ABSTRACT

Background: Learning a regression function using controlled or interval-valued output data is an important problem in fields such as geology and medicine. The goal is to learn a real-valued prediction function, and the training output labels indicate an interval of possible values.

Methods: Whereas most existing algorithms for this task are linear models, we investigate learning nonlinear tree models. We propose to learn a tree by minimizing a margin-based discriminative objective function, and we provide a dynamic programming algorithm for computing the optimal solution in log-linear time.

Results: We show empirically that this algorithm achieves state-of-the-art speed and prediction accuracy in a benchmark of several real data sets.

INTRODUCTION

Learning setting

We are given a set of labelled examples, where the target values are intervals indicating a range of acceptable values:

\[ \mathcal{D} = \{(x_i, [y_i, y_i]) : i = 1, \ldots, n\} \]

where

- \( x_i \) is a vector of features
- \( y_i \) is an interval, such that \( y_i = [\mu_i, \delta_i] \), with \( \mu_i \leq \delta_i \) and \( \mu_i, \delta_i \in \mathbb{R} \)

2) It is an unknown data-generating distribution.

Objective

Learn a model \( f : \mathbb{R}^d \rightarrow \mathbb{R} \) that, given an example \( x \), predicts a value inside the corresponding interval \( y \). To achieve the following convex objective function:

\[ \min_{\theta \in \Theta} \sum_{i=1}^{n} \ell(y_i, f(x_i)) \]

where \( \ell(y, f) = \max\{0, |y - f| - \epsilon\} \) is the corresponding hinge loss, where \( |\cdot| \) is the positive part function, i.e., \( \max(0, x) = x \) if \( x > 0 \) and \( = 0 \) otherwise.

Loss functions

We consider two types of loss functions:

- Linear hinge loss: \( \ell(y, f) = \max\{0, \epsilon - y + f(x)\} \)
- Squared hinge loss: \( \ell(y, f) = (y - f(x))^2 \)

DYNAMIC PROGRAMMING ALGORITHM

Each hinge loss in the minimization problem can be rewritten in the general form:

\[ \ell(y, f) = \max\{0, \epsilon - y + f(x)\} \]

A sum of such hinge losses is a convex piecewise function:

\[ \ell_n(y, f) = \sum_{i=1}^{n} \max\{0, \epsilon - y_i + f(x_i)\} \]

Optimization step – tracking the function’s minima

The dynamic programming algorithm maintains a pointer on the left-most breakpoint that is to the right of all minima of \( \ell_n(y, f) \). The optimization step consists in finding the new position of the pointer after a hinge loss is added to the sum.

CONCLUSION

We proposed a new margin-based decision tree algorithm for the interval regression problem.

We showed that it could be trained by solving a sequence of convex subproblems, for which we proposed a new dynamic programming algorithm.

Like classical regression trees (Breiman et al., 1984), our tree growing algorithms time complexity is linear in the number of features and log-linear in the number of examples.

We studied the prediction accuracy in several real and simulated data sets, showing that our algorithm is competitive with other linear and nonlinear models for interval regression.

REFERENCES


MMITs recover a good approximation in simulations with nonlinear patterns

MMITs achieve state-of-the-art accuracies on real and simulated data sets

MMITs recover a good approximation in simulations with nonlinear patterns

MMITs achieve state-of-the-art accuracies on real and simulated data sets

RESULTS

The dynamic programming algorithm runs in log-linear time for both hinge losses

The dynamic programming algorithm runs in log-linear time for both hinge losses

The dynamic programming algorithm runs in log-linear time for both hinge losses

The dynamic programming algorithm runs in log-linear time for both hinge losses

The dynamic programming algorithm runs in log-linear time for both hinge losses

The dynamic programming algorithm runs in log-linear time for both hinge losses

The dynamic programming algorithm runs in log-linear time for both hinge losses

The dynamic programming algorithm runs in log-linear time for both hinge losses

The dynamic programming algorithm runs in log-linear time for both hinge losses

The dynamic programming algorithm runs in log-linear time for both hinge losses

The dynamic programming algorithm runs in log-linear time for both hinge losses

The dynamic programming algorithm runs in log-linear time for both hinge losses

The dynamic programming algorithm runs in log-linear time for both hinge losses

The dynamic programming algorithm runs in log-linear time for both hinge losses

The dynamic programming algorithm runs in log-linear time for both hinge losses

The dynamic programming algorithm runs in log-linear time for both hinge losses

The dynamic programming algorithm runs in log-linear time for both hinge losses