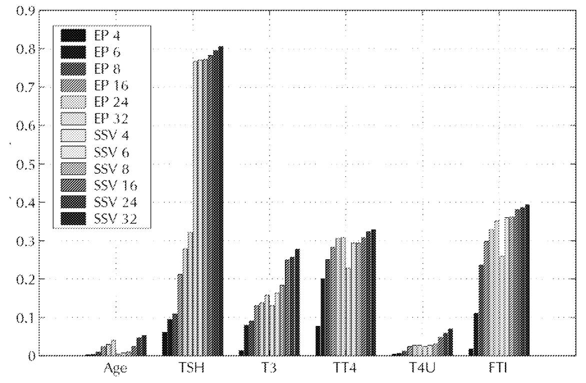
Hypothyroid screening data, 5 continuous features, MI shown.

Compare the amount of mutual information or the correlation between class labels and discretized values of features using equal width discretization and decision tree discretization into 4, 8 .. 32 bins.

Discretization example

MI index for 5 continuous features of the hypothyroid screening data.

EP = equal width partition into 4, 8 .. 32 bins
SSV = decision tree partition (discretization) into 4, 8 .. 32 bins



Feature selection

Selection requires evaluation of mutual information or other indices on subsets of features $S=\{X_{j1},X_{j2},..X_{jl}\}$, with discretization of l-dimensional feature values $\boldsymbol{X}$ into R_k bins:

$$H\left(\boldsymbol{\omega},\left\{X_{j1},..X_{jl}\right\}\right)=-\sum_{k=1}^{M(S)}\sum_{i=1}^{K}P\left(\omega_i,X\in R_k\right)\lg_2 P\left(\omega_i,X\in R_k\right)$$

The difficulty here is reliable estimation of distributions for large number $M(S)$ of l-dimensional partitions. Feedforward pair approximation tries to maximize *MI* of new feature, minimizing sum of *MI* with features in *S*

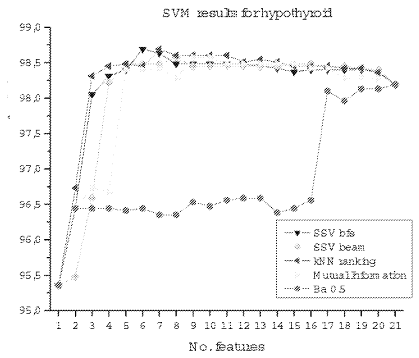
$$\max MI\left(\boldsymbol{\omega};X_i\right),\quad \min\sum_{j\in S}MI\left(X_j;X_i\right),i\neq j$$

Ex. select feature maximizing: $MI\left(\boldsymbol{\omega};X_i\right)-\beta\sum_{j\in S}MI\left(X_j;X_i\right)$

with some $\beta<1$.

Influence on classification

Selecting best 1, 2, ... *k*-dimensional subsets, check how different methods perform using the reduced number of features.



MI using SSC discretization.

SSV bfs/beam – selection of features that are at the top of the SSV decision tree, with best first search, or beam search tree creation method.

kNN ranking – backward wrapper using kNN.

Ba – pairwise approximation fails in this example.

Hypothyroid data, 21 features, 5 continuous.

GM example, wrapper/manual selection

Look at GM 3 wrapper-based feature selection.

Try GM on Australian Credit Data, with 14 features.

1. Standardize the data.
2. Select “transform and classify”, add feature selection wrapper and choose SVM leaving default features.
3. Run it and look at “Feature Ranking”, showing accuracy achieved with each single feature; with default 0.8 cut-off level only one feature is left (F8) and 85.5% accuracy is achieved.
4. Check that lowering level to 0.7 does not improve results.
5. Checked that SVM 10xCV with all features gives ~85% on the test part, and with one feature F8 only result is ~86% for train & test.
6. This is a binary feature, so a simple rule should give the same accuracy; run SSV tree on raw data with optimal pruning.

GM example, Leukemia data

Two types of leukemia, ALL and AML,
7129 genes, 38 train/34 test.

Try SSV – decision trees are usually fast, no preprocessing.

1. Run SSV selecting 3 CVs; train accuracy is 100%, test 91% (3 err) and only 1 feature F4847 is used.
2. Removing it manually and running SSV creates a tree with F2020, 1 err train, 9 err test. This is not very useful ...
3. SSV Stump Field shows ranking of 1-level trees, SSV may also be used for selection, but trees separate training data with F4847
4. Standardize all the data, run SVM, 100% with all 38 as support vectors but test is 59% (14 errors).
5. Use SSV for selection of 10 features, run SVM with these features, train 100%, test 97% (1 error only).

Small samples ... but a profile of 10-30 genes should be sufficient.