Master EIT Digital in Data Science

Massively Parallel Machine Learning

Practical Assessment Task (2017-18)

Parallel Implementation and Evaluation of a supervised ML algorithm

Description of tasks to be completed

1. Using the labelled dataset "spam.data" that can be downloaded from https://web.stanford.edu/~hastie/ElemStatLearn/data.html, implement a parallel version of the logistic regression classifier on Spark with Python.

   a. Implement train and predict procedures

```
def train (X, Y, iterations, learning_rate, lambda_reg):

    Arguments:
    X -- Train dataset of size (56 features, number of examples)
    Y -- Train labels (0 if no spam, 1 if spam) of size (1, number of examples)
    iterations -- number of iterations of the optimization loop
    learning_rate -- learning rate of the gradient descent
    lambda_reg -- regularization

    Returns:
    parameters -- dictionary containing the weights w and bias b


def predict(w, b, X):

    Arguments:
    w -- weights
    b -- bias
    X -- examples to be predicted: data of size (56, number of examples)

    Returns:
    Y_pred -- a vector containing all predictions (0/1) for the examples in X
```

The train procedure can be implemented using Gradient Descent. You can use this implementation but are encouraged to do research to find possible alternatives, optimizations or improvements of any kind.

```
initialize w1; w2; ... wn; b
           dw1; dw2; ... dwn; db
for it in range (iterations):
    compute dw1; dw2; ... dwn; db
    w1 = w1 − α ∗ dw1
    w2 = w2 − α ∗ dw2
    ...
    wk = wk − α ∗ dwk
    b = b − α ∗ db
```

Cost function

$$J(W) = -\frac{1}{m}\sum_{j=1}^{m}(y^{(j)}log(\hat{y}^{(j)}) + (1-y^{(j)})(1-log(\hat{y}^{(j)})) + \frac{\lambda}{2m}\sum_{i=1}^{k} w_i^2$$

Derivatives

$$dw_1 = \frac{1}{m}\sum_{j=1}^{m}(\hat{y}^{(j)} - y^{(j)}) * x_1^{(j)} + \frac{\lambda}{m}w_1$$
$$dw_2 = \frac{1}{m}\sum_{j=1}^{m}(\hat{y}^{(j)} - y^{(j)}) * x_2^{(j)} + \frac{\lambda}{m}w_2$$
$$\ldots$$
$$dw_k = \frac{1}{m}\sum_{j=1}^{m}(\hat{y}^{(j)} - y^{(j)}) * x_k^{(j)} + \frac{\lambda}{m}w_k$$

$$db = \frac{1}{m}\sum_{j=1}^{m}(\hat{y}^{(j)} - y^{(j)})$$

Hints:

- Normalize (or scale) the dataset before training the model
- Print the value of the cost function at the end of each gradient descent iteration to observe how the training process is converging.
- Try to parallelize computations that involve processing all the elements of the dataset.
- Use vectorized operations to speed up the computations

2. Implement a procedure to perform cross validation.
    a. The purpose of cross validation is to estimate the performance of the model and to choose the best set of hyperparameters.
    b. You can modify hyperparameter (in this context the amount of regularization, lambda) to trade off between bias and variance
    c. Consider different approaches to exploit parallelization and perform cross validation more efficiently. Don't be afraid to try anything that comes to mind and test if it works.

Report structure

Written report including the following sections: Introduction, data set description, algorithm implementation, cross validation procedure, experiments (including performance and speedup curves as well as error metrics of the machine learning tasks), and conclusions.

The performance curve shows execution time versus number of workers. The speedup curve shows the ratio (running time using 1 worker) /(running time using n workers) versus the number of workers.

The machine learning error metrics to measure are the ones seen in class for classification: cost error, precision, recall and f1-score.

Oral presentation: set of slides