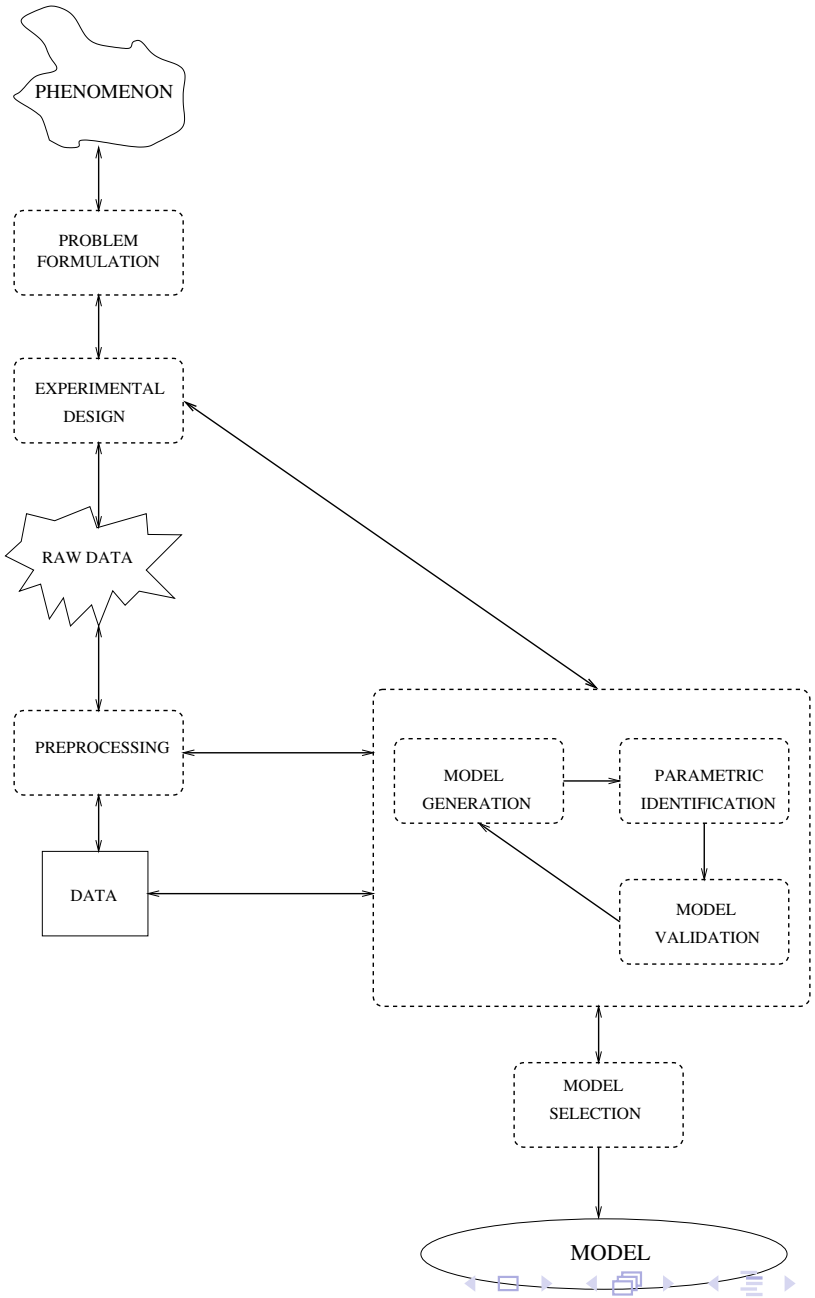
INFO-F-422: Statistical foundations of machine learning

Feature selection

Gianluca Bontempi

Machine Learning Group
Computer Science Department
mlg.ulb.ac.be

The learning procedure



Feature selection problem

- ML algorithms degrade in accuracy when faced with many inputs (aka *features*) that are not necessary.
- Tasks with thousands of features: bioinformatics classification where n (number of genes for which expression is measured) may range from 6000 to 60000.
- Original learning techniques not designed to cope with lot of irrelevant features.
- Feature selection: selecting some subset of features to use as inputs.
- Using all features may negatively affect generalization, because of irrelevant/ redundant features.
- Feature selection as a **model selection** problem.

Benefits and drawbacks of feature selection

There are many potential benefits of feature selection:

- facilitating data visualization and data understanding,
- reducing the measurement and storage requirements,
- reducing training and utilization times of the final model,
- defying the curse of dimensionality to improve prediction performance.

Drawbacks are

- additional layer of complexity: the search in the hypothesis space is augmented by another dimension, finding the optimal subset of relevant features.
- additional time for learning.

Neighbourhood and dimension n

- n dimensional space
- query point $x_q \in \mathbb{R}^n$
- unit volume around x_q.
- hypercube of edge length $d < 1$ which contains a portion V of the unit volume

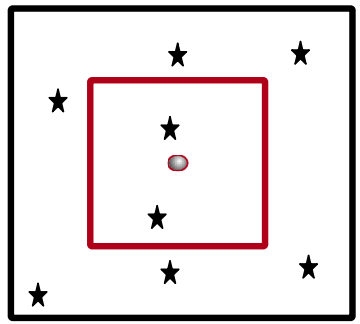
$$d^n = V \Rightarrow d = V^{1/n}$$

- $$V = \{x \in \mathbb{R}^n : |x_j| < d/2, \quad j = 1, \ldots, n\}$$

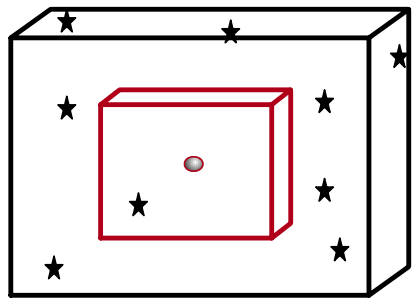
  where d stands for a measure of **locality**.
- If I want to cover half of the unit volume, how local should I be?
- For instance if $V = 0.5$ we have $d = 0.7, 0.87, 0.98$ for $n = 2, 5, 50$.

n=1 d=1/2 V=1/2 N_d=4

n= 2 d=1/2 V=1/4 N_d=2



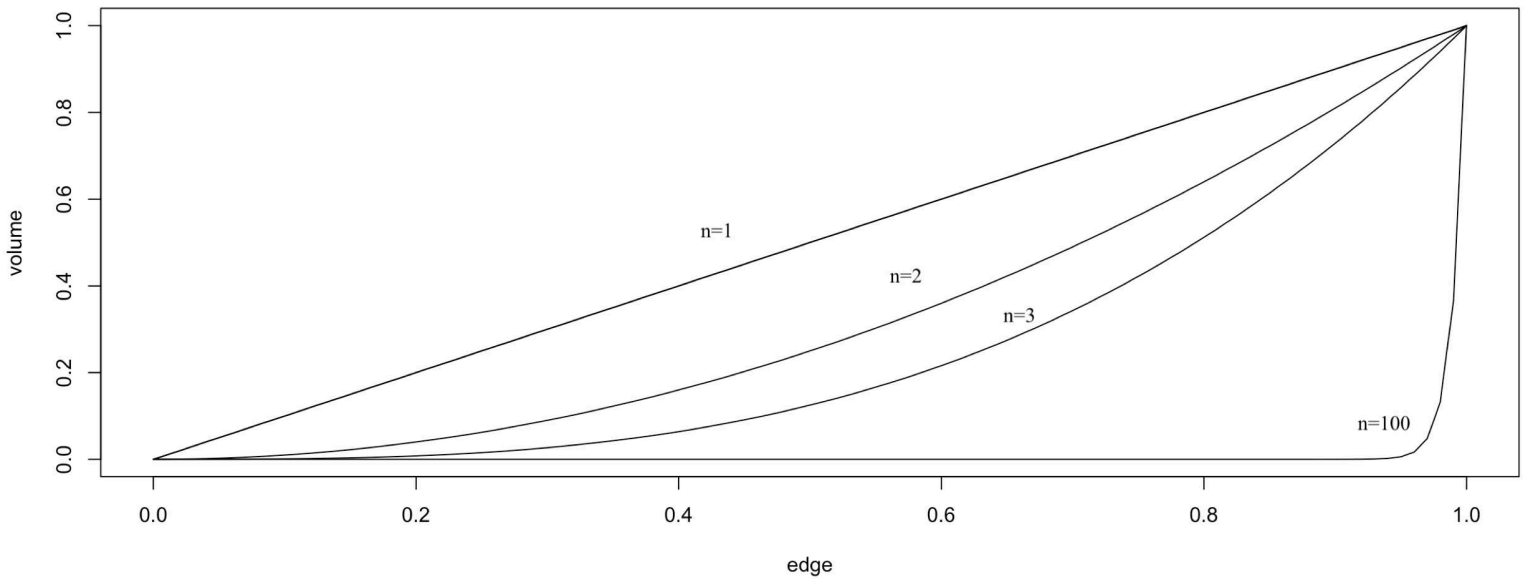
n=3 d=1/2 V=1/8 N_d=1



Keeping the same degree of locality, as n increase:

- the volume of the cubic neighbourhood decreases
- I cover less and less of the unit volume
- I will find less and less points

Neighborhood volume vs. edge size



- given neighbourhood volume V: edge length increases for increasing n.
- given edge d: neighbourhood volume decreases for increasing n.

Curse of dimensionality issues

- As n increases the amount of local data goes to zero: all data sets are sparse.
- Given two supervised learning tasks with different input dimension n_1 and n_2, if we want to preserve the same degree of locality we $N_1 >> N_2$ if $n_1 > n_2$.
- In high dimensions, all datasets show multicollinearity.
- In high dimensions, the number of possible models to consider increases superexponentially in n.

Methods of feature selection

1. **Filter methods:** preprocessing methods that assess features ignoring the role of learning algorithm. Examples: ranking, compression techniques (like PCA or clustering).
2. **Wrapper methods:** assess subsets of variables according to usefulness to a given predictor. Search using the learning algorithm as part of the evaluation function. Examples: stepwise methods in linear regression.
3. **Embedded methods:** selection as part of learning procedure. Examples: classification trees, random forests, methods based on regularization (e.g. lasso) and representation learning (e.g. autoencoder).

Pros-cons analysis

- **Filter methods:**
  - Pros: easily scale to very high-dimensional datasets, computationally simple and fast, and independent of the classification algorithm. FS performed only once and different classifiers are evaluated.
  - Cons: ignore interaction with the classifier, often univariate or low-variate. Each feature is considered separately, ignoring feature dependencies.

Pros-cons analysis

- **Wrapper methods:**
  - Pros: take into account learning algo and feature dependencies.
  - Cons: higher risk of overfitting than filters and very computationally intensive.
- **Embedded methods:**
  - Pros: less computationally intensive than wrapper methods.
  - Cons: specific to a learning machine.

Principal component analysis (PCA)

- One of the most popular methods for linear dimensionality reduction.
- It projects the data from the original space into a a lower dimensional space in an unsupervised manner.
- New axes (called principal components (PC)) are created as linear combinations of original ones.

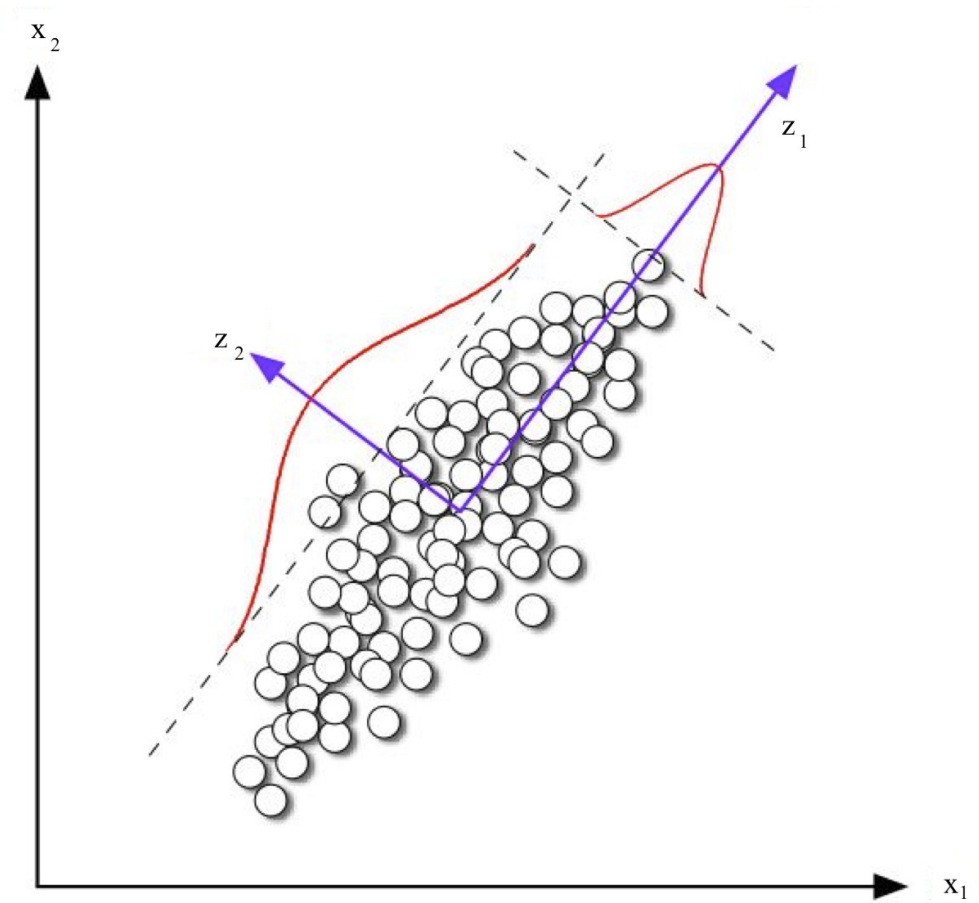
Principal component analysis

- PC1 is the axis along which there is the greatest variation of the projected observations.
- PC1 is $a = [a_1, \ldots, a_n] \in \mathbb{R}^n$ such that

$$\mathbf{z} = a_1 \mathbf{x}_{\cdot 1} + \cdots + a_n \mathbf{x}_{\cdot n} = a^T \mathbf{x}$$

  has the largest variance.
- Optimal a is the eigenvector of $\mathrm{Var}[\mathbf{x}]$ corresponding to the largest eigenvalue λ_1.
- PC2 is the axis orthogonal to PC1 that has the greatest variation of the projected observations; and so forth.

PCA example ($n = 2$)



PCA: the algorithm

Input: matrix $X_{[N,n]}$ of size $[N, n]$:

1. X is normalized into $\tilde{X}$ such that each column $\tilde{X}[, i]$, $i = 1, \ldots, n$, has mean 0 and variance 1.
2. Singular Value Decomposition (SVD) of $\tilde{X}$

$$\tilde{X} = UDV^T$$

   where $U_{[N,N]}$ is orthogonal, $D_{[N,n]}$ is rectangular diagonal with diagonal elements $d_1 \geq d_2 \geq \cdots \geq d_n$ and $V_{[n,n]}$ is orthogonal.
3. $\tilde{X}$ transformed into $Z = \tilde{X}V = UD$ where each column of $Z_{[N,n]}$ is a linear combination of original features and its importance is diminishing.
4. The first $h < n$ columns of Z (aka eigen-features) are chosen to represent the dataset.

How many PCs?

1. Fix a threshold on the proportion of variance to be explained by the PCs, e.g. choose h PCs such that

$$\frac{\lambda_1 + \cdots + \lambda_h}{\sum_{j=1}^{n} \lambda_j} \geq \alpha$$

   where $\lambda_j = \frac{d_j^2}{N-1}$ is the jth largest eigenvalue and is equal to the variance of the jth component.
2. Scree plot: decreasing plot of λ_j as a function of j. Choose the value of h corresponding to a knee in the curve
3. Selection by cross validation.

Python code

```
## LEAVE-ONE-OUT loop
for i in range(N):
    # Create the training set by leaving out the i-th observation
    Xtr = np.delete(X, i, axis=0)
    Ytr = np.delete(Y, i, axis=0)
    # Normalize training inputs (column-wise standardization using sample std)
    Xtr_mean = np.mean(Xtr, axis=0)
    Xtr_std = np.std(Xtr, axis=0, ddof=1)
    Xtr_scaled = (Xtr - Xtr_mean) / Xtr_std
    # normalization of the input test
    Xts = (X[i, :] - Xtr_mean) / Xtr_std
    # SVD on scaled training data divided by sqrt(N-1)
    U, d, Vt = np.linalg.svd(Xtr_scaled / sqrt(N-1), full_matrices=False)
    V = Vt.T  # In Python, np.linalg.svd returns Vt; columns of V are the right singular vectors.
    ## PC loop
    for h in range(1, n + 1):
        # Vh = first h columns of V
        Vh = V[:, :h]
        # Compute the principal component scores for training data
        Zh = np.dot(Xtr_scaled, Vh)
        # Compute the principal component scores for the test input
        Zi = np.dot(Xts, Vh)
        # knn prediction using PCA transformed data
        YhatPCAi = predKNN( Zh, Ytr, Zi)
        # knn prediction using the first h columns of the original scaled data
        Yhati = predKNN( Xtr_scaled[:, :h], Ytr, Xts[:h])
        EPCA[i, h - 1] = (Y[i] - YhatPCAi) ** 2
        E[i, h - 1] = (Y[i] - Yhati) ** 2
# Calculate mean squared error for each number of PCs and print the results
mse_loo_PCA = np.mean(EPCA, axis=0)
best_pc_number = np.argmin(mse_loo_PCA) + 1  # +1 for 1-indexed result
```

See also script `FeatureSel/fs_pca.py`

Clustering (unsupervised learning)

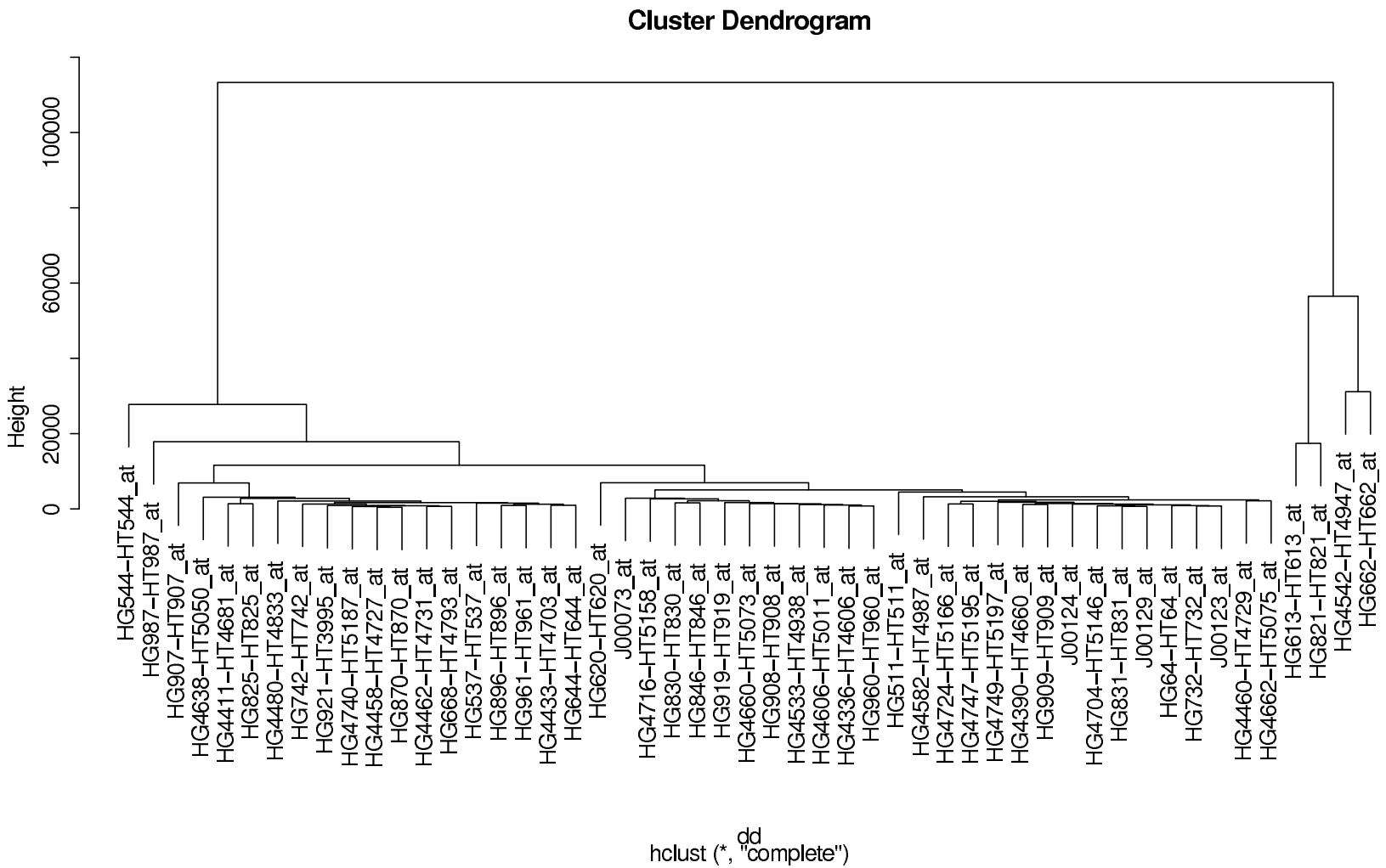
- Returns groups of features or observations with similar patterns (e.g. patterns of gene expressions in microarray data).
- Require a distance function between variables and between clusters.
- **Nearest neighbor clustering:** number of clusters fixed in advance, then each variable is assigned to each cluster. Examples: Self Organizing Maps (SOM) and K-means.
- **Agglomerative clustering:** bottom-up methods where clusters start as empty and variables are successively added. An example is hierarchical clustering

Hierarchical clustering

- It begins by considering all the observations as separate clusters and starts by putting together the two samples that are nearest to each other.
- In subsequent stages also clusters can be merged.
- A measure of dissimilarity between sets of observations is required.
- It is based on an appropriate metric (a measure of distance between pairs of observations), and a linkage criterion which specifies the dissimilarity of sets as a function of the pairwise distances of observations in the sets.

Dendrogram

The output of hierarchical clustering is a *dendrogram*, a tree diagram used to illustrate the arrangement of the clusters.



Ranking

- Assesses the importance (or relevance) of each variable with respect to the output by using a univariate measure. Supervised techniques of complexity $O(n)$.
- Measures of relevance:
  - Pearson correlation (the greater the more relevant) which assumes linearity;
  - in case of binary classification significance p-value of an hypothesis test (the lower the more relevant) which aims at detecting the features that split well the dataset. Parametric (t-test) and nonparametric (Wilcoxon) tests have been proposed in litterature;
  - mutual information (the greater the more relevant).
- After the univariate assessment the method ranks the variables in a decreasing order of relevance.

Ranking methods (II)

- (+): fast (complexity $O(n)$), intuitive output and easy to understand.
- (-) they disregard redundancies and higher order interactions between variables (e.g. genes).
- (-) best k individual features are not necessarily constitute the best k variate vector.

Wrapping search

- Search in space $W = \{0,1\}^n$ where $w \in W$ is such that

$$w[j] = \begin{cases} 0 & \text{if the input } j \text{ does NOT belong to the set of features} \\ 1 & \text{if the input } j \text{ belongs to the set of features} \end{cases}$$

- We look for optimal $w^* \in \{0,1\}^n$ such that

$$w^* = \arg \min_{w \in W} \text{MISE}_w$$

where MISE_w is the generalization error of the model based on w.

- the number of vectors in W is equal to 2^n.
- for moderately large n, the exhaustive search is no more possible.

Wrapping search strategies

Greedy methods evaluate $\sum_{i=0}^{n-1}(n-i) = \frac{n(n+1)}{2}$ $(O(n^2))$ subsets by either adding or deleting one variable at a time.

1. **Forward selection:** starts with no variables and incorporates features. First input is the one with the lowest generalization error: the second is the one that, together with the first, has the lowest error, and so on. R script `FeatureSel/fs_wrap.R`.
2. **Backward selection:** opposite direction of forward approach by progressively removing feature from the full set. At each step, input to be removed is the one whose absence causes the lowest increase (or highest decrease) of generalisation error.
3. **Stepwise selection:** it combines the previous two techniques, by testing for each set of variables, first the removal of features belonging to the set, then the addition of variables not in the set.

Wrapper limitation

- Wrapper techniques search for the best subset of features by performing a large number of comparisons and selections.
- Despite the use of validation procedures, it is highly possible that high correlations or low prediction errors are found only due to chance.
- A bad practice consists in using the same set of observations to select the feature set and to assess the generalization accuracy of the classifier (see external cross-validation).

External cross-validation

Internal cross-validation: assessment process where the entire available data is used to select a feature set.
This is a wrong procedure since it can induce sampling bias (i.e. too optimistic estimation of the generalization error due to testing on samples already considered in the feature selection process).
The *external cross-validation* consists into

- Leave out a single test fold of the data set
- Use the remaining training folds for selecting **both variables and model**
- Evaluate the generalization error (of both model and feature set) on the test fold.

Assessment by permutation

- If no additional data are available, an alternative consists in estimating how often random data would generate correlation or classification accuracy of the magnitude obtained in the original analysis.
- permutation testing: repeating the procedure several times with data where the dependency between the input and the output is artificially removed (for exemple by permuting the inputs ordering). This would provide us with a distribution of the accuracy in case of random data where no information is present in the data.

Combining instead of selecting

- Instead of choosing one particular FS method, different FS methods can be combined using ensemble approaches.
- Since there is not an optimal feature selection technique and due to the possible existence of more than one subset of features that discriminates the data equally well, model combination approaches have been adapted to improve the robustness and stability of final, discriminative methods.
- ensemble techniques include averaging over multiple feature subsets.
- use of ensemble approaches requires additional computational resources.

Shrinkage methods

- Improve an estimator by adding constraints to reduce variance.
- **Ridge regression** is a shrinkage method applied to least squares regression

$$\hat{\beta}_r = \arg\min_b \{\sum_{i=1}^{N}(y_i - x_i^T b)^2 + \lambda \sum_{j=1}^{p} b_j^2\} =$$

$$= \arg\min_b \left((Y - Xb)^T (Y - Xb) + \lambda b^T b\right)$$

where $\lambda > 0$ is a complexity parameter that controls the amount of shrinkage: the larger the value of λ the stronger the constraint.

Ridge regression

- An equivalent way to write the ridge problem is

$$\hat{\beta}_r = \arg\min_b \sum_{i=1}^{N} (y_i - x_i^T b)^2,$$

$$\text{subject to } \sum_{j=1}^{p} b_j^2 \leq L$$

  where there is a one-to-one correspondence between the parameter λ and L

- It can be shown that the ridge regression solution is

$$\hat{\beta}_r = (X^T X + \lambda I_p)^{-1} X^T Y$$

  where I_p is a $[p, p]$ identity matrix.

Python code

```
LAM = np.arange(20,50,1)
E = np.empty((N, len(LAM)))
E[:] = np.nan

## LEAVE-ONE-OUT loop
for i in range(N):
    Xtr = np.column_stack((np.ones(N - 1), np.delete(X, i, axis=0)))
    Ytr = np.delete(Y, i)  # Y[-i]

    # Xts: c(1, X[i,])
    Xts = np.concatenate(([1], X[i,]))

    cnt = 0
    for lam in LAM:
        A = Xtr.T @ Xtr + lam * np.eye(n + 1)
        b = Xtr.T@ Ytr
        betahat = np.linalg.inv(A) @ b
        Yhati = Xts@ betahat
        E[i, cnt] = (Y[i] - Yhati) ** 2
        cnt += 1

mseloo = np.mean(E, axis=0)
lambest = LAM[np.argmin(np.mean(E, axis=0))]
print("best lambda=",lambest)
```

See also script `FeatureSel/fs_ridge.py`

Lasso

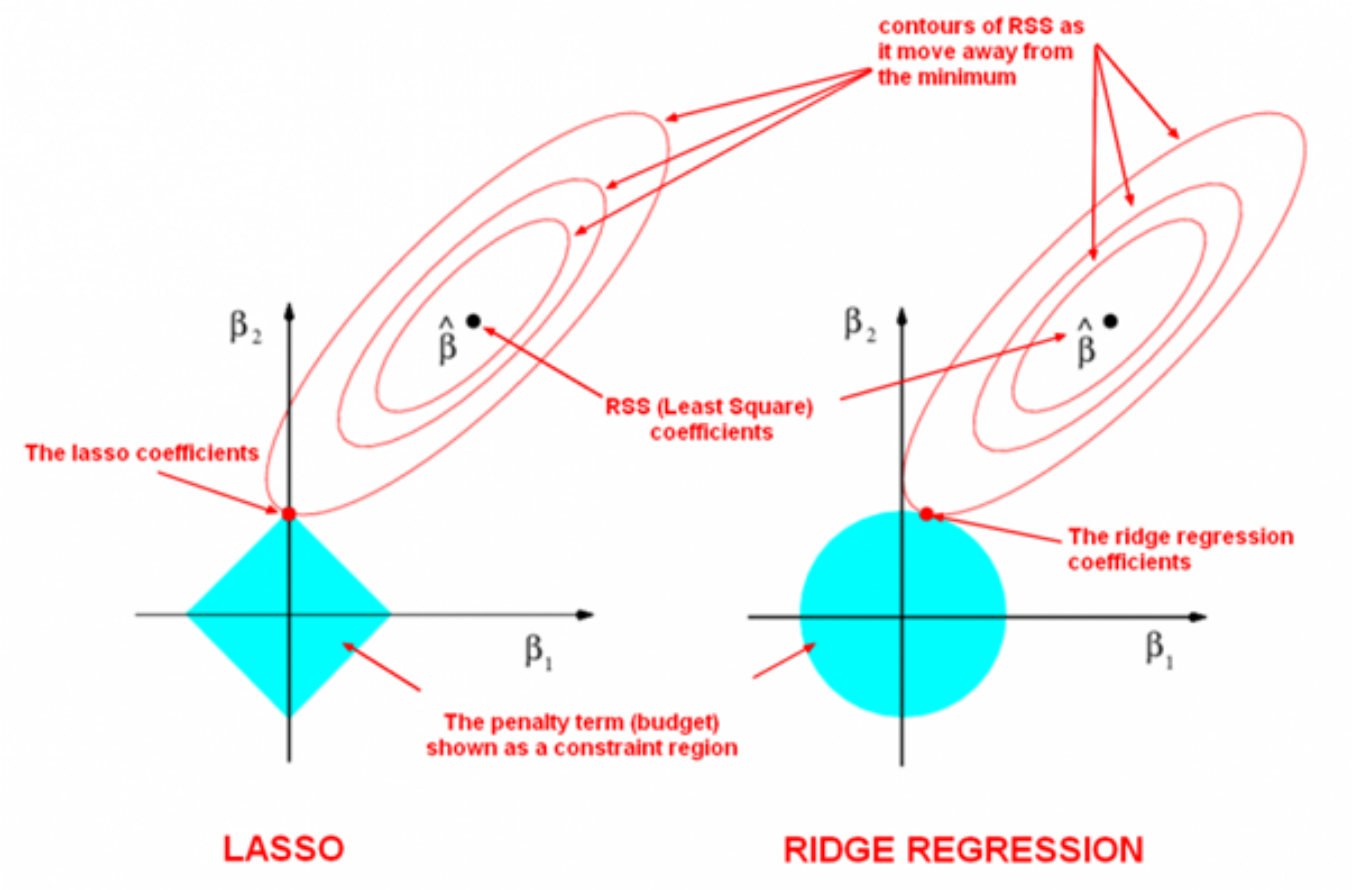
- Estimate of the linear parameters is returned by

$$\hat{\beta}_r = \arg\min_b \sum_{i=1}^{N} (y_i - x_i^T b)^2,$$

$$\text{subject to } \sum_{j=1}^{p} |b_j| \leq L$$

- The 1-norm penalty of the lasso approach makes the solution nonlinear and requires a quadratic programming algorithm.
- Note that if $L > \sum_{j=1}^{p} |\hat{\beta}_j|$ the lasso returns the common least-squares solution.
- Lasso returns sparser solutions (more coefficients forced to zero)
- How to choose the penalty factor L? In practice we have recours to cross-validation strategies.

Ridge regression vs. lasso



From `www.quora.com`

Notions of entropy (continuous case)

Consider a continuous r.v. $\mathbf{y}$. The *(differential) entropy* of $\mathbf{y}$ is defined by

$$H(\mathbf{y}) = -\int \log(p(y))p(y)dy = E_{\mathbf{y}}[-\log(p(y))]$$

knowing that $0 \log 0 = 0$.

- Entropy if a functional of the distribution of $\mathbf{y}$.
- Entropy is a measure of the predictability of a r.v. $\mathbf{y}$. The higher the entropy, the less reliable are our predictions about $\mathbf{y}$.
- for a scalar normal r.v. $\mathbf{y} \sim \mathcal{N}(0, \sigma^2)$

$$H(\mathbf{y}) = \frac{1}{2}\left(1 + \ln 2\pi\sigma^2\right) = \frac{1}{2}\left(\ln 2\pi e\sigma^2\right)$$

Conditional entropy

Consider two continuous r.v.s $\mathbf{x}$ and $\mathbf{y}$ and their joint density $p(x, y)$.
The *conditional entropy* is defined as

$$H(\mathbf{y}|\mathbf{x}) = -\int\int \log(p(y|x))p(x,y)dxdy = E_{\mathbf{x},\mathbf{y}}[-\log(p(y|x))] =$$

$$= E_{\mathbf{x},\mathbf{y}}\left[\log\frac{1}{p(y|x)}\right] = E_{\mathbf{x}}[H(\mathbf{y}|x)]$$

This quantity quantifies the remaining uncertainty of $\mathbf{y}$ once $\mathbf{x}$ is known.

Some properties

- for continuous r.v.s the differential entropy may be negative (logarithmic scale)
- in general $H(\mathbf{y}|\mathbf{x}) \neq H(\mathbf{x}|\mathbf{y})$
- conditioning reduces entropy

$$H(\mathbf{y}|\mathbf{x}) \leq H(\mathbf{y})$$

with equality if $\mathbf{x}$ and $\mathbf{y}$ are independent, i.e. $\mathbf{x} \perp\!\!\!\perp \mathbf{y}$,

- $$H(\mathbf{y}) - H(\mathbf{y}|\mathbf{x}) = H(\mathbf{x}) - H(\mathbf{x}|\mathbf{y})$$

Mutual information of two vars (II)

- Given two random variables $\mathbf{x}$ and $\mathbf{y}$, their *mutual information* is defined in terms of their probabilistic marginal density functions $p_{\mathbf{x}}(x)$, $p_{\mathbf{y}}(y)$ and the joint $p_{(\mathbf{x},\mathbf{y})}(x, y)$:

$$I(\mathbf{x};\mathbf{y}) = \int\int \log \frac{p(x,y)}{p(x)p(y)} p(x,y)dxdy =$$
$$= H(\mathbf{y}) - H(\mathbf{y}|\mathbf{x}) = H(\mathbf{x}) - H(\mathbf{x}|\mathbf{y})$$

with the convention that $0 \log \frac{0}{0} = 0$.

Mutual information in the normal case

Let $(\mathbf{x}, \mathbf{y})$ a normally distributed random vector with correlation coefficient ρ.
The mutual information between $\mathbf{x}$ and $\mathbf{y}$ is given by

$$I(\mathbf{x}; \mathbf{y}) = -\frac{1}{2} \log(1 - \rho^2)$$

Equivalently the normal correlation coefficient can be written as

$$\rho = \sqrt{1 - \exp(-2I(\mathbf{x}; \mathbf{y}))}$$

Note that $\rho = 0$ when $I(\mathbf{x}; \mathbf{y})) = 0$ or equivalently $\mathbf{x} \perp\!\!\!\perp \mathbf{y}$.

Mutual information of two vars

- Note that if $\mathbf{x}$ and $\mathbf{y}$ are independent, then $H[\mathbf{y}|\mathbf{x}] = H[\mathbf{y}]$.
- Mutual information

$$I(\mathbf{x};\mathbf{y}) = H(\mathbf{y}) - H(\mathbf{y}|\mathbf{x}) = H(\mathbf{x}) - H(\mathbf{x}|\mathbf{y})$$

  is one of the widely used measures to define dependency of variables.
- It is a measure of the amount of information that one random variable contains about another random variable.
- It can also be considered as the *distance from independence* between the two variables. Indeed if $\mathbf{x}$ and $\mathbf{y}$ are independent $I(\mathbf{x};\mathbf{y}) = 0$.
- This quantity is always non negative and zero if and only if the two variables are stochastically independent.

Conditional mutual information

- Consider three r.v.s $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$. The *conditional mutual information* is defined by

$$I(\mathbf{y};\mathbf{x}|\mathbf{z}) = H(\mathbf{y}|\mathbf{z}) - H(\mathbf{y}|\mathbf{x},\mathbf{z})$$

- The conditional mutual information is null iff $\mathbf{x}$ and $\mathbf{y}$ are conditionally independent given $\mathbf{z}$.
- For a larger number n of variables $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ a chain rule holds

$$I(\mathbf{X};\mathbf{y}) = I(\mathbf{X}_{-i};\mathbf{y}|\mathbf{x}_i) + I(\mathbf{x}_i;\mathbf{y}) = I(\mathbf{x}_i;\mathbf{y}|\mathbf{X}_{-i}) + I(\mathbf{X}_{-i};\mathbf{y}), \quad i = 1, \ldots, n$$

This means that for $n = 2$

$$I(\{\mathbf{x}_1,\mathbf{x}_2\};\mathbf{y}) = I(\mathbf{x}_2;\mathbf{y}|\mathbf{x}_1) + I(\mathbf{x}_1;\mathbf{y}) = I(\mathbf{x}_1;\mathbf{y}|\mathbf{x}_2) + I(\mathbf{x}_2;\mathbf{y})$$

Strong and weak relevance

Let us consider a set $\mathbf{X}$ of n input variables and a target $\mathbf{y}$.

- A variable $\mathbf{x}_i$ is *strongly relevant* to the target $\mathbf{y}$ if

$$I(\mathbf{X}_{-i}; \mathbf{y}) < I(\mathbf{X}; \mathbf{y})$$

  i.e. if it carries information about $\mathbf{y}$ that no other variable carrries.

- A variable is *weakly relevant* to the target $\mathbf{y}$ if it is not strongly relevant and

$$\exists \mathbf{S} \subseteq \mathbf{X}_{-i} : I(\mathbf{S}; \mathbf{y}) < I(\{\mathbf{x}_i, \mathbf{S}\}; \mathbf{y})$$

  i.e. it exists a certain context $\mathbf{S}$ in which it carries information about the target.

Strongly and weakly relevant variables

- Strong relevance indicates that the feature is always necessary for an optimal subset.
- Weak relevance suggests that the feature is not always necessary but may become necessary at certain conditions.
- Irrelevance indicates that the feature is not necessary at all.

Example: Consider a learning problem where $n = 4$, $\mathbf{x}_3 = -\mathbf{x}_2$

$$\mathbf{y} = \begin{cases} 1 + \mathbf{w}, & \mathbf{x}_1 + \mathbf{x}_2 > 0 \\ 0, & \text{else} \end{cases}$$

Which variables are strongly, weakly relevant and irrelevant?

Markov blanket

Let us consider a set $\mathbf{X}$ of n r.v.s., a target variable $\mathbf{y}$ and a subset $\mathbf{M}_y \subset \mathbf{X}$. The subset $\mathbf{M}$ is said to be a *Markov blanket* of $\mathbf{y}$, $\mathbf{y} \notin \mathbf{M}$ iff

$$I(\mathbf{y}; \mathbf{X}_{-(M_y)}) | \mathbf{M}_y) = 0$$

Theorem (Total conditioning)

$$\mathbf{x} \in \mathbf{M}_y \Leftrightarrow I(\mathbf{x}; \mathbf{y} | \mathbf{X}_{-(x,y)}) > 0$$

Definition (Relevance)

The variable relevance of $\mathbf{x}_2$ to $\mathbf{y}$ given $\mathbf{x}_1$, is the conditional mutual information

$$I(\{\mathbf{x}_1, \mathbf{x}_2\}; \mathbf{y}) - I(\mathbf{x}_1; \mathbf{y}) = I(\mathbf{x}_2; \mathbf{y} | \mathbf{x}_1)$$

We can define $\mathbf{x}_1$ as the *context* and consider the relevance of a variable $\mathbf{x}_2$ as *context-dependent*.

For an empty context, the relevance boils down to the mutual information $I(\mathbf{x}_2; \mathbf{y})$ of $\mathbf{x}_2$ to $\mathbf{y}$.

Interaction

Note that since

$$I(\{\mathbf{x}_1, \mathbf{x}_2\}; \mathbf{y}) = I(\mathbf{x}_2; \mathbf{y}|\mathbf{x}_1) + I(\mathbf{x}_1; \mathbf{y}) = I(\mathbf{x}_1; \mathbf{y}|\mathbf{x}_2) + I(\mathbf{x}_2; \mathbf{y})$$

we have

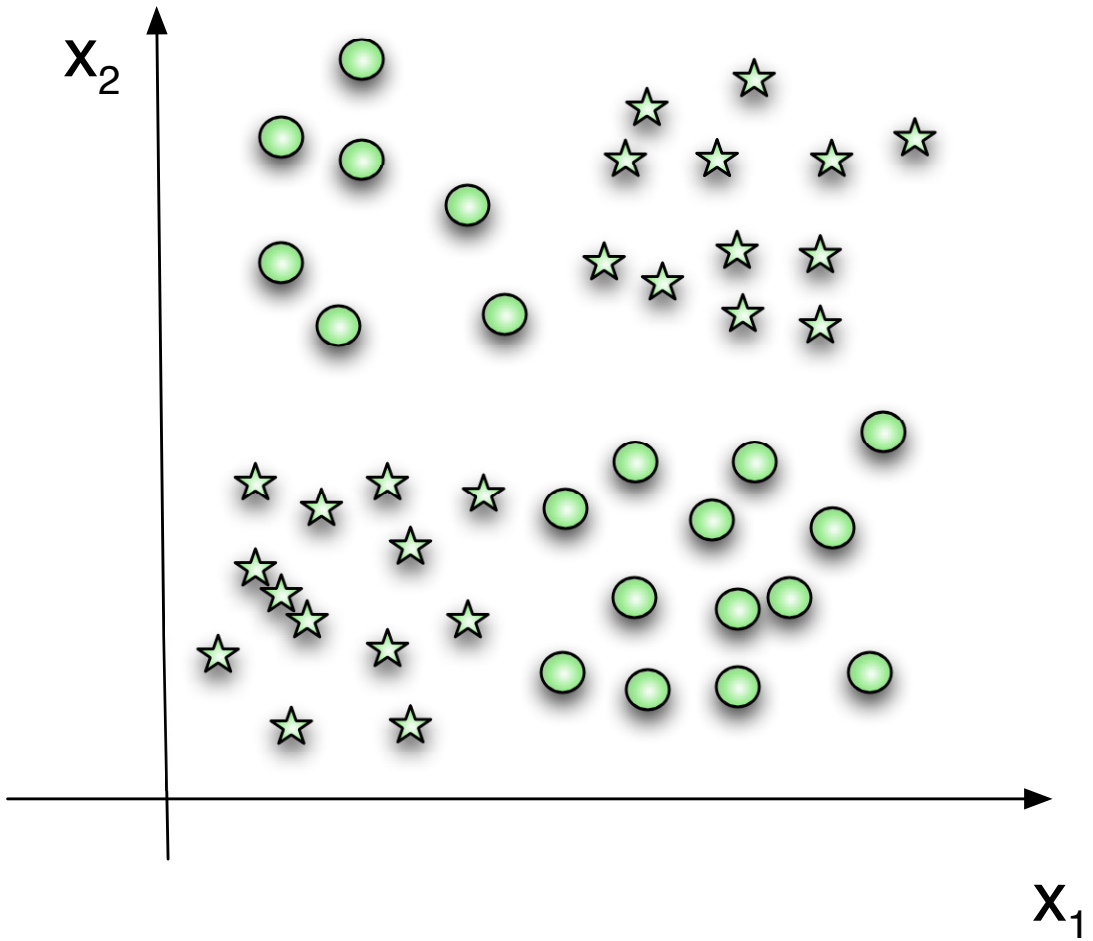
$$I(\mathbf{x}_2; \mathbf{y}|\mathbf{x}_1) = I(\mathbf{x}_2; \mathbf{y}) - I(\mathbf{x}_1; \mathbf{y}) + I(\mathbf{x}_1; \mathbf{y}|\mathbf{x}_2)$$

It follows that

$$I(\{\mathbf{x}_1, \mathbf{x}_2\}; \mathbf{y}) = I(\mathbf{x}_1; \mathbf{y}) + I(\mathbf{x}_2; \mathbf{y}) - \underbrace{[I(\mathbf{x}_1; \mathbf{y}) - I(\mathbf{x}_1; \mathbf{y}|\mathbf{x}_2)]}_{\text{interaction}}$$

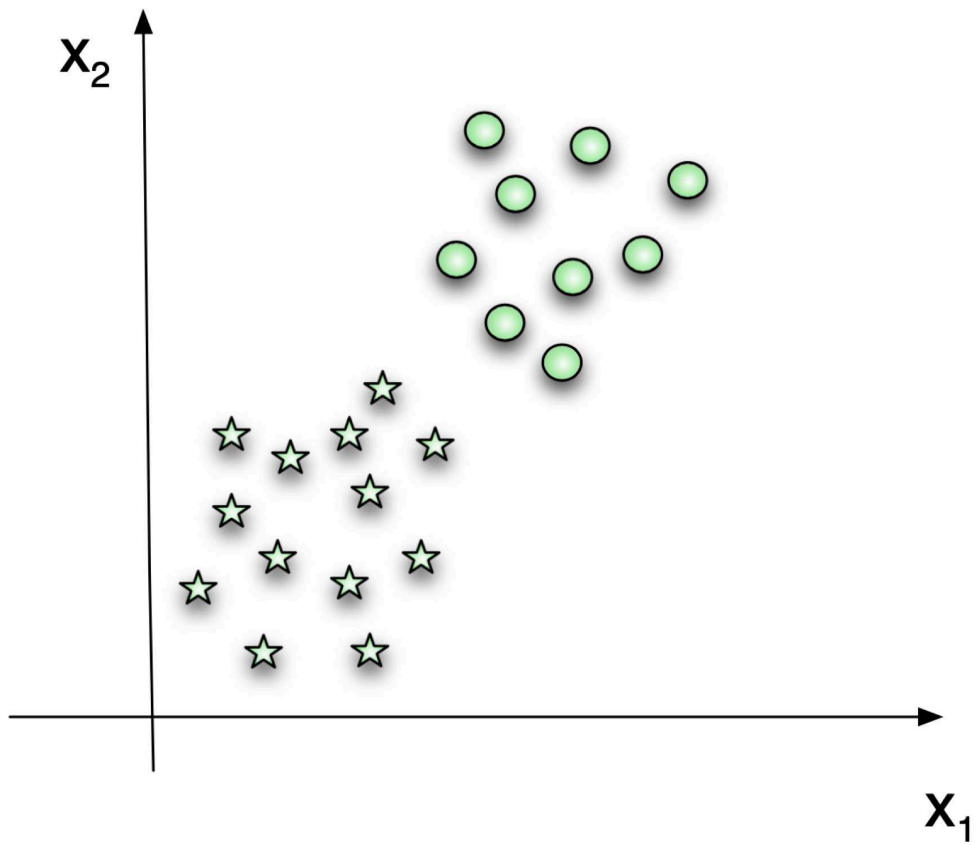
Negative interaction: complementary variables, i.e. the variable together have more information than the sum of the univariate informations (XOR).
Positive interaction: redundant variables.

Negative interaction (XOR)



$I(\mathbf{x}_1; \mathbf{y}) = 0, I(\mathbf{x}_2; \mathbf{y}) = 0$ but $I(\mathbf{x}_1; \mathbf{y}|\mathbf{x}_2) > 0$

Positive interaction



$I(\mathbf{x}_1; \mathbf{y}) > 0, I(\mathbf{x}_2; \mathbf{y}) > 0$ but $I(\mathbf{x}_1; \mathbf{y}|\mathbf{x}_2) = 0$

Feature selection and mutual information

- Given an output target $\mathbf{y}$ and a set of input variables $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ selecting the optimal subset of d variables boils down to

$$X^* = \arg \max_{X_S \subset X, |X_S| = d} I(X_S; \mathbf{y})$$

- Maximization can be done in a forward manner.
- Let $X = \{x_i\}, i = 1, \ldots, n$ the whole set of variables and $\mathbf{X}_S$ the current set of selected variables. The task of adding a variable $x^* \in S - X$ can be addressed by solving

$$x^* = \arg \max_{\mathbf{x}_k \in X - X_S} I(\{\mathbf{X}_S, \mathbf{x}_k\}; \mathbf{y})$$

  This requires a multivariate estimation of the mutual information.
- Filter approaches rely on some low variate approximation.

The mRMR approach

The mRMR (mimimum-Redundancy Maximum-Relevancy) feature selection strategy approximates

$$\arg \max_{\mathbf{x}_k \in X - X_S} I(\{\mathbf{X}_S, \mathbf{x}_k\}; \mathbf{y})$$

with

$$x^*_{\text{MRMR}} = \arg \max_{\mathbf{x}_k \in X - X_S} \left[I(\mathbf{x}_k; y) - \frac{1}{m} \sum_{\mathbf{x}_i \in X_S} I(\mathbf{x}_i; \mathbf{x}_k) \right]$$

where m is the size of X_S.