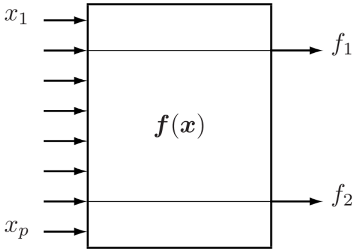
Feature selection

Victor Kitov

v.v.kitov@yandex.ru

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.)



(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:

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)
        - L1 regularization

Types of features

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

- **Strongly relevant feature:**

$$p(y|f,\tilde{F})\neq p(y|\tilde{F})$$

- **Weakly relevant feature:**

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

- **Irrelevant feature:**

$$\forall S\subset\tilde{F}:\ p(y|f,S)=p(y|S)$$

Types of features

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

- **Strongly relevant feature:**

$$p(y|f,\tilde{F})\neq p(y|\tilde{F})$$

- **Weakly relevant feature:**

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

- **Irrelevant feature:**

$$\forall S\subset\tilde{F}:\ p(y|f,S)=p(y|S)$$

Aim of feature selection

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

Categorization of feature selection algorithms

- Completeness of search:
  - Complete
    - exhaustive search complexity is 2^D.
    - may be not exhaustive under certain conditions on $J(S)$[1]
  - Suboptimal
    - deterministic
    - random (deterministic with randomness / completely random)
- Integration with final predictor
  - independent (filter methods)
  - uses predictor quality (wrapper methods)
  - is embedded inside predictor (embedded methods)

[1] $J(S)$ is a score of feature subset S.

Table of Contents

Individual feature importances approach

- Estimate importances for individual features $I(f_1), I(f_2), ...I(f_D)$.
- Generate feature subset based on importances.

1. Individual feature importances approach
   - Feature subset generation
   - Feature importance estimation

Incomplete search with suboptimal solution

- Order features with respect to feature importances $I(f)$:

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

option 1: select top m

$$\hat{F} = \{f_1, f_2, ...f_m\}$$

option 2: 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
  - when features are correlated, it will take many redundant features

1. Individual feature importances approach
    - Feature subset generation
    - Feature importance estimation

Application of feature importances

- Feature importances can be used:
  - for feature selection
  - for rescaling features for adapting their impact on the model:
    - e.g.: in K-NN, in linear methods with regularization
  - for adapting feature sampling probability in random forest, extra random trees.

Correlation

- two class:

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

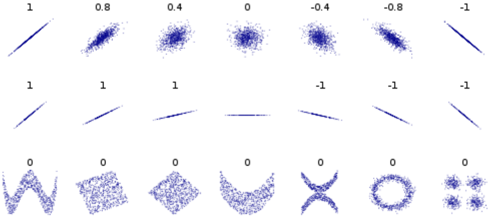
- multiclass $\omega_1,\omega_2,...\omega_C$ (micro averaged $\rho(f,y_c)\; c=1,2,...C$)

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

- Benefits:
    - simple to compute
    - applicable both to continuous and discrete features/output.
    - does not require calculation of probability density function.

Correlation for non-linear relationship

- **Correlation captures only linear relationship**.
- *Example: consider X-random variable, with $\mathbb{E}X = 0$, $\mathbb{E}X^3 = 0$ and random variable $Z = X^2$. Then X, Z are uncorrelated but dependent.*
- Other examples of data and its correlation:



- May consider correlation between ranks.

Defintitions

- Entropy[2] of random variable Y:

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

- Kullback-Leibler divergence for two p.d.f. $P(x)$ and $Q(x)$: $KL(P||Q) := \sum_x P(x) \ln \frac{P(x)}{Q(x)}$
- Mutual information:

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

[2] measures level of uncertainty of r.v. Y

Properties of MI

- Properties of *MI*:
    - identifies arbitrary non-linear dependencies
    - requires calculation of probability distributions
    - continuous variables need to be discretized

Table of Contents

Tree feature importances

- 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: $\sum_{t \in T(f)} n_t \Delta I(t)$
- Alternative: difference in decision tree prediction quality for
  1. original validation set
  2. validation set with j-th feature randomly shuffled

Feature importances from linear model

- 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 $|w_i|$.
- Features should be normalized beforehand!

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
    - may consider random subset of variants