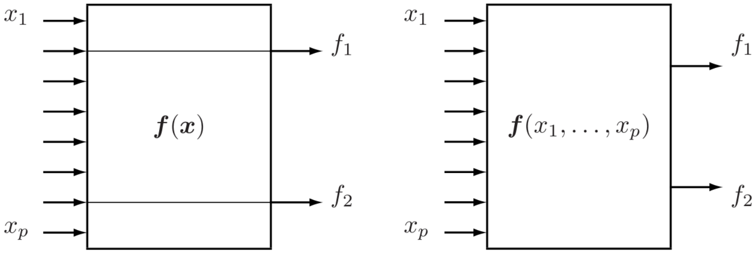
Feature selection

Victor Kitov

Feature selection

Feature selection is a process of selecting a subset of original features with minimum loss of information related to final task (classification, regression, etc.)


x_1
x_p
f_1
f_2
$\boldsymbol{f}(\boldsymbol{x})$
x_1
x_p
f_1
f_2
$\boldsymbol{f}(x_1, \ldots, x_p)$

(a) feature selector (b) feature extractor

Applications of feature selection

- Why feature selection?
  - increase predictive accuracy of classifier
  - improve optimization stability by removing multicollinearity
  - increase computational efficiency
  - reduce cost of future data collection
  - make classifier more interpretable
- Not always necessary step:
  - some methods have implicit feature selection
    - decision trees and tree-based (RF, ERT, boosting)
  - regularization

Types of features

Define f - the feature, $F=\{f_1, f_2, ...f_D\}$ - full set of features, $S=F\backslash\{f\}$.

- **Strongly relevant feature:**

$$p(y|f,S)\neq p(y|S)$$

- **Weakly relevant feature:**

$$p(y|f,S)=p(y|S),\ \text{but}\ \exists S'\subset S:\ p(y|f,S')\neq p(y|S')$$

- **Irrelevant feature:**

$$\forall S'\subset S:\ p(y|f,S')=p(y|S')$$

Aim of feature selection

Find minimal subset $S\subset F$ such that $P(y|S)\approx P(y|F)$, i.e. leave only *relevant* and *non-redundant* features.

Specification

- Need to specify:
    - quality criteria $J(X)$
    - subset generation method $S_1, S_2, S_3, ...$

Types of feature selection algorithms

- Completeness of search:
  - Complete
    - exhaustive search complexity is $2^D - 1$.
  - Suboptimal
    - deterministic
    - random (deterministic with randomness / completely random)
- Integration with predictor
  - independent (filter methods)
  - uses predictor quality (wrapper methods)
  - is embedded inside predictor (embedded methods)

Classifer dependency types

- **filter methods**
    - rely only on general measures of dependency between features and output
    - more universal
    - are computationally efficient

Classifer dependency types

- **filter methods**
  - rely only on general measures of dependency between features and output
  - more universal
  - are computationally efficient
- **wrapper methods**
  - subsets of variables are evaluated with respect to the quality of final classification
  - give better performance than filter methods
  - more computationally demanding

Classifer dependency types

- **filter methods**
  - rely only on general measures of dependency between features and output
  - more universal
  - are computationally efficient
- **wrapper methods**
  - subsets of variables are evaluated with respect to the quality of final classification
  - give better performance than filter methods
  - more computationally demanding
- **embedded methods**
  - feature selection is built into the classifier
  - feature selection and model tuning are done jointly
  - example: classification trees, methods with L_1 regularization.

Table of Contents

1 Filter methods
- Kullback-Leibler divergence & entropy
- Mutual information
- Probability measures
- Context relevant measures
- Cluster measures

2 Feature subsets generation

Correlation

- two class:

$$\rho(f,y)=\frac{\sum_i (f_i-\bar{f})(y_i-\bar{y})}{\left[\sum_i (f_i-\bar{f})^2\sum_i (y_i-\bar{y})^2\right]^{1/2}}=\frac{a}{b}$$

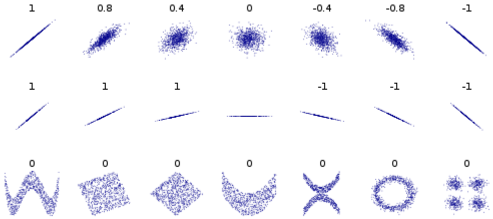
- multiclass $\omega_1,\omega_2,...\omega_C$ (micro averaged $\rho(f,y_c)\, c=1,2,...C.$)

$$R^2=\frac{\sum_{c=1}^{C}\left[\sum_i (f_i-\bar{f})(y_{ic}-\bar{y}_c)\right]^2}{\sum_{c=1}^{C}\sum_i (f_i-\bar{f})^2\sum_i (y_{ic}-\bar{y}_c)^2}=\frac{\sum_c a_c^2}{\sum_c b_c^2}$$

- Benefits:
  - simple to compute
  - applicable both to continuous and discrete features/output.
  - does not require calculation of p.d.f.

Correlation for non-linear relationship

- **Correlation captures only linear relationship.**
- *Example:* $X \sim Uniform[-1,1]$, $Y = X^2$. *X, Y are uncorrelated but dependent.*
- Other examples of data and its correlation:



1. Filter methods
    - Kullback-Leibler divergence & entropy
    - Mutual information
    - Probability measures
    - Context relevant measures
    - Cluster measures

Kullback-Leibler divergence

Kullback-Leibler divergence

For two p.d.f. $P(x)$ and $Q(x)$ Kullback-Leibler divergence $KL(P||Q)$ equals $\sum_x P(x) \ln \frac{P(x)}{Q(x)}$

- Properties:
  - defined only for $P(x)$ and $Q(x)$ such that $Q(x) = 0 \Rightarrow P(x) = 0$
  - $KL(P||Q) \geq 0$
  - $P(x) = Q(x) \forall x <=> KL(P||Q) = 0$ (for discrete r.v.)
  - $KL(P||Q) \neq KL(Q||P)$

Kullback-Leibler divergence

- Symmetrical distance: $KL_{sym}(P||Q) = KL(P||Q) + KL(Q||P)$
- Information theoretic meaning:
    - true data distribution $P(x)$
    - estimated data distribution $Q(x)$

$$KL(P||Q) = -\sum_x P(x) \ln Q(x) + \sum_x P(x) \ln P(x)$$

- $KL(P||Q)$ shows how much longer will be the average length of the code word.

Entropy

- Entropy of random variable Y:

$$H(Y) = -\sum_{y} p(y) \ln p(y)$$

  - level of uncertainty of Y
  - proportional to the average number of bits needed to code the outcome of Y using optimal coding scheme ($-\ln p(y)$ for outcome y).

- Entropy of Y after observing X:

$$H(Y|X) = -\sum_{x} p(x) \sum_{y} p(y|x) \ln p(y|x)$$

1 Filter methods
- Kullback-Leibler divergence & entropy
- Mutual information
- Probability measures
- Context relevant measures
- Cluster measures

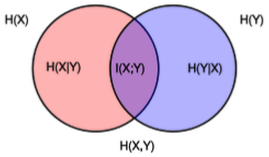
Mutual information

Mutual information measures how much X gives information about Y:

$$MI(X,Y)=\sum_{x,y}p(x,y)\ln\left[\frac{p(x,y)}{p(x)p(y)}\right]=KL\left(p(x,y)||p(x)p(y)\right)$$

Properties:

- $MI(X,Y)=MI(Y,X)$
- $MI(X,Y)=KL(p(x,y)||p(x)p(y))\geq 0$
- $MI(X,Y)=H(Y)-H(Y|X)$
- $MI(X,Y)\leq\min\left\{H(X),H(Y)\right\}$
- X,Y- independent <=> $MI(X,Y)=0$ (for discrete r.v.)
- X completely identifies Y, then $MI(X,Y)=H(Y)\leq H(X)$

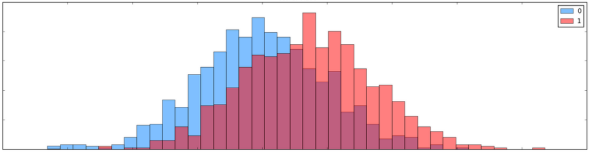


Mutual information for feature selection

- Normalized variant $NMI(X,Y)=\frac{MI(X,Y)}{H(Y)}$ equals
    - zero, when $P(Y|X)=P(Y)$
    - one, when X completely identifies Y.
- Properties of MI and NMI:
    - identifies arbitrary non-linear dependencies
    - requires calculation of probability distributions
    - continuous variables need to be discretized

1 Filter methods
- Kullback-Leibler divergence & entropy
- Mutual information
- Probability measures
- Context relevant measures
- Cluster measures

Relevance based on probabilistic distance



Measure of feature f relevance - distance between $p(f|\omega_1)$ and $p(f|\omega_2)$

Distances between probability density functions

Let $f(x) = p(f|\omega_i)$ and $g(x) = p(f|\omega_j)$.

- Total variation: $\frac{1}{2}\int |f(x) - g(x)| dx$,
- Euclidean: $\frac{1}{2}\left(\int (f(x) - g(x))^2 dx\right)^{1/2}$
- Hellinger: $\left(\frac{1}{2}\int \left(\sqrt{f(x)} - \sqrt{g(x)}\right)^2 dx\right)^{1/2}$
- Symmentrical KL: $\int (f(x) - g(x)) \ln \frac{f(x)}{g(x)} dx$

Distances between cumulative probability functions

Let $F(x)=P(f\le x|\omega_i)$ and $G(x)=P(f\le x|\omega_j)$:

- Kolmogorov: $\sup_x|F(x)-G(x)|$
- Kantorovich: $\int|F(x)-G(x)|dx$
- L_p: $\left(\int|F(x)-G(x)|^p dx\right)^{1/p}$

Other

Multiclass extensions:

Suppose, we have a two-class distance score $J(\omega_i, \omega_j)$.
We can extend it to multiclass case using:

$$J = \max_{\omega_i, \omega_j} J(\omega_i, \omega_j)$$

$$J = \sum_{i<j} p(\omega_i) p(\omega_j) J(\omega_i, \omega_j)$$

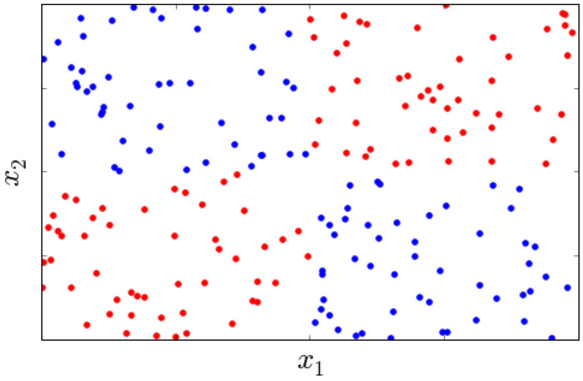
Presented criteria compare probabilities given 2 different classes. We may also compare class-unconditional feature distribution with class-conditional feature distribution.

1. Filter methods
   - Kullback-Leibler divergence & entropy
   - Mutual information
   - Probability measures
   - Context relevant measures
   - Cluster measures

Relevance in context

Individually features may not predict the class, but may be relevant together:

$$p(y|x_1) = p(y),\ p(y|x_2) = p(y),\ \text{but}\ p(y|x_1, x_2) \neq p(y)$$



Relief criterion

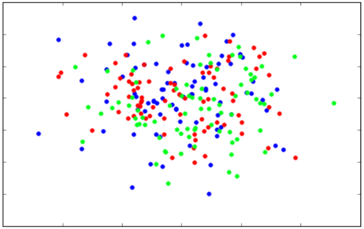
```
INPUT:
    Training set $(x_1, y_1), (x_2, y_2), ...(x_N, y_N)$
    Number of neighbours $K$
    Distance metric $d(x, x')$ # usually Euclidean

for each pattern $x_n$ in $x_1, x_2, ...x_N$:
    calculate $K$ nearest neighbours of the same class $y_n$:
        $x_{s(n,1)}, x_{s(n,2)}, ...x_{s(n,K)}$
    calculate $K$ nearest neighbours of class different from $y_n$:
        $x_{d(n,1)}, x_{d(n,2)}, ...x_{d(n,K)}$

for each feature $f_i$ in $f_1, f_2, ...f_D$:
    calculate relevance $R(f_i) = \sum_{n=1}^{N} \sum_{k=1}^{K} \frac{|x_n^i - x_{d(n,k)}^i|}{|x_n^i - x_{s(n,k)}^i|}$

OUTPUT:
    feature relevances $R$
```

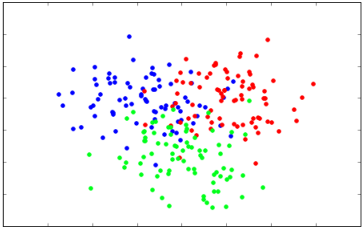
1 Filter methods
- Kullback-Leibler divergence & entropy
- Mutual information
- Probability measures
- Context relevant measures
- Cluster measures

Cluster measures

General idea of cluster measures

Feature subset is good if observations belonging to different classes group into different clusters.





Cluster measures

Define:

- $z_{ic} = \mathbb{I}[y_i = \omega_c]$, N-number of samples, N_i-number of samples belonging to class ω_i.
- $m = \frac{1}{N}\sum_i x_i$, $m_c = \frac{1}{N_c}\sum_i z_{ic} x_i$, $j = 1, 2, ...C$.
- Global covariance: $\Sigma = \frac{1}{N}\sum_i (x - m)(x - m)^T$,
- Intraclass covariances: $\Sigma_c = \frac{1}{N_c}\sum_i z_{ic}(x_i - m_c)(x_i - m_c)^T$
- Within class covariance: $S_W = \sum_{c=1}^{C} \frac{N_c}{N}\Sigma_c$
- Between class covariance: $S_B = \sum_{c=1}^{C} \frac{N_c}{N}(m_c - m)(m_c - m)^T$

Interpretation

Within class covariance shows how samples are scattered within classes.
Between class covariance shows how classes are scattered between each other.

Scatter magnitude

Theorem 1

Every real symmetric matrix $A \in \mathbb{R}^{nxn}$ can be factorized as

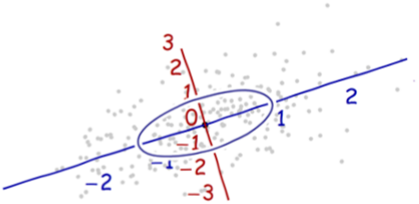
$$A = U \Sigma U^T$$

where Σ is diagonal and U is orthogonal. $\Sigma = \text{diag}\{\lambda_1, \lambda_2, ...\lambda_n\}$ and $U = [u_1, u_2, ...u_n]$ where $\lambda_i, i = 1, 2, ...n$ are eigenvalues and $u_i \in \mathbb{R}^{nx1}$ are corresponding eigenvectors.

- U^T is basis transform corresponding to rotation, so only Σ reflects scatter.

Measuring scatter of symmetric matrix

Scaling in basis U



- Aggregate measures of scatter $\operatorname{tr}\Sigma = \sum_i \lambda_i$ and $\det\Sigma = \prod_i \lambda_i$
- Since $\operatorname{tr}\left[P^{-1}BP\right] = \operatorname{tr} B$ and $\det\left[P^{-1}BP\right] = \det B$, we can estimate scatter with $\operatorname{tr} A = \operatorname{tr}\Sigma$ and $\det A = \det\Sigma$

Clusterization quality

- Good clustering: S_W is small and S_B, Σ are big.
- Cluster discriminability metrics:

$$Tr\{S_W^{-1}S_B\}, \frac{Tr\{S_B\}}{Tr\{S_W\}}, \frac{\det \Sigma}{\det S_W}$$

Resume

- Pairwise feature measures
  - fail to estimate relevance in context of other features
  - are robust to curse of dimensionality
- Context aware measures:
  - estimate relevance in context of other features
  - prone to curse of dimensionality if distances are calculated (such as Relief criterion)

Table of Contents

1 Filter methods

2 Feature subsets generation
- Deterministic feature selection
- Randomised feature selection

2. Feature subsets generation
    - Deterministic feature selection
    - Randomised feature selection

Incomplete search with suboptimal solution

- Consider not all but only the most promising feature subsets.
- Order features with respect to $J(f)$:

$$J(f_1) \geq J(f_2) \geq ... \geq J(f_D)$$

  - select top m
  $$\hat{F} = \{f_1, f_2, ...f_m\}$$
  - select best set from nested subsets: $S = \{\{f_1\}, \{f_1, f_2\}, ...\{f_1, f_2, ...f_D\}\}$
  $$\hat{F} = \arg\max_{F \in S} J(F)$$

- Comments:
  - simple to implement
  - if $J(f)$ is context unaware, so will be the features
  - example: when features are correlated, it will take many redundant features

Sequential search

- Sequential forward selection algorithm:

```
init: $k = 0$, $F_0 = \emptyset$
while $k < max\_features$:
    $f_{k+1} = \arg\max_{f \in F} J(F_k \cup \{f\})$
    $F_{k+1} = F_k \cup \{f_{k+1}\}$
    if $J(F_{k+1}) < J(F_k)$: break
    k=k+1
return $F_k$
```

- Variants:
    - sequential backward selection
    - up-k forward search
    - down-p backward search
    - up-k down-p composite search
    - up-k down-(variable step size) composite search

2. Feature subsets generation
   - Deterministic feature selection
   - Randomised feature selection

Genetic algorithms

- Analogy to genetic inheritance in biology.
- Each feature set $F=\{f_{i(1)},f_{i(2)},...f_{i(K)}\}$ is represented using binary vector $[b_1,b_2,...b_D]$ where $b_i=\mathbb{I}[f_i\in F]$
- Genetic operations:
  - $\textit{crossover}(b^1,b^2)=b$, where $b_i=\begin{cases}b_i^1 & \text{with probability } \frac{1}{2}\\ b_i^2 & \text{otherwise}\end{cases}$
  - $\textit{mutation}(b^1)=b$, where $b_i=\begin{cases}b_i^1 & \text{with probability } 1-\alpha\\ \neg b_i^1 & \text{with probability } \alpha\end{cases}$

    for some small α.

Genetic algorithms

```
INPUT:
    size of population $B$
    size of expanded population $B'$
    parameters of mutation $\theta$ (and possibly crossover)
    maximum number of iterations $T$, minimum quality change $\Delta J$

ALGORITHM:
generate $B$ feature sets $S_1, S_2, ...S_B$ randomly.
set $t = 1$, $P^0 = \{S_1, S_2, ...S_B\}$, $J^0 = J(P^0)$
while $t <= T$ and $|J^t - J^{t-1}| > \Delta J$:
    modify $P^{t-1}$ using crossover and mutation:
        $S_1', S_2', ...S_{B'}' = \text{modify}(P^{t-1}|\theta)$
    order transformed sets by decreasing quality:
        $J(S'^t_{i(1)}) \geq J(S'^t_{i(1)}) \geq ...J(S'^t_{i(B')})$
    set next population to consist of best representatives:
        $P^t = \{S'_{i(1)}, S'_{i(2)}, ...S'_{i(B)}\}$
    $J^t = J^t(P^t)$
    $t = t + 1$

OUTPUT: suboptimal set of feature sets $P^t$
```

Modifications of genetic algorithm

- Preserve best features and best feature subsets:
    - Augment P'^t with K best representatives from P^{t-1}.
    - Make mutation probability lower for good features (that frequently appear in inside representatives).
- Increase breadth of search:
    - Crossover between more than two parents
- To prevent convergence to local optimum:
    - simultaneously modify several populations and allow rare random transitions between them.

Extra

- Tree feature importances (*clf.feature_importances_* in sklearn).
  - Consider feature f
  - Let $T(f)$ be the set of all nodes, relying on feature f when making split.
  - efficiency of split at node$t$: $\Delta I(t) = I(t) - \sum_{c \in childen(t)} \frac{n_c}{n_t} I(c)$
  - feature importance of f: $\sum_{t \in T(f)} n_t \Delta I(t)$
- Feature importances from linear classification:
  1. fit linear classifier with regularization to data
  2. retrieve w (*clf.coef_* in scikit-learn)
  3. importance of feature f_i is equal to $|w_i|$.
- We can reweight features for methods, relying on scaling by feature importances (such as K-NN).