Adaptive Classification Under a Budget
Feng Nan, Venkatesh Saligrama
Boston University

Problem
Resource constraints during prediction:
- Feature acquisition cost (e.g., Health, Security, Sensor Network)
- Communication, latency cost (e.g., Mobile/Cloud computing)
- Computation, latency cost (e.g., Search engine)
- Limited Memory, Battery (e.g., IoT, Mobile)

Consider two applications:

IoT/Mobile computing application:
- Complex DNN models achieve high accuracy in Vision/NLP applications.
- In test-time, such models are slow to evaluate and can only be deployed on powerful computers with massive GPU accelerators.
- How to deploy a low-cost model on a smartphone such that it achieves the same accuracy as the complex model on the Cloud?
- Must limit communication to Cloud to reduce latency

HPC: High Prediction Cost model
LPC: Low Prediction Cost model

Most examples are predicted locally on LPC. Only a small fraction of “hard” examples are gated to the HPC.

In summary, existing complex models (e.g., kernel SVMs, boosted trees, random forests, deep neural networks):
- Pros:
  - Highly accurate for specific tasks
  - Requires to extract all (or nearly all) features – Expensive/Slow/Impractical
  - Requires powerful CPU/GPU, large memory – Must be run on server/cloud
- Cons:
  - Requires additional features
  - Expensive

Our approach:
- Given a general complex model (HPC), jointly learn a low-cost gating (g) and prediction model (LPC) to “adaptively approximate” the HPC.
- g distinguishes “easy” vs “hard” examples
- “easy” examples are predicted using LPC
- “hard” examples are sent to HPC
- Overall accuracy matches that of HPC
- Average prediction cost (over the examples) is significantly reduced

Key Idea
- HPC in general has higher probability of correct prediction conditioned on every example x (~ Bayes classifier)
- LPC can match that of HPC only on a subset of the input space (to the right of the gating threshold)
- Examples to the left of the gating threshold should still be predicted by the HPC

Formulation

\[
\begin{align*}
\text{minimize} & ~ \sum_{f \in F} \sum_{i=1}^{N} |1 - q(f_i(x), y)| D(f_i(x), g(x)) + \Omega(f_i, g) + \sum_{i=1}^{N} f_i \quad \text{(Fraction to f_i)} \\
\text{subject to} & ~ \sum_{i=1}^{N} q(f_i(x), y) \leq P_{\text{full}} \quad \text{(Constraints)}
\end{align*}
\]

Gating approximation:
- Directly parameterizing q leads to non-convexity
- Instead, we leave q non-parametric and parameterize g to approximate q
- Ensure approximation through minimizing KL distance between g and q

\[
D_KL(q(x)||g(x)) = \sum_{x} q(x) \log(q(x)/q(x))
\]

Budget constraint:
- Low cost g and f_i by selecting F and G, as well as cost penalty \(\Omega(f_i, g)\)
- Limit \(P_{\text{full}}\) fraction of examples sent to the HPC

High Level: Alternating minimization
1) Fix q, solve for g and f_i
2) Fix g and f_i, solve for q

The second step solves the following:

\[
\begin{align*}
\text{min} & ~ \frac{1}{N} \sum_{i=1}^{N} \left[ (1 - q_i) A_i + q_i B_i - H(q_i) \right] \\
\text{s.t.} & ~ \frac{1}{N} \sum_{i=1}^{N} q_i = P_{\text{full}}
\end{align*}
\]

Where \(q_i = q(0)^{\alpha_i}\), \(A_i\), and \(B_i\) are constants w.r.t. q

Algorithms

Results

Synthetic Example: Adaptive Feature Acquisition
- Key: How to choose subset of features for LPC and gating?
- Why not just do sparse feature selection?
- Run L1-regularized logistic regression
- Identify the sparse feature subset
- Train LPC and gating based on it

Consider the 2-D experiment on the right:
- Cluster 1 and 2 have slightly more examples
- L1-based classifier selects Feature 1

Optimal choice should be Feature 2 (Adapt-Lin converges to)

Take-away: Adapt-Lin benefits from joint optimization for \(f_i\) and g

Real datasets: Adaptive Feature Acquisition

- ADAPT-GBRT out-performs other budgeted learning algorithms.
- ADAPT-GBRT out-performs ADAPT-Lin
- ADAPT-GBRT can approximate different HPC models (e.g., RBF-SVM, RF, GBRT)