Bo Lei

Your Andrew ID: bolei

Homework 5

0. Statement of Assurance

I certify that all of the material that I submit is original work that was done only by me.

1. Gradient Descent Approach (15%)

1.1 Derive the gradient of the likelihood

Derive the formula of the gradient

Objective function:

$$\hat{W}^{RLR} = \arg\max_{W} \{f(W)\}$$

$$f(W) = \ln P(D|w) - 0.5C \parallel W\parallel^2 = \sum_{i=1}^{n} \left(y_i \ln \sigma(W^T X_i) + (1-y_i)\ln(1-\sigma(W^T X_i)) - \frac{C}{2n} \parallel W\parallel^2\right)$$

For an each instance X_i, define it's log likelihood:

$$l_i(W) = y_i \ln \sigma(W^T X_i) + (1-y_i)\ln(1-\sigma(W^T X_i)) - \frac{C}{2n} \parallel W\parallel^2$$

define: $\sigma(z) = \frac{1}{1+e^{-z}}$, $z_i = W^T X_i = w_0 + \sum_{j=1}^{m} w_j x_{ij}$

Then: $\frac{\partial}{\partial w_j} l_i(W) = \frac{dl_i(W)}{d\sigma}\frac{d\sigma}{dz_i}\frac{\partial z}{\partial w_j} - \frac{C}{2n}\frac{\partial}{\partial w_j} \parallel W\parallel^2 = (y_i \frac{1}{\sigma(z_i)} - (1-y_i)\frac{1}{1-\sigma(z_i)}) * \sigma(z_i)(1-\sigma(z_i)) * \frac{\partial}{\partial w_j} z_i - \frac{C}{n} w_j$

$$= (y_i(1-\sigma(z_i)) - (1-y_i)\sigma(z_i))x_{ij} - \frac{C}{n} w_j$$

$$= (y_i - \sigma(W^T X_i))x_{ij} - \frac{C}{n} w_j$$

1.2 Gradient descent algorithm

Outline the gradient descent algorithm

Since I implemented gradient ASCENT algorithm, I'll outline my implementation here. Gradient descent algorithm is similar except that we do

weightVector -= learningRate * gradientVecto

to update the weight vector

Goal: Maximize the objective function:

$$f(W) = \ln P(D|w) - 0.5C \parallel W\parallel^2 = \sum_{i=1}^{n} \left(y_i \ln \sigma(W^T X_i) + (1 - y_i) \ln(1 - \sigma(W^T X_i)) - \frac{C}{2n} \parallel W\parallel^2\right)$$

The algorithm:

initialize weight vector to random values

oldObj = INT_MIN

newObj = calculate objective function with weightVector

while newObj - oldObj > precision

do

    for each instance, do

        calculate gradientVactor

        weightVector += learningRate * gradientVector

    end

    oldObj = newObj

    newObj = objective function with updated weightVector

end

2. Evaluation Results (40%)

2.1 Logistic Regression

learning rate: 0.1

convergence criteria: new objective - old objective < 1E-4

C	MICRO-AVG			MACRO-AVG		
	Precision	Recall	F1	Precision	Recall	F1
0.0001	0.6564246	0.6564246	0.6564246	0.6511652	0.6445510	0.6431381
0.001	0.6438547	0.6438547	0.6438547	0.6496387	0.6269069	0.6230366
0.01	0.5851955	0.5851955	0.5851955	0.6345580	0.5614255	0.5564242
0.1	0.3833799	0.3833799	0.3833799	0.5408466	0.3623594	0.3275472
1	0.1466480	0.1466480	0.1466480	0.2308998	0.1370701	0.0788264
10	0.0649441	0.0649441	0.0649441	0.0066365	0.0592867	0.0079962
50	0.0886872	0.0886872	0.0886872	0.0052169	0.0588235	0.0095838
100	0.0886872	0.0886872	0.0886872	0.0052169	0.0588235	0.0095838

learning rate: 0.01

convergence criteria: new objective - old objective < 1E-4

C	MICRO-AVG			MACRO-AVG		
	Precision	Recall	F1	Precision	Recall	F1
0.1	0.5425978	0.5425978	0.5425978	0.6523928	0.5150835	0.5132660
1	0.3826816	0.3826816	0.3826816	0.5424660	0.3614863	0.3266481
10	0.146648	0.146648	0.146648	0.230976	0.137070	0.078917

	0	0	0	6	1	0
50	0.0914804	0.0914804	0.0914804	0.0369337	0.0888576	0.0334539
100	0.0649441	0.0649441	0.0649441	0.0066365	0.0592867	0.0079962

learning rate: 0.001

convergence criteria: new objective - old objective < 1E-4

C	**MICRO-AVG**			**MACRO-AVG**		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
10	0.3826816	0.3826816	0.3826816	0.5426292	0.3614863	0.3266246
50	0.1766760	0.1766760	0.1766760	0.2864786	0.1642132	0.1227760
100	0.1459497	0.1459497	0.1459497	0.2309643	0.1362858	0.0786591

2.2 SVM

C	**MICRO-AVG**			**MACRO-AVG**		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
0.0001	0.5432961	0.5432961	0.5432961	0.6206298	0.5422486	0.5369033
0.001	0.6138268	0.6138268	0.6138268	0.5992586	0.6050103	0.5892401
0.01	0.6236034	0.6236034	0.6236034	0.6129197	0.6134529	0.5985171

0.1	0.6256983	0.6256983	0.6256983	0.6136163	0.6161320	0.6010657
1	0.6354749	0.6354749	0.6354749	0.6222243	0.6247685	0.6144372
10	0.6194134	0.6194134	0.6194134	0.6161295	0.6122009	0.6108887
50	0.6173184	0.6173184	0.6173184	0.6144542	0.6108113	0.6094651
100	0.6173184	0.6173184	0.6173184	0.6144542	0.6108113	0.6094651

2.3 Training time

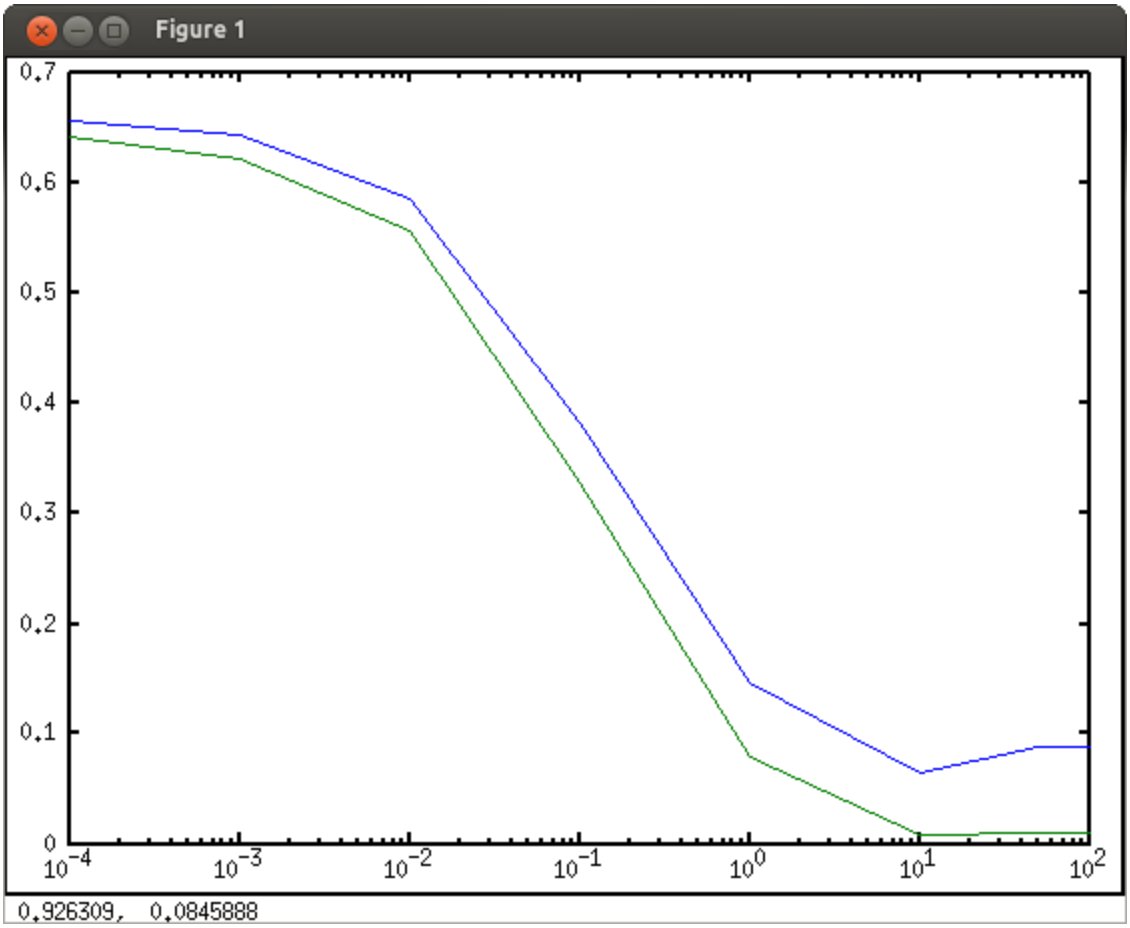
C	LR Time (s)	SVM Time (s)
0.0001	3924.646	4.880
0.001	563.028	3.087
0.01	85.509	3.410
0.1	49.300	3.680
1	176.486	3.937
10	42.292	4.585
50	17.338	4.562
100	17.419	4.510

2.4 Logistic Regression graph

Plot a graph of the Micro-F1, Macro-F1 for different values of “C” for Logistic Regression.

learning rate: 0.1

convergence criteria: new objective - old objective < 1E-4

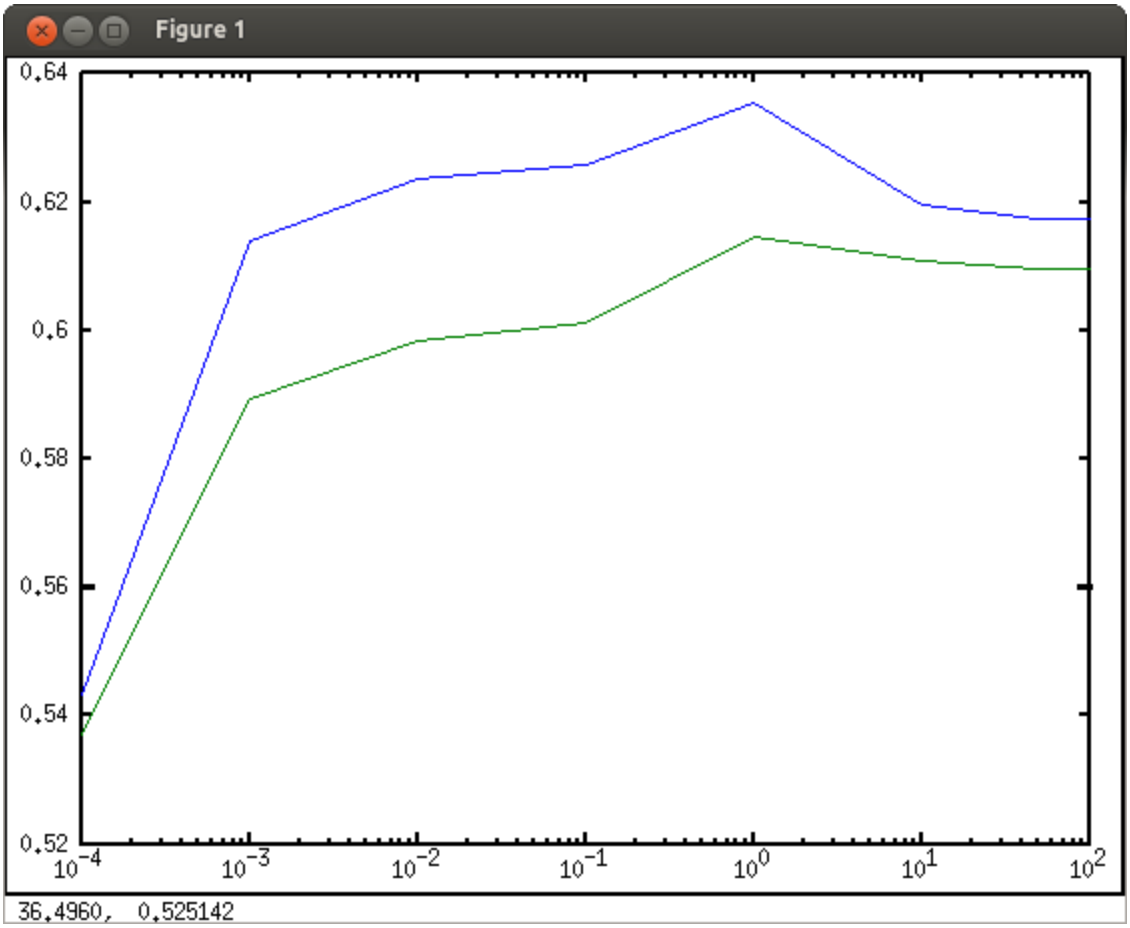


Green: Micro-F1

Blue: Macro-F1

2.5 SVM graph

Plot a graph of the Micro-F1, Macro-F1 for different values of “C” for SVM.



Green: Micro-F1

Blue: Macro-F1

3. Analysis of results (30%)

Compare the SVM and Logistic regression classifiers. Discuss specifically about the performance of each and the time taken to train.

From the experiments, it is shown that the best Logistic Regression result is similar to SVM result. Both of them achieve over 60% accuracy in multi-class classification.

For Logistic regression, if the learning rate if fixed, the coefficient of L2 norm C plays an important role in the prediction accuracy. As shown in section 2.4, when C increases, the accuracy drops dramatically. In addition, C also affects the convergence speed as shown in section 2.3. The algorithm converges faster with a bigger C. To compensate the effect of C change, I also tried to use smaller learning rate when C is big (see the additional charts in section 2.1). It turns out that the accuracy is improved with smaller learning rate. In my opinion it means when C increases, the step of each iteration in gradient ascent also increases. Because L2 norm comes from using MAP as the

objective function, and $C = \frac{1}{2\sigma^2}$ where σ^2 is the variance of the W distribution, I think the best C should be calculated by taking the sample (training data) variance and calculate $\frac{1}{2\sigma^2}$ if the sample variance is a good estimate of true variance of W.

For SVM, the change of C doesn't affect the training time that much. It doesn't affect the prediction accuracy either. Because a bigger C indicates that the model tolerates more training error and results in smaller margin, it is a tradeoff between the two factors. Notice that in the experiment, when C=1, the model gives best result.

4. The software implementation (15%)

Description of your code including any data structures used, design considerations etc.

My program can be divided into 2 parts. Logistic Regression and SVM. They have different entry points. However they share the same framework that automatically trains and test the file (AbstractModel.java).

Upon each run, the program will use a helper program TrainingDataPreprocessor to make 17 copies of the training file (each corresponds to a class) and put them in a certain folder. Then AbstractModel automate the entire training and testing process:

AbstractModel holds a reference to a wrapper of classifier (Logistic Regression or SVM). It first uses AbstractOneVsRestClassifierAdapter.trainOneVsRest method to iteratively reads training file and train a specific classifier for each class. The model for this class is written into a file and stored in a predefined location. Training process on each specific class is implemented by corresponding algorithms (LogRegAdapter.train and SvmAdapter.train). training time is measured and printed into standard error.

Then AbstractOneVsRestClassifierAdapter.testOneVsRest is then called. It loads the the corresponding model file for this class into memory and uses the specific classifier to make predictions for this class based on the model it previously loaded. Predictions for different classes are written into different files and are again stored into a predefined location. After prediction of all the classes has been done, AbstractOneVsRestClassifierAdapter will make the final prediction based on One-VS-Rest approach. This final prediction is printed out to standard out.