Project Proposals

Xiang Zhang

Department of Computer Science
Courant Institute of Mathematical Sciences
New York University

March 26, 2013

Contents

What are there?

1. A General Prallelized Trainer for Torch
2. Fast Ada-boost Using Heuristic Decision Trees
3. Parallelization of kernel SVM

A General Prallelized Machine Learning Trainer for Torch

Utilize Multiple Machines by Distributing the Datasets

- Goal: Write a general trainer within the neural network framework of torch, that can utilize multiple machines to finish a single training task.
- Means: Alaternating Direction Method of Multipliers
  - Resource: *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*, S. Boyd et al, chapter 7, *Consensus Optimization*.
  - `http://www.stanford.edu/~boyd/papers/admm_distr_stats.html`
- Techniques
  - The torch provides the 'parallel' package which can be used to fork processes on other machines using the zmq library.
  - You can choose to implement it in SPMD (single-process, multiple-data) or Master-Slave paradigms.
  - The libraries *opt* and *xtrain* can be good starting code. (The later was originally programmed for your first assignment, but the protocols can be similarly used).
- Apply to non-linear hypotheses and non-convex problems!

- The optimization problem is tranformed to

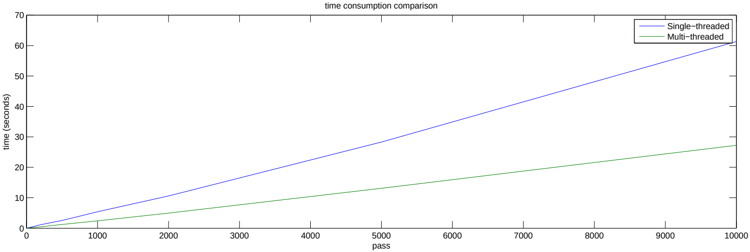
$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} f_i(x_i) \\ \text{subject to} \quad & x_i - z = 0, \quad i = 1, \dots, N \end{aligned}$$

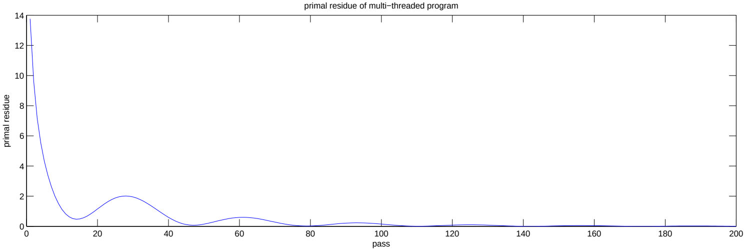
- Using augmented Lagrangian, the decomposed dual ascent algorithm is

$$\begin{aligned} x_i(t+1) &\leftarrow \operatorname*{argmin}_{x_i} \left(f_i(x_i) + y_i(t)^T (x_i - z(t)) + (\rho/2) \|x_i - z(t)\|_2^2 \right) \\ z(t+1) &\leftarrow \tfrac{1}{N} \textstyle\sum_{i=1}^{N} (x_i(t+1) + (1/\rho) y_i(t)) \\ y_i(t+1) &\leftarrow y_i(t) + \rho(x_i(t+1) - z(t+1)) \end{aligned}$$

A General Parallelized Trainer for Torch

Some Results on Linear Regression Using a Multi-thread Approach





Fast Ada-boost Using Heuristic Decision Trees

The Approach Has to Have a Theoretical Foundation

- Goal: Implement a fast ada-boost library taking advantage of the weak learning guarantee, using heuristic decision trees that does not find the best tree at every step.
- The weak learning guarantee of Adaboost:

Theorem (Weak Learning Guarantee of Adaboost)

The empirical error of the classifier returned by Adaboost verifies the following if for all $t \in [1, T]$*,* $\gamma \leq (1/2 - \epsilon_t)$*:* $\hat{R}(h) \leq \exp(-2\gamma^2 T)$*.* ϵ_t *is the empirical error of the weak classifier at step* t*, calculating according to the reweights on the samples at that step.*

Fast Ada-boost Using Heuristic Decision Trees

Can We Compete with Vowpal Wabbit?

- The heuristic knowledge: each feature of the data has limited precision far worse than the precision of the numerical capability of digital computers. We can do preprocessing on the dataset, for each feature:
  1. Cluster together adjacent data points who have the same labels. The decision tree does not have to set a node separating in the middle of a cluster. This is feasible by weak learning guarantee.
  2. Cluster together adjacent data points who has the same values. To compute ϵ_t, we need to store the number of negative labels and positive labels for the clustered value.
- After this, you have to initialize the initial weights for boosting in a clever way, rather than just initialize them as $1/|S|$.
- Resource (in Python 3): `https://github.com/zhangxiangxiao/XBoost`
- Let's hope we cam compete with Vowpal Wabbit! (but then you have to implement this in C and using hashing on the features)

Parallelization of kernel SVM

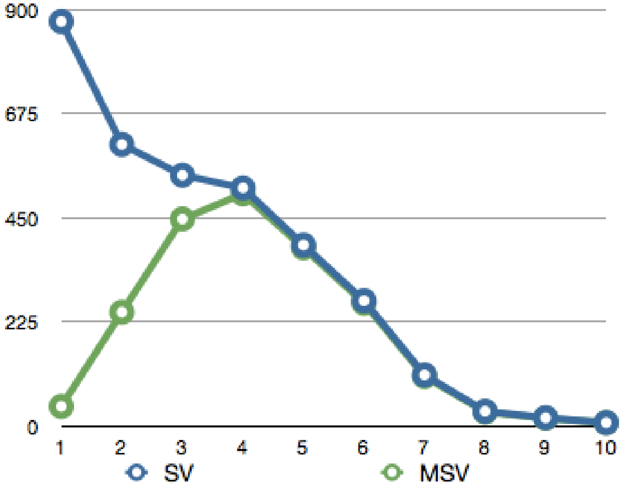
Parallelization in the Dual Space

- Goal: Parallelize a kernel support vector machines algorithm, such as sequential minimal optimization (SMO).
- Difficulty: Unlike primal gradient descent, the kernel SVM algorithms update variables in the dual space by utilizing its sparsity. Thus, synchronization of two separately trained kernel SVMs must also preserve such sparsity to ensure the next step will work fast enough.
- Startup code: https://github.com/zhangxiangxiao/XSVM
- Mathematics: derive something on your own! ADMM may be a good start, but you should need a more clever way of updating the consensus variable rather than just decayed averaging.

Parallelization of kernel SVM

Parallelization in the Dual Space

- The computational complexity is proportional to the suqare of the number of marginal support vectors.


SV
MSV