MEHTHAB 20161205

SMAI MINI Project 2:

Practical issues in building the classifiers:

Feature - scaling & Normalization - PCA and most algorithms work well when the data is ‘similar’ in magnitude therefore data scaling is needed

Number of iterations for convergence of the optimal can be very large and hence time-taking so usually a little gain in accuracy can be sacrificed for faster running if it is not too sensitive/critical classification task

Algorithms that work better on large datasets may not work that well on smaller ones. A lot of hyper-parameter tuning need to be done for very good accuracies.

Classifiers can be combined to give better results if the two separately are not very good.

When the number of samples is far lesser than number of features classifiers may be overfit on the train data i.e., accuracy on train set is very high but accuracy on validation/test set is very low. This can be handled using regularization with the objective function , soft-margin in case of svm and adding more data.

There is also the case of outliers and noisy data this again can be handled using regularization and pre-conditioning the data.(feature selection and normalization like pca also helps)

Overfitting can be handled by -

1. Add more data.
2. Use data augmentation.
3. Use architectures that generalize well.
4. Add regularization (mostly dropout, L1/L2 regularization are also possible)
5. Reduce architecture complexity. (in the below mlp it uses only two hidden layers with 50 neurons each

Hyper-parameter tuning to improve results.
Here results from previous experiments models can be used to choose the parameters.

Results :-

(X) - X = number of batches used for that run
N - normalized data (after scaling)

1. RandomForest classifier :

   Hyper perameters used - number of estimators,max_depth of an estimator,minimum number of splits per level ,min_samples_leaf.

   Number_of _estimators = number of trees:
   For the cifar-10 150 - 200 estimators have given the best results at minimum number of splits 3 minimum sample leaves = 2 and max depth 8-10
   As 10 classes are there the depth 10 almost shows a binary -directed graph like classification happening with this setting of parameters.

   Max_depth = max_depth of each tree. This should be controlled as otherwise for large datasets the tree size can become very large hence longer time and larger space for classification.

   Data was scaled but not normalized for this classifier .Normalized data has been observed to produce better results.

   Representations :pca
   Hyperparameters for pca - no of components

   Best result : accuracy = 28.26 , fscore = 28.26 on first batch only

   180,8,3,2
   600

   First 3 batches best result :
   Acc = 29.53  ,fscore = 29.53

   200,8,3,2
   600

   With 180,8,3,2,600 :acc,fscore = 23.85,0.2385

   All five batches - 200,8,3,2,600 - caa,fscore = 31.54,0.3154

Other results :-
(5) 200,10,3,2,600 acc,f = 23.75,0.2375
(1)150 ,6,3,2,800 ac,f = 20.75,0.2075
(1)100,6,3,2,400 ac,f = 13.65 , 0.1365
(1)150,6,3,2,800 ac,f = 16.75,0.1675
(1)150,8,3,2,800 ac,f = 22.57 , 0.2257
(1)150,6,3,2,800 ac,f = 21,xx ,0.21xx
(1)150,8,3,2,600 ac,f  = 26.78,0.2678

Raw data -
Same hyperparameters for RF:

Best result:
(1)200,20,3,2 ac,f = 42.43,0.4243
(3)200,20,3,2 ac,f = 46.25,0.4625
(5)200,20,3,2 ac,f = 47.21,0.4721
Other results:
(1)200,8,3,2 ac,f = 38.61,0.3861
(1)100,10,3,2 = 39.87,0.3987
(1)100,15,3,2 = 40.91,0.4091
(1)100,15,2,2 = 41.17,0.4117
(1)50,15,2,2 = 39.71,0.3971

Lda :-

(1)N100,20,3,2 = 50.12,0.5012
(1)N80,20,3,2 = 45.9,0.459
(1)N180,20,3,2 = 37.54,0.3754
(1)200,20,3,2 - 23.07, 0.2306

Logistic Regression :
Hyper - C

C - regularization constant .Lower values of C imply stronger regularization - hence better fit on test set but more iterations required for converging.Some of the below runc dont converge even with max_iterations set to 1000.

Solvers - lbfgs ,sag and their respective paramters were tuned.

The ‘newton-cg’, ‘sag’, and ‘lbfgs’ solvers support only L2 regularization with primal formulation. The ‘liblinear’ solver supports both L1 and L2 regularization, with a dual formulation only for the L2 penalty.

Algorithm to use in the optimization problem.

- For small datasets, ‘liblinear’ is a good choice, whereas ‘sag’ and ‘saga’ are faster for large ones.
- For multiclass problems, only ‘newton-cg’, ‘sag’, ‘saga’ and ‘lbfgs’ handle multinomial loss; ‘liblinear’ is limited to one-versus-rest schemes.
- ‘newton-cg’, ‘lbfgs’ and ‘sag’ only handle L2 penalty, whereas ‘liblinear’ and ‘saga’ handle L1 penalty.

**multi_class** : str, {‘ovr’, ‘multinomial’, ‘auto’}, default: ‘ovr’
If the option chosen is ‘ovr’, then a binary problem is fit for each label. For ‘multinomial’ the loss minimised is the multinomial loss fit across the entire probability distribution, *even when the data is binary*. ‘multinomial’ is unavailable when solver=’liblinear’. ‘auto’ selects ‘ovr’ if the data is binary, or if solver=’liblinear’, and otherwise selects ‘multinomial’.

Raw data:
(1)lbfgs,1.4 ac,f = 31.74, 0.3174
(1)lbf,0.8 = 30.98,0.3098
(1)lbf,2 ac,f = 31.71,31.71
(1)lbf,5 31.55,31.55
(1)sag,5 ac,f = 32.61,0.3261
(1)sag,1.4 32,77
(1)sag 0.8 33.15

PCA data
(1)lbfgs 1 ac,f = 25.45,0.2545  max_iter = 1000
(1)lbfgs 0.8 ac,f = 27.86 0.2786 max_iter = 1000
(1) sag 1 ac,f = 30.77 0.3077 max_iter = 1000
(1) sag 0.8 ac,f = 31.33 0.3133 max_iter = 1000
(1)sag 1.4 ac,f =28.56,0.2856 max_iter = 1000
(1)lbfgs 1.4 ac,f = 28.56 ,0.2856 max_iter = 1000

Takes a lot of time to converge

LDA data :max_iter = 1000
(1) lbfgs 1 ac,f = 29.45,0.2945
(1)lbfgs 2 ac,f = 27.86 0.2786
(1)lbfgs 0.8 ac,f = 31.77 0.3177
(1)sag 1 ac,f = 31.03 ,0.3103
(1)sag 2 ac,f = 28.66 ,0.2966

(1)sag 0.8 ac,f = 32.33,0.3233

MLP :
Parameters changed - activation= 'relu',solver='adam',
alpha=1e-5,hidden_layer_sizes=(50,50),momentum = 0.9,
random_state=1,max_iter=200,nesterovs_momentum = True,learning_rate_init =
0.001,learning_rate = 'constant'

On one training batch lbfgs has beem observed to give better results than adam
But with five bathces adam gave better results but the difference was not much.sgd gave the best results, with alpha = 0.00001 and learning_rate = ‘adaptive’

The hidden layer structure used was 2 layers with the number of neurons in each given by the tuple (num_firstlayer,num_secondlayer)
No of pca = 600 - 800 gave good results
N - normalized data (after scaling)

Hyperparameters : hidden_layer_sizes ( )-tuple
Raw -
(1)50 50 ac,f = 38.59,0.3859
(1)40 10 34.59
(1)50 60 ac,f = 37.89
(1)5 2 ac,f =26.07
(1)100 40 =39.78
(1)100 60 = 37.38
(1)120 40 39.99

PCA -
(1)120 40 ac,f 18.99 600 adam
(1)50 50 ac,f = 19 800 adam
(1)100 40 ac,f = 20.77 lbfgs
(1) sgd alpha = 0.0001 , 120 60 ac,f = 26.77
(1)sgd alpha = 0.00001 ,100 40 ac,f = 30.04 N(normalization of data) max_iter = 200
(1)sgd alpha = 0.00001 ,100 40 ac,f = 33.51 ,0.3351 N ,max_iter = 1000,learning_rate = adaptive
(1)sgd alpha = 0.00001, 100 40 ac,f = 2887 N,max_iter = 200, learning_rate = adaptive
(1)sgd alpha = 0.00001 100 40 2956 N ,200.constant
(1)sgd alpha = 0.00001 100 40 N,1000 invscaling power_t = 0.7/0.6/0.5 0.0936
(1)sgd 0.00001 100 40 N 200 ac,f = 31.37
(1)sgd 0.00001 120 60 N 200 ac,f = 25.67
(1)sgd 0.00001 120 120 N 200 ac,f =24.67
(1)sgd 0.00001 50 50 N 200 ac,f = 26.07
(1)sgd 0.00001 80 40 N ac,f = 28.87

(1)sgd 0.00001 100 30 N ac,f = 25.98
(1)70 30 32.64

LDA:sgd 0.00001 N adaptive no_of components = 9(max no_classes -1)

(1)sgd 0.00001 100 40 N ac,f = 34.69 adaptive
(1)50 50 ac,f = 32.51
(1)30 50 ac,f = 31.34
(1)70 30 ac,f = 49.95
(1)70 30 0.0001 38.85
(1)70 30 0.000001  24.23
(1)100 40 0.000001 27.83
(1)50 50 0.000001 27.82

Kernel SVM:

Parameters - kernel ,C (regularization constant),gamma
Kernel = ‘rbf’
C = 1.0 , gamma = default

The C parameter trades off correct classification of training examples against maximization of the decision function’s margin. For larger values of C, a smaller margin will be accepted if the decision function is better at classifying all training points correctly. A lower C will encourage a larger margin, therefore a simpler decision function, at the cost of training accuracy. In other words``C`` behaves as a regularization parameter in the SVM.

The behavior of the model is very sensitive to the gamma parameter. If gamma is too large, the radius of the area of influence of the support vectors only includes the support vector itself and no amount of regularization with C will be able to prevent overfitting.

Raw:
N - normalization of data
WN - no normalization of data
(1)default,default ac,f = 37.95 N
(1)def,def , ac,f = 10.2 ac,f(train) = 99.99 WN
(1)81.275 on training c=1.5 no convergence

LDA:-
(1)Rbf ac,f = 28 N train - 87 : case of overfitting C = 1,gamma = ‘auto-deprecated’

25.21 C = 10

C = 0.5 ,gamma = auto 27.2
LDA :
Rbf C = 10 ,gamma = 0.1 N take care of overfitting ac,f = 40.07, 40.07
Hyperperametertuning ,adding more batches (now 3 batches ) are used take care of over-fitting

RAW - after overfitting is handled -
Rbf , C = 10 gamma = 0.1 ac,f = 43.96

PCA - C = 10 gamma = 0.1 rbf
Ac,f = 35.32, 35.32

C = 100 gamma = 0.1 ac,f = 29.16
C = 10 gamma = 0.01 ac,f = 27.5

Best accuracies :-

| Representation | Classifier | Accuracy | f1_score |
|---|---|---|---|
| RAW | RF | 47.21 | 47.21 |
| LDA | RF | 45.9 | 45.9 |
| PCA | RF | 31.54 | 31.54 |
| RAW | LR | 33.15 | 33.15 |
| LDA | LR | 32.33 | 32.33 |
| PCA | LR | 31.33 | 31.33 |
| RAW | MLP | 39.99 | 39.99 |
| LDA | MLP | 49.95 | 49.95 |
| PCA | MLP | 33.51 | 33.51 |
| RAW | KSVM | 43.96 | 43.96 |
| LDA | KSVM | 40.06 | 40.06 |
| PCA | KSVM | 35.32 | 35.32 |

Over-Fitting :-

Over -fitting was observed in the case of kernel svm.
On scaled but un-normalized data it gave a 99% accuracy on training set but 12% on validation set and 10% on training set.
Normalizing the data and addition of data (3 batches total) this was reduced to 87% on train set and 10% on test set .
Hypertuning the parameter C and gamma finally took care of over-fitting.
C = 10 and gamm = 0.1 gave the best results.