Applying Deep Neural Networks to HEP Job Classification

Lu Wang (Lu.Wang@ihep.ac.cn)
Institute of High Energy Physics, Chinese Academy of Sciences

Introduction

The cluster of CC-IHEP is a middle sized computing system which provides ~10 thousands CPU cores, 3 PB disk storage, and 40 GB/s IO throughput. Its 1000+ users come from serials of HEP experiments. On such a cluster, *Job Classification* is an indispensable task:

- Different job types may hit different performance pitfalls of software and may have variant requirements or dependencies on CPU, memory and bandwidth.

Cluster users are not obligated to provide type information of their jobs.

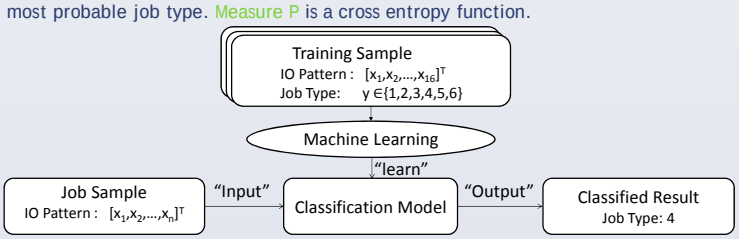
Experienced administrators can estimate a job's type from its IO pattern.

- For example, simulation jobs tend to have a *few large read and many small writes*, while analysis jobs may have *more* irregular *seeks and fewer writes*.

However, manual classification for millions of jobs is unpractical. This paper demonstrates how to solve this problem by a machine learning algorithm called Deep Neural Network (DNN).

Machine learning is " a computer program which learns from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E." Mitchell, T. (1997). *Machine Learning*, McGraw Hill. ISBN 0-07-042807-7, p.2.

In our case, task T is "to classify a job into one of six pre-defined job categories by its IO pattern"; Experience E is "the labeled training samples which have both IO pattern and type information"; The result of machine learning is a classification model. The model is a set of linear and nonlinear functions which maps a job's IO pattern to the most probable job type. Measure P is a cross entropy function.



Deep Neural Network is a non-linear machine learning algorithm inspired by the brain. It has properties of:

a) scalability to large datasets and large feature space,

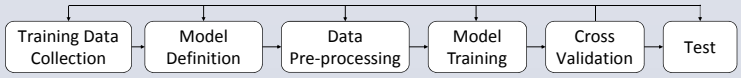
b) support to on-line training,

c) capability of feature learning etc.,

Recently, it has achieved better performance than other state-of-art algorithms in domains of natural language processing, computer vision, speech recognition and data mining.

Implementation

The implementation involves six steps showed in following diagram. According to conditions happen at cross validation stage, the procedure will be iterated many times to achieve ideal performance.



1.Training Data Collection

In this case, one training sample includes a IO pattern and a Job Type.

IO pattern counts different file operations in a certain time window(showed in the bottom left figure). It is collected by an in-kernel agent.

- For Lustre file system, the agent parses two /proc files to get the IO pattern of a specified process.
  - /proc/fs/lustre/llite/*/extents_stats_per_process and
  - /proc/fs/lustre/llite/*/offset_stats
- For other file systems, the agent inserts an SystemTap module which filters and counts IO operations for a specified process on VFS layer.

A job type is tagged in an semi-automatic way. Firstly, a key word array is provided by administrator for each job type(showed in the bottom right figure). Then, a job's type is tagged by keyword matching. While matching a job type, number prefixes and suffixes are skipped. Jobs which hit >1 keywords or 0 keyword are discard.

How an IO pattern Looks like ?

SR	MR	LR	XLR	SW	MW	LW	XLW
SRS	MRS	LRS	XLRS	SWS	MWS	LWS	XLWS

- S: small sized operation, [0-4KB]
- M: middle sized operation, [4KB,1M]
- L: large sized operation, [1M,4M]
- XL: extremely large sized operation [>4M]

R: read RS: seek before read W: Write WS: seek before write

How key words of simulation job looks like?

```
my @sim=(" sim ", " Sim ", " SIM ", " simch ", " simcheck ", " simMcPhiEta ",
"simPhiEta ", " simu ", " simulation ", " bgsim ", " DDstarbarsim ", " testMC ",
" zerowidthMC ","DbarDstarsim ","CocktailMC", "MCggphi", "McJpsi",
"psiMC", "psipMC", "SMALLJPSIPSMC", "ddbarmc", "DexMC", "dimumc",
"DipigamInclusiveMC", "exclusiveMC","gammapsipmc", "incmc", "incMC",
"qqbarmc", "Incmc", "IncMc", "jobMC", "KapiAlgMC", "kkmc", "KKMC",
"mc", "MC", "sigmc", "sigMC", "SigMC", "signalmc", "signalMC", "unSkimMC
");
```

There are six main job types on the cluster: "analysis", "reconstruction", "simulation", "calibration", "skim" and "scan". They are represented by a integer ranging from 1 to 6.

On the first half 2014, we collected about 3,200,000 job samples. 10% of them were selected to be used in the target DNN model training.

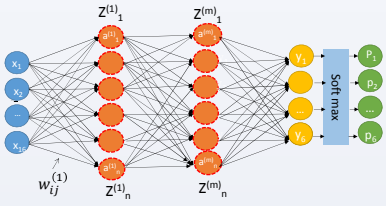
2. Model definition

- Linear function(x -> z) $z = b + \sum_{i=1} w_i x_i$
- Tanh function (z -> a) $a = \frac{1-e^{z}}{1+e^{z}}$
- Softmax function (y->p) $p_i = \frac{e^{y_i}}{\sum_{jegroup} e^{y_j}}$
- Cost function (Optimization target, measure P)

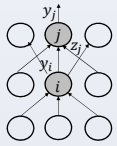
$$\min_{w_{ij}^{(k)}} C = -t_i \log p_i$$

Target value or the class label

- $w_{ij}^{(k)}$ corresponding to every black line are the parameters of DNN.



Back Propagation



$$\frac{\partial C}{\partial z_j} = \frac{dy_j}{dz_j}\frac{\partial C}{\partial y_j} = y_j(1-y_j)\frac{\partial C}{\partial y_j}$$

$$\frac{\partial C}{\partial y_i} = \sum_j \frac{dz_j}{dy_i}\frac{\partial C}{\partial z_j} = \sum_j w_{ij}\frac{\partial C}{\partial z_j}$$

$$\frac{\partial C}{\partial w_{ij}} = \frac{\partial z_j}{\partial w_{ij}}\frac{\partial C}{\partial z_j} = y_i\frac{\partial C}{\partial z_j}$$

BP computes derivatives of C over every w_{ij} simultaneously .

3. Data Preprocessing

- Organize samples into a matrix [$X_1, X_2, \ldots X_{16}$, t],
- Make a row-reshuffle of the matrix,
- Take square-root of X_i, $X_i = X_i : sqrt()$
- Nominalization of X_i, $Mean_i = X_i : mean()$,

$$Std_i = X_i : std(), X_i = \frac{X_i - Mean_i}{Std_i}$$

- Divide the matrix into three parts:
  - training set (60%), cross-validation set (20%), test set (20%)

4. Model Training with Mini-batched SGD

repeat until converge {

b=batch_size

for s=0: data_size, b {

$$w_j := w_j - \alpha \sum_{k=s:s+b} \left(\frac{\partial C}{\partial w_j}\right)^{(k)}$$

// k is the index of training sample

// W_j is every parameter of the model

// α is the learning rate

}}

Problem of over fitting

- The model has good performance on the training set, but perform poor on the cross-validation set.
- Reason: the model is excessively complex, such as having too many parameters relative to the number of training data.
- Solution: use less features, get more training data; decrease numbers of hidden layers and hidden layer unites ;add a penalty(called weight decay) on parameters;drop out

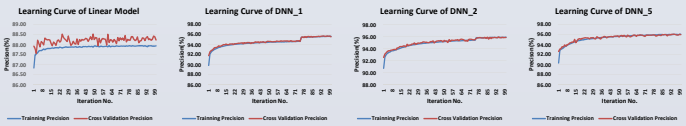
Problem of bias

- The model has poor performance both on the training set and the cross-validation set.

Reason: the model is too simple to fit the training dataset.

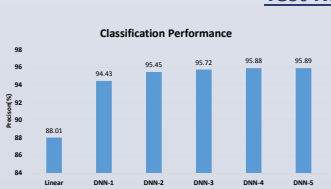
Solution: get more features ; add numbers of hidden layers and hidden layer unites ;Increase learning rate to avoid the training falling into local optimal

5. Cross Validation



- Learning curves show that all the models have converged after 100 training iterations.
- There is not a phenomenon of over-fitting , therefore, tricks of weight decay and drop out were not used .

Test Results



Meta Parameter	Value
Batch_size	1
Learning rate	1e-03
Weight Decay	0
Learning rate decay	1e-07
Momentum	0
Unites per hidden layer	32
Training Iterations	100

- The suffix number behind "DNN_" is the number of hidden layers. Results are tested on 64k samples.
- A simple linear classification model was trained for purpose of comparison.
- The result show that DNNs outperform linear model 6%+ by metric of precision. performance gain from increasing hidden lays is not so significant in this task.

The confusion matrix of DNN_5 model

	Analysis	Re-construction	Simulation	Calibration	Scan	Skim	Row Precision
Analysis	8843	50	509	0	0	2	94.034
Re-construction	50	35540	316	105	0	2	98.687
Simulation	895	567	16428	2	0	0	91.818
Calibration	0	134	2	622	0	0	82.058
Scan	0	4	0	0	124	0	96.875
Skim	6	5	2	0	0	172	92.973

- The confusion matrix shows details of classification precision, the larger number on the diagonal the better.
- Average row correct is 92.74 % , average rowUcol correct (VOC measure) is 88.26%, global correct is 95.88 %.
- The right figure shows average time consumption of one iteration training on 256k training samples. This number was got on a 24 core Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz server. Algorithms were implemented with Torch 7, a scientific computing framework with wide support for machine learning algorithms. http://torch.ch/

Future works

- Continuously update the model to adapt the changing job behaviors.
- As the dataset grows, using GPU or Map-reduce to accelerate the training speed.
- Combining DNN with other tasks of cluster administration.
  - log analysis, fault prediction, scheduling optimization ...