Higgs Boson classification

Laurent Lejeune, Tatiana Fountoukidou, Guillaume de Montauzon

I. Introduction

II. Data pre-processing and exploration

A. Data clean-up

About 70% of the samples contain missing values. Replacing missing values by the expectation over the valid samples leads to a distortion of the variable's distribution, which introduces a severe bias in regression procedures. The strategies that were implemented and tested are now introduced.

1) Least-squares regression
2) Attribute removal Among the 30 attributes given in the datasets, 11 contain missing values. The most straight-forward appraoch is to remove those attributes from the sets and thus reduce the dimensionality of the problem.
3) K-Nearest-Neighbors regression A variant of the K-Nearest-Neighbors algorithm was implemented to fill-in missing values [**?**]. To alleviate the computational cost, a random uniform sampling of the valid samples (samples without missing values) was performed prior to the nearest neighbors search. The missing values are replaced by the weighted average value over its K nearest neighbors using Euclidean distance.

B. Data preparation

Computing the eigen-values of the covariance matrix reveals that the second highest eigen value is xxx% smaller than the highest.

C. Analysis of missing values

The class probabilities given the presence or absence of missing values are computed on the training set.

- $P(Y = 1|X\text{has missing attributes}) \approx 0.30$
- $P(Y = -1|X\text{has missing attributes}) \approx 0.70$
- $P(Y = 1|X\text{has no missing attributes}) \approx 0.47$
- $P(Y = -1|X\text{has no missing attributes}) \approx 0.53$

The above results justifies the choice of adding an indicator variable x_{miss} that takes value 0 if no attributes are missing and 1 otherwise.

III. Methods

A. AdaBoost with decision stumps

The idea of adding weak learners in a stage-wise manner to produce a strong classifier is commonly referred to as boosting. Discrete AdaBoost, described in [**?**], consists in adapting the weights of samples based on the missclassification error. The goal is to penalize missclassified samples and re-inject the weights for the computation of the next stage.

Algorithm 1 Discrete AdaBoost

```
1: Start with weights $w_i = \frac{1}{N}, i = 1, ..., N$
2: for $t = 1, 2, ..., T$ do
3:     Fit the classifier $h_t(\boldsymbol{x}) \in \{-1, 1\}$ using weights $w_i$
4:     Compute $\boldsymbol{e}_t = \sum_{i=1}^{N} \boldsymbol{w}_i, t$, where $h_t(x_i) \neq y_i$
5:     Choose $\alpha_t = \frac{1}{2} \log \frac{1-\boldsymbol{e}_t}{\boldsymbol{e}_t}$
6:     Add to ensemble: $\boldsymbol{F}_t(\boldsymbol{x}) = \boldsymbol{F}_{t-1}(\boldsymbol{x}) + \alpha_t h_t(x)$
7:     Update weights: $\boldsymbol{w}_{i,t+1} = \boldsymbol{w}_{i,t} e^{-\boldsymbol{y}_i \alpha_t \boldsymbol{h}_t(x)}$
8:     Renormalize $\boldsymbol{w}_{i,t+1} such that \sum_i \boldsymbol{w}_{i,t+1} = 1$
9: end for
10:
```

A very simple version of AdaBoost was implemented that finds optimal decision stumps as weak-learners, that is, thresholding function that best separates the positive and negative classes based on a single attribute.