18. Feature Selection, Transformation, and Learning

Feature Selection

The Big Picture

Statistical learning is the process of learning functions that associate features with corresponding responses in order to maximize some information gain.

This often involves extracting quantities -- known as features -- from real-world data that can be used by statistical learning algorithms.

Such procedure – involving the extraction of features from real-world data – is a manifestation of our prior knowledge about the problem.

Feature Selection

Feature Selection

It is well known that all the features are now always relevant to perform a certain statistical learning task.

Different metrics such as correlation coefficients, mutual information, statistical tests, p-values, information gain etc can be used to gauge the relevance of different features.

The use of a large number of features can be problematic for various reasons:

- These may impair statistical prediction accuracy
- These may entail prohibitive computational power
- These often obstruct interpretability

The solution is to reduce the input feature space (N) to a lower dimensional feature space ($n, n \ll N$) using feature engineering techniques.

Feature Selection

Feature Selection

Feature selection, also known as variable selection, attribute selection or variable subset selection is the process of selecting the most *parsimonious* feature subset with *maximum joint information content*

It involves retaining features that contribute towards predicting the response, and discarding features that do not.

X Y

samples	feature1	feature2	..	featureN	outcome
s_1	1.2	4.2	..	2.1	C_1
s_2	3.0	6.4	..	3.2	C_1
s_3	12	2.8	..	1.4	C_0
..	..	..	..	..	..
s_m	4.7	1.5	..	0.75	C_0

Note that feature 2 is twice feature N hence it is redundant.

Feature Selection

Procedure

Ideally, feature selection would involve running a brute force search over all subsets of n out of N features in order to choose the subset of features leading to the best performing predictor.

This feature selection procedure is however not feasible – even for moderate values of n and N – because it involves testing $\binom{N}{n}$ combinations

The feature selection procedure therefore leverages two components

- A search strategy used to guide exploration of space of all possible feature combinations
- A objective function used to assess the quality of possible feature combinations

Feature Selection

Procedure – Objective Functions

Filters

This is a simple approach measuring the quality of each individual feature using quantities such as

- interclass distance measures: These measure class separability using metrics such as Euclidean or other distances.
- information-theoretic measures: These measure feature quality leveraging the idea that good feature subsets contain features highly correlated with the response, but uncorrelated with each other.

Wrappers

Wrappers evaluate subsets on the basis of their classification accuracy (on test data).

Feature Selection

Procedure – Search Strategies

Greedy (or Sequential) Algorithms

The simplest instance of greedy selection is forward greedy selection involving

- Start with an empty set of features I.
- Gradually add one feature $i \notin I$ at a time to the set of selected features $I \cup \{i\}$ depending on its effect on the objective function
- Repeat until one either selects n features, where n is a predefined budget of allowed features, or one achieves an accurate enough predictor

Another popular greedy selection approach is backward elimination where we start with the full set of features and then we gradually remove one feature at a time from the set of features.

These methods tend to be trapped in local minima but this can be partly offset by incorporating randomness in the search procedure.

Feature Selection

Other Procedures

Sparsity Inducing Norms:

Given a predefined feature budget n, feature selection can also be carried out by solving the constrained optimization problem given by:

$$\min_{\theta} L_S(\boldsymbol{\theta}) \qquad s.t. \quad \|\boldsymbol{\theta}\|_0 \leq n$$

where $L_S(\boldsymbol{\theta})$ is an empirical risk, $\boldsymbol{\theta}$ is a feature vector, and $\|\boldsymbol{\theta}\|_0 = |\{i|\theta_i \neq 0\}|$ is the l_0 norm. This is a computationally hard optimization problem.

The feature selection problem can also be carried out efficiently using the proxy optimization problem – corresponding to a convex relaxation of the previous one – given by:

$$\min_{\theta} L_S(\boldsymbol{\theta}) \qquad s.t. \quad \|\boldsymbol{\theta}\|_1 \leq n$$

where $\|\boldsymbol{\theta}\|_1 = \sum_i |\theta_i|$ is the l_1 norm**.** This is now a computationally feasible convex optimization problem.

These problems allow one to carry out both feature tuning and feature selection simultaneously in view of the fact they promote sparse solutions.

Example: Feature Selection with Random Forest

Random Forest Algorithm

- Random Forest is a supervised learning algorithm that:
  - builds an ensemble of Decision Trees
  - combines trees together to obtain a more accurate and stable prediction
  - can be used for both classification and regression

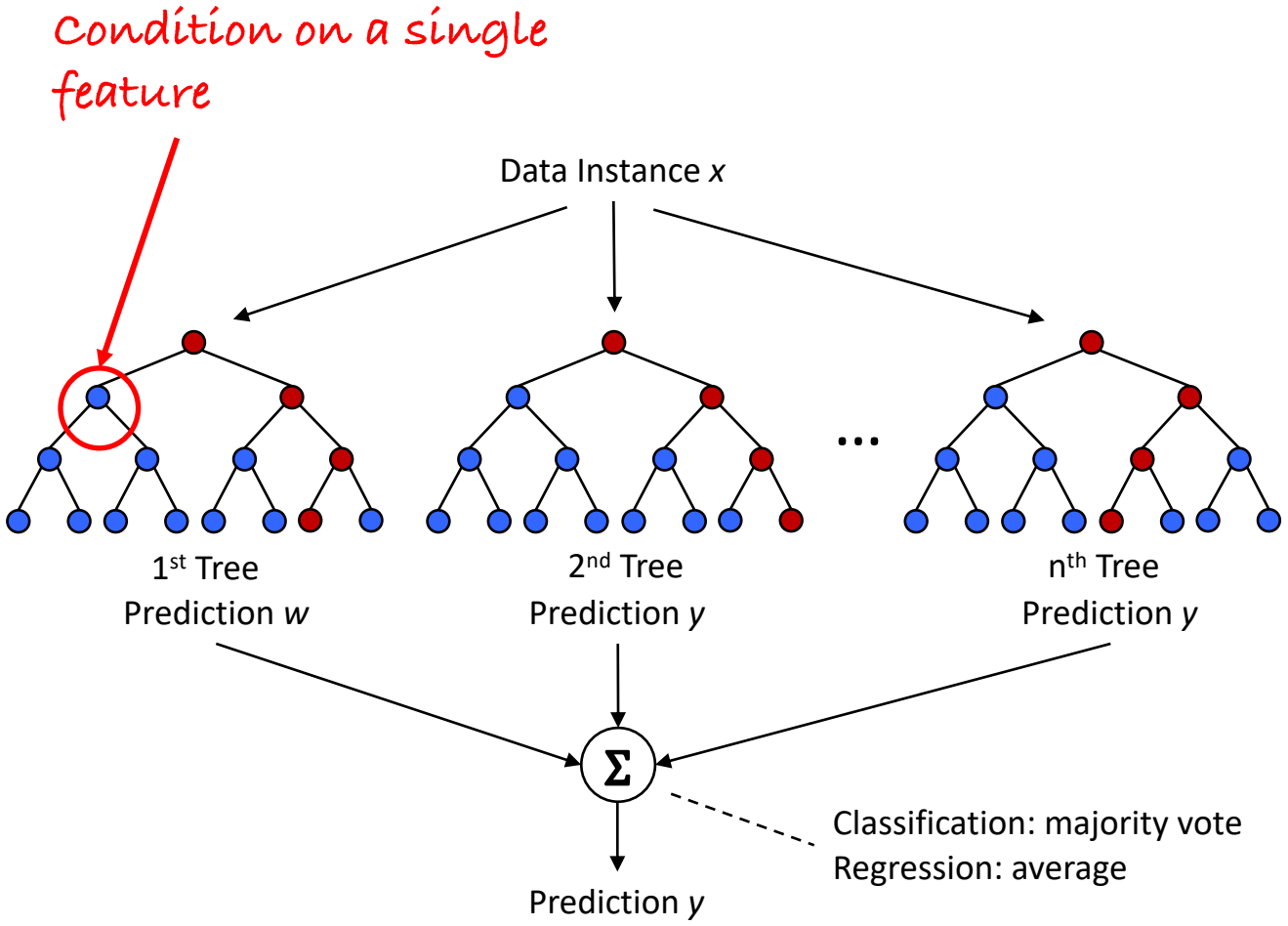
- Random Forests are often used to measure the relative importance of each feature on the prediction

Example: Feature Selection with Random Forest

Random Forest Algorithm

- Random Forest is a supervised learning algorithm that:
  - builds an ensemble of Decision Trees
  - combines trees together to obtain a more accurate and stable prediction
  - can be used for both classification and regression

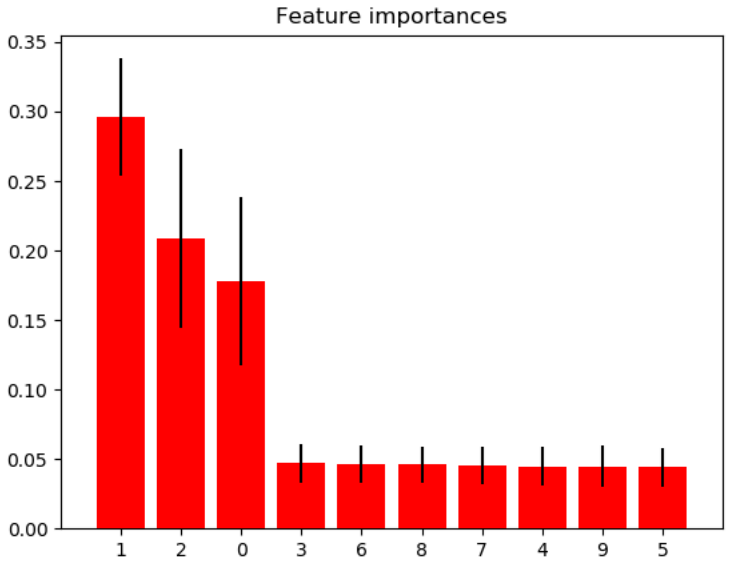
- Random Forests are often used to measure the relative importance of each feature on the prediction



Example: Feature Selection with Random Forest

- The *scikit-learn* library provides a function that measures feature importance by assessing at how much the tree nodes, which use that feature, reduce impurity across all trees in the forest

- Dimensionality reduction is achieved by selecting the top k features or features with importance above a certain threshold




```
Feature ranking:
1. feature 1 (0.295902)
2. feature 2 (0.208351)
3. feature 0 (0.177632)
4. feature 3 (0.047121)
5. feature 6 (0.046303)
6. feature 8 (0.046013)
7. feature 7 (0.045575)
8. feature 4 (0.044614)
9. feature 9 (0.044577)
10.feature 5 (0.043912)
```

Source: https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html

Feature Transformation

Feature Transformation

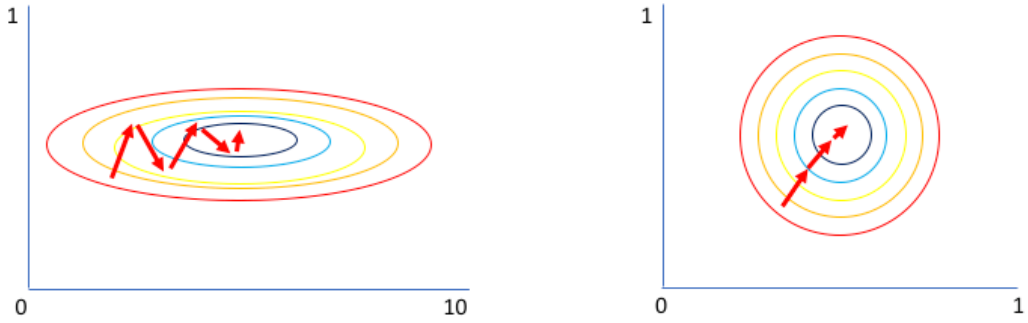
Feature manipulations such as normalization and clipping can lead to various benefits such as: (1) improve approximation error (2) improve estimation error or (3) yield faster learning algorithms

Such feature manipulation techniques often depend on the statistical procedure or the data.

It is also sometimes helpful to combine some of the feature transformations (e.g., centering + scaling).

Example

Two features having completely different scales may result in a slow gradient descent algorithm



Source: https://www.jeremyjordan.me/batch-normalization/

Feature Learning

Feature Learning

Feature learning involves learning a mapping function, $\psi : \mathcal{D} \rightarrow \mathbb{R}^d$ that maps instances in the data space $\mathcal{D}$ into d-dimensional feature vectors.

Feature learning automates the feature design process typically in a data-driven manner.

The No-Free-Lunch theorem tells us that some prior knowledge on the data distribution must be incorporated in order to build a good feature representation.

There are currently various feature learning approaches including deep learning based techniques (supervised), auto-encoder based techniques (unsupervised), or dictionary learning based approaches