Feature Cost Sensitive Random Forest

Anna Thomas
Stanford University

Summary

In recent years cost-sensitive classification and regression have emerged as key challenges in the practical implementation of machine learning methods. Here we focus on the random forest model [1] and explore strategies for cost-sensitive forest training. We develop and test two algorithms for this task, and apply them to the problems of cost-sensitive diabetes diagnosis, digit recognition, and spam filtering. We demonstrate that for these three real world problems, computational cost at test time can be substantially reduced without significantly compromising accuracy.

Problem Statement

Similar to the formulation in [2], our goal is to learn a classifier F from a family of functions $\mathcal{F}$ that minimizes the sum of the expected errors and the computational cost of the final feature set:

$$\min_{F \in \mathcal{F}} E_{xy}[L(y, f(x))] + \lambda E_x[C(f, x)]$$

where $L(y, \hat{y})$ is a loss function and $C(f, x)$ is the cost of evaluating the function of f on example x.

Our formulation differs from [2] in that we do not have a constraint on the feature costs, but rather incorporate the cost minimization into the objective itself.

Since in practice we are given a training set, not a distribution, we will instead solve the following problem:

$$\min_{F \in \mathcal{F}} \sum_{i=1}^{N} L(y^{(i)}, f(x^{(i)})) + \lambda \sum_{j=1}^{|F_S|} C_j$$

Randomized Greedy Algorithm

Algorithm 1: Cost Sensitive RF

Input: $X \in \mathbb{R}^{mxn}$, $y \in \mathbb{Z}^m$, $C \in \mathbb{R}^n$, $N \in \mathbb{Z}$, $\lambda \in \mathbb{R}$

$\mathcal{T} \leftarrow \emptyset$;
for *each tree i=1:N* **do**
  Randomly sample training data to form X^i and y^i
  $T, C' \leftarrow \text{GreedyTree}(X^i, y^i, C, \lambda)$
  $C := C'$
  $\mathcal{T} \leftarrow \mathcal{T} \cup T$
return $\mathcal{T}$

Randomized Greedy Algorithm

Algorithm 2: GreedyTree

for *each attribute i=1:M* **do**
  Randomly sample splits s_{ij} and compute $F(s_{ij}) = H(T) - H(T|s_{ij}) - \lambda C_i$, where $H(T)$ is the information entropy
$\hat{s} \leftarrow \text{argmin}_s F(s)$
$C_i := 0$
Create new node using feature i and split value j.
for *each child node* **do**
  $\text{GreedyTree}((X^i)_{\hat{s}}, (y^i)_{\hat{s}}, C, \lambda)$
return T, C

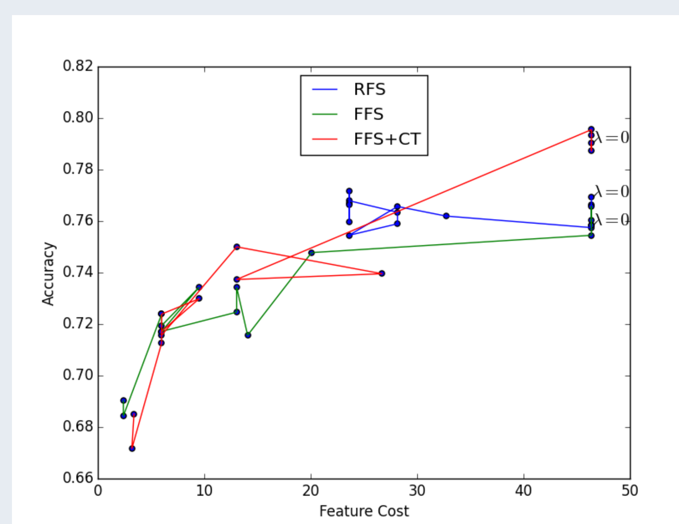
Complementary Tree Training

Algorithm 3: Complementary Cost Sensitive RF

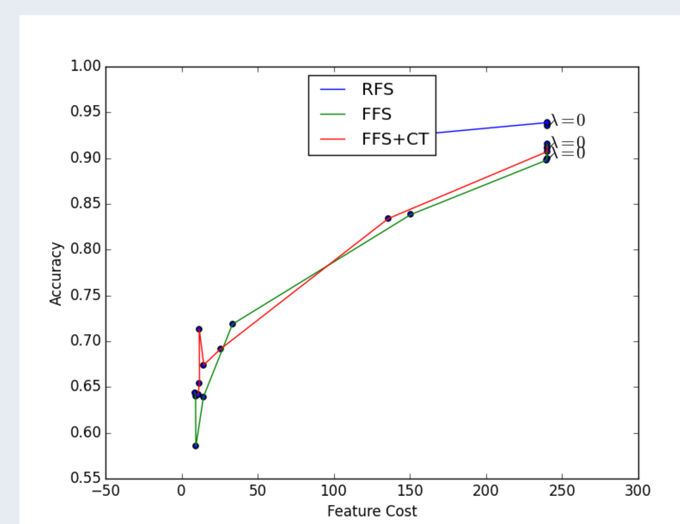
Input: $X \in \mathbb{R}^{mxn}$, $y \in \mathbb{Z}^m$, $C \in \mathbb{R}^n$, $N \in \mathbb{Z}$, $\lambda \in \mathbb{R}$

$\mathcal{T} \leftarrow \emptyset$;
for *each training example j=1:M* **do**
  $W_j = \frac{1}{M}$
for *each tree i=1:N* **do**
  Randomly sample training data to form X^i and y^i
  $T, C' \leftarrow \text{GreedyTree}(X^i, y^i, C, \lambda)$
  $C := C'$
  for *each training example j=1:M* **do**
    $\epsilon_j = \frac{1}{|\mathcal{T}|} \Sigma_{T_i \in \mathcal{T}} I(h_i(x_j) = y_j)$
    $W_j = \epsilon_j$
  $\mathcal{T} \leftarrow \mathcal{T} \cup T$
return $\mathcal{T}$

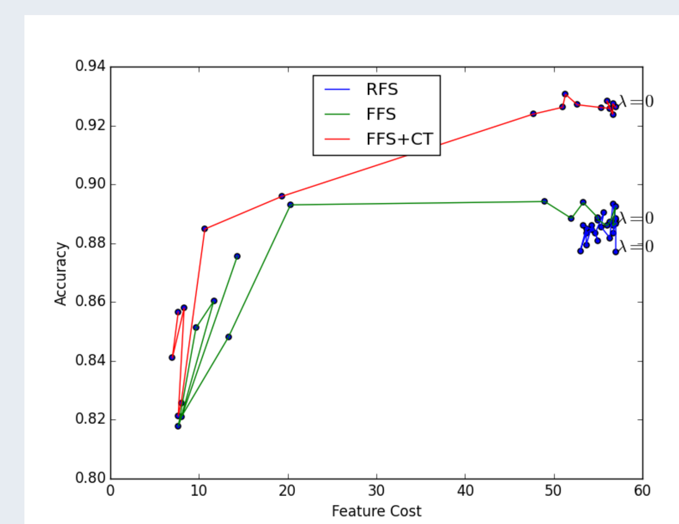
Results



(a) Pima Indian diabetes diagnosis dataset, with provided costs shown in Table 2.



(b) Digit recognition dataset. Costs were not provided, so we use $C(S) = |S|$.



(c) Spam filtering dataset. Costs were not provided, so we use $C(S) = |S|$.

Figure 1: Cost versus accuracy tradeoff for three datasets. FFS - full feature set considered at each split; RFS - randomized feature set of size $\sqrt{n}$ considered at each split; FFS + CT - full feature set and complementary training (Algorithm 3).

Datasets

Dataset	Num. Instances	Num. Attributes
Diabetes Diagnosis	768	8
Spam Filtering	4601	57
Digit Recognition	2000	240

Table 1: Datasets used in this work, obtained from the UCI Machine Learning Repository located at `http://archive.ics.uci.edu/ml/`.

Feature Costs

Feature	Cost
Num. Times Pregnant	1.00
Glucose Tolerance	17.61
Diastolic Pb	1.0
Triceps	1.0
Insulin	22.78
Mass Index	1.0
Pedigree	1.0
Age	1.0

Table 2: Feature costs provided in the diabetes dataset.

Conclusion

Here we show that a simple modification to the random forest algorithm allows for user control over the computational cost of the trained classifier. In order to account for the increased intra-forest correlation when more variables are considered at each node split, we also test a boosting-like iterative sample reweighting strategy (based on [3]), which generally improves performance. Overall, these results indicate that for several real world problems it is possible to significantly reduce test time cost with minimal effects on accuracy.

Future Work

- Instead of selecting features greedily while constructing forest splits, select features before building forest. Consider variations on standard wrapper and filter methods.
- In the case where common feature computational subroutines exist, treat as a submodular optimization problem, for which efficient approximation algorithms exist.
- At each split, optimize the number of variables to include in random subset, rather than using a fixed number.

Selected References

[1] Leo Breiman.
Random forests.
Machine Learning, 45, 2001.

[2] Feng Nan, Joseph Wang, and Venkatesh Saligrama.
Feature-budgeted random forest.
Journal of Machine Learning, 37, 2015.

[3] Simon Bernard, Sebastian Adam, and Laurent Heutte.
Dynamic random forests.
Pattern Recognition Letters, 33, 2012.

Acknowledgements

Thank you to Junjie Qin for valuable feedback.