The Greedy Miser: Learning under Test-time Budgets

Zhixiang (Eddie) Xu, Kilian Q. Weinberger
Department of Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO 63130
zhixiang.xu, kilian@wustl.edu

Olivier Chapelle
Yahoo Research
Santa Clara, CA, 95054
chap@yahoo-inc.com

As machine learning algorithms increasingly enter real-world settings, there is rising interest in controlling the cpu-cost during test-time. In industry, computational resources must be budgeted and costs must be strictly accounted for. At its very core, this problem is inherently a *tradeoff* between accuracy and test-time computation. Test-time computation consists of two components: 1. the actual running time of the algorithm; 2. the time required for feature extraction. The latter can vary drastically if the feature set is diverse.

In this abstract, we propose a novel algorithm that explicitly considers the feature extraction cost during training. We first state the (non-continuous) global objective, which explicitly trades off feature cost and accuracy, and then relax it into a continuous loss function. Subsequently, we derive an update rule that shows the resulting loss lends itself naturally to greedy optimization with stage-wise regression [4]. The resulting learning algorithm is much simpler than any prior work, yet leads to superior test-time performance. Its accuracy matches that of the unconstrained baseline (with unlimited resources) while achieving an order of magnitude reduction of test-time cost.

Cost-sensitive learning. We use gradient-boosting [4] to learn a classifier $H(\mathbf{x}) = \sum_{t=1}^{T} \beta_t h_t(\mathbf{x})$ to minimize some loss $\ell(H)$. Here, $h_t \in \mathcal{H}$ where $\mathcal{H}$ is the set of all possible regression trees [1] of some limited depth b (e.g. $b = 4$), with $T = |\mathcal{H}|$. To account for the test-time cost, we construct an explicit constraint that limits the function-evaluation cost of all trees $h_t(\cdot)$ with $\beta_t \neq 0$ and the feature extraction cost for all features that are used in these trees to be within a budget $B \geq 0$. Let $e > 0$ be the cost to evaluate one tree $h_t()$ if all features are extracted and $c_\alpha > 0$ denote the extraction cost for feature α. With this notation the two parts of the test-time cost can be expressed in a single l_0-norm cost-constraint as

$$e\|\boldsymbol{\beta}\|_0 + \sum_{\alpha=1}^{d} c_\alpha \| \sum_{t=1}^{T} F_{\alpha,t}\beta_t \|_0 \leq B, \tag{1}$$

where the indicator $F_{\alpha,t} \in \{0, 1\}$ is defined as $F_{\alpha,t} = 1$ iff h_t splits on feature α.

Relaxation. The cost function, as stated in eq. (1) is non-continuous, because of the l_0-norm in the cost term — and non-trivial to optimize. The l_0-norm is often relaxed into the convex and continuous l_1-norm. If $\boldsymbol{\beta}$ is found with gradient boosting and step-size η, it is typically the case that $\frac{1}{\eta}\boldsymbol{\beta}$ is a binary vector. In this case, the re-scaled l_1 norm is identical to the l_0 norm — and the relaxation is exact. We use this approach for the first term, the tree-evaluation cost:

$$e\|\boldsymbol{\beta}\|_0 \longrightarrow \frac{e}{\eta}\|\boldsymbol{\beta}\|_1. \tag{2}$$

For the feature cost, the l_1 norm is not a very good approximation of the original l_0-norm formulation. The reason is that features are re-used many times, in different trees. Using the l_1-norm would imply that features that are used more often would be penalized more than features that are only used once. In contrast, the l_0-norm penalty in (1) makes features *free* after their initial construction — a much more realistic cost model. We therefore define a new function $q()$, which is an amputated version of the ℓ_1-norm, and obtain:

$$\sum_{\alpha=1}^{d} c_\alpha \| \sum_{t=1}^{T} F_{\alpha,t}\beta_t \|_0 \longrightarrow \sum_{\alpha=1}^{d} c_\alpha q\left(\sum_{t=1}^{T} F_{\alpha,t}\beta_t \right), \text{ where: } q(x) = \begin{cases} \left|\frac{x}{\eta}\right| & \text{for } |x| \in [0, \eta) \\ 1 & \text{for } |x| \in [\eta, \infty). \end{cases} \tag{3}$$

This penalty function $q()$ behaves like the regular ℓ_1 norm when $|x|$ is small, but is capped to a constant when $x \geq \eta$. Similar to the previous case, if $\frac{1}{\eta}\boldsymbol{\beta}$ is binary this relaxation is exact. This holds because in (3) all

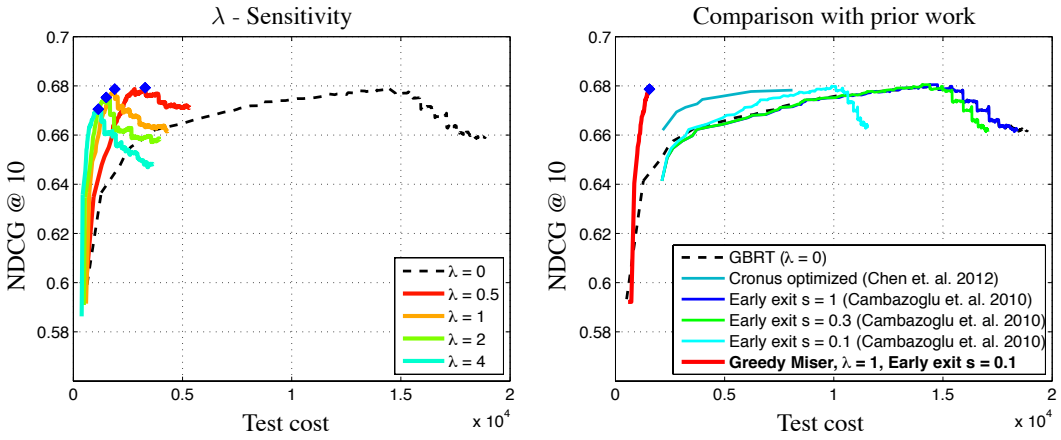

Figure 1: Ranking results on the Yahoo ranking data set.

arguments of $q()$ are non-negative multiples of η and it is easy to see from the definition of $q()$ that for all $k = 0, 1, \ldots$, we have $q(k\eta) = \|k\eta\|_0$.

Optimization. If we explicitly divide the budget into tree-evaluation cost and feature cost $B = B_t + B_f$, we can derive the relaxed optimization problem

$$\min_{\boldsymbol{\beta}} \ \ell(\boldsymbol{\beta}) + \lambda \sum_{\alpha=1}^{d} c_\alpha q\left(\sum_t F_{\alpha,t}\beta_t\right) \text{ such that: } \frac{1}{\eta}\|\boldsymbol{\beta}\|_1 \leq \frac{B_t}{e}. \tag{4}$$

Here, $\lambda \geq 0$ regulates the feature-cost budget B_f. Following the approach from [5], and with some mild assumption, we prove (details omitted) that the optimization problem (4) can be solved (locally) by stage-wise regression [4]. We define $t_i = \frac{\partial \mathcal{L}}{\partial H(x_i)}$, the derivative w.r.t. the current prediction of x_i. During iteration t, the next tree h_t is computed with the help of the CART [1] algorithm with the following impurity function, and $\beta_t = \eta$:

$$h_t = \operatorname*{argmin}_{h_t \in \mathcal{H}} \frac{1}{2}\sum_i^n (h_t(x_i) - t_i)^2 + \lambda \sum_{\alpha=1}^{d} c_\alpha \Omega_\alpha(h_t), \text{ with: } \Omega_\alpha(h_t) = \begin{cases} \frac{1}{\eta}F_{\alpha,t} & |\sum_t F_{\alpha,t}\beta_t| < \eta \\ 0 & |\sum_t F_{\alpha,t}\beta_t| \geq \eta. \end{cases} \tag{5}$$

Because our algorithm is greedy and minimizes cost, we refer to it as *the Greedy Miser* (short *miser*).

Results. Figure 1 shows the performance of the *Greedy Miser* on the Yahoo Learning to Rank data set[1]. The left plot compares the performance of *miser* under different values of λ. The blue dots indicate the best-performing iteration according to the validation data set. The right plot compares the best performing *miser* settings (based on validation result) with other state-of-the-art cost-sensitive algorithms. We include gradient-boosting (GBRT) [4] as baseline (this is identical to *miser* with $\lambda = 0$), Early Exit [2] and Cronos [3]. Two trends can be observed: 1. Larger λ reduces the cost at early stages; and 2. *Miser* significantly improves the cost/precision performance over other algorithms. In terms of running-time, *miser* is identical with GBRT and, unlike Cronus, can easily scale to data sets with millions of examples.

References

[1] L. Breiman. *Classification and regression trees*. Chapman & Hall/CRC, 1984.

[2] B.B. Cambazoglu, H. Zaragoza, O. Chapelle, J. Chen, C. Liao, Z. Zheng, and J. Degenhardt. Early exit optimizations for additive machine learned ranking systems. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 411–420. ACM, 2010.

[3] Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, and Olivier Chapelle. Classifier cascade for minimizing feature evaluation cost. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, page to appear. AISTATS, 2011.

[4] J.H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[5] S. Rosset, J. Zhu, and T. Hastie. Boosting as a regularized path to a maximum margin classifier. *The Journal of Machine Learning Research*, 5:941–973, 2004.

[1] `http://learningtorankchallenge.yahoo.com/datasets.php`