Machine Learning Final Assignment: E corp Challenge(1609 words)

Ang Li @netid:shawnli @ID:4593820

1. Feature selection

In this assignment, the number of features is larger than the size of training set. Theoretically feature selection methods could be helpful in automatically select those features in your data that contributes most to the prediction variable or output. Tree-based estimators[1] can be used to compute feature importances, which in turn can be used to discard irrelevant features. After the selection, 89 features remain in the dataset, yielding the results. In this classification problem, different classifiers all have the potential to have a good performance, so different classifiers are investigated and compared by training with the 200 labeled samples and testing on the 20,000 unlabeled samples. Due to the form and size of the dataset and classes, six popular classifiers are chosen by previous experience, including k-nearest neighbors algorithm(KNN)[2], logistic regression(LR)[3] , Random forest(RF)[4] , Decision tree(DT)[5], Support Vector Machine(SVM)[6], and gradient boosting classifier(GBDT)[7] The compression of these classifiers after feature selection can be found in the following table, showing that in most cases feature selection has negative effect on the results. Therefore, the feature reduction technique it not very suitable in this problem. Similar results can be found with implementing dimension reduction techniques including Principle Component Analysis(PCA)[8] and Linear Discriminative Analysis(LDA)[9].

Table 1.1 Training and test errors of different classifiers after feature selection

	Changing in Test error	Changing in Training error
DT	-1E-05	0
GBDT	0.0158	0
KNN	0.00835	-0.09
LR	0.08665	0
RF	0.01555	0.01
SVM	-0.00035	0

2. Choosing classifiers

In this classification problem, different classifiers all have the potential to have a good performance, so different classifiers are investigated and compared by training with the 200 labeled samples and testing on the 20,000 unlabeled samples. Due to the form and size of the dataset and classes, six popular classifiers are chosen by

previous experience as shown in the previous chapter. The compression of these classifiers can be found in the following table. From the results we can observe that all of these classifiers suffer from severe overfitting. The possible reasons could be the fact that the dimensions are larger than the training sets, the testset is corrupted by a uniformly distributed noise from p-norm ball, and the number of training samples are insufficient. Also, it can be observed that the logistic regressions perform better than the other classifiers, due to its smoothing nature which alleviating the effect of noises.

Table 2.1 Training and test errors of different classifiers

	Training error	Test error
DT	0	0.48865
GBDT	0	0.4669
KNN	0.26	0.46955
LR	0	0.39165
RF	0.015	0.4826
SVM	0	0.47865

3. Noise source identification

To counter the negative effect of noises, the noises will be actively added to the features of training samples to generalize the model[10]. The uniform sample generation in p-norm balls can be generated by the following algorithm[11]:

1) generate n independent random real scalars $\varepsilon_i = \bar{G}(\frac{1}{p}, p)$, where $\bar{G}(\mu,\sigma^2)$ is the Generalized Gaussian distribution (with a different power in the exponent $e^{-|x|^p}$ instead of just p=2)
2) construct the vector x of components $s_i * \varepsilon_i$, where s_i are independent random signs
3) Generate $z = w^{1/n}$, where ww is a random variable uniformly distributed in the interval [0, 1]
4) return $y = rz\frac{x}{\|x\|_p}$

Then for each sample, a uniform noise from p-norm balls will be generated, and all the samples will be added by this noise through iteration. The algorithm for the can be found as below, where the output is the new training set with additive noises.

```
# Generate noise from p-norm ball
def noise_pnorm(R,p,n,train_data):
    train_n=np.zeros((200,204))
    for i in enumerate(train_data):
        epsilon=signal.general_gaussian(n, p, sig=p)
        epsilon[abs(epsilon)<1e-5]=0
        sign=[(-1)**np.random.randint(2) for i in range(n)]
        x=np.multiply(epsilon,sign)
        z=(np.random.uniform(0,1))**(1/n)
        x_pnorm=np.linalg.norm(x,ord=p)
        y=R*z*x/x_pnorm
        train_n[i[0]]=train_data[i[0]] + y
    return train_n
```

To identify the source of the noise, including the norm p and radius R, here we obtained the P and R by transforming this into a optimization problem. In this case, with the input P and R, as well as noisy training set with 200 samples, we can minimize the error rate on the testset with 20,000 samples by using a classifier. Mathematically, it can be written as a unconstrainted optimization problem,

$$\min \text{costfunc}(\mathrm{R}, \mathrm{p})$$
$$s.t. \begin{matrix} R \geq 0 \\ p \geq 0 \end{matrix}$$

where Rmax and pmax are prescribed upper boundaries based on the characteristic of the feature space. Costfunc is the cost function created by mapping the inputs onto the output by using random forest classifier since it is computationally efficient, and the cost function considers the effect of noises.

```
#noise optimization
def costfunc(R,p,n,train_data,train_label_true,test_data,test_label_true):
    train_data=noise_pnorm(R,p,n,train_data)
    er_test,er_train=random_forest_classifier(train_data,train_label_true,test_data,test_label_true)
    return er_test
```

Two optimization algorithms are utilized to obtain the optimal R,p. The first is random searching algorithm[12], and the second is the particle swarm optimization(PSO) algorithm[13]. Since the former is computationally inefficient and requires upper boundary, the latter is performed in this study. The pseudo code of PSO can be found as the following figure, where the number of particles are set as 30, searching dimensions as 2, and the maximum iterations as 100, initialization parameters as w = 0.8, c1 = 2, c2 = 2, r1= 0.6, and r2=0.3. The optimal set of R, p, and the best trained classifier will be outputted after convergence, leading to results shown as Figure 3.1. It can be shown that the cost function is converged, so the explicit form of the noise can be demonstrated as a uniform distribution from a 17.382-norm ball with radius=19.771. However, the convergence speed remains at a low level, and the maximum improvement of error rate is approximately 4% with maximum iterations=100.

Initialization:

Including swarm size N, maximum number of iterations n, initial velocities v and positions x, c_1 and c_2

Loop:

Evaluation:

Evaluate each particle's position according to the fitness function

Termination criterion check:

If a criterion is met, exit loop

Find the personal best:

Find the best solution of each particle so far (If a particle's current position is better than its previous best position, update it)

Find the global best:

Find the best solution of all the particles until now (Determine the best particle according to the particle's previous best positions)

Update the velocities:

Update the velocity of each particle

Update positions:

Move particles to their new positions

Return to loop.

Figure 3.1 PSO pseudo code

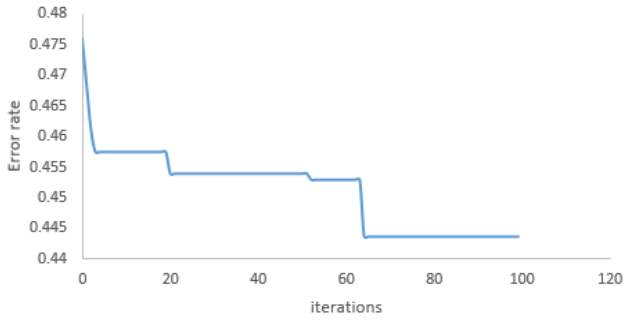


Figure 3.2 Cost function vs number of iterations

4. Semi-supervised learning algorithm

Since the unlabeled data is much more than labeled data, it is possible to use semi-supervised learning[14] to generalize the classifier and learn the features of noise distribution in the unlabeled dataset. The general steps of the semi-supervised learning algorithm in this study can be expressed as:

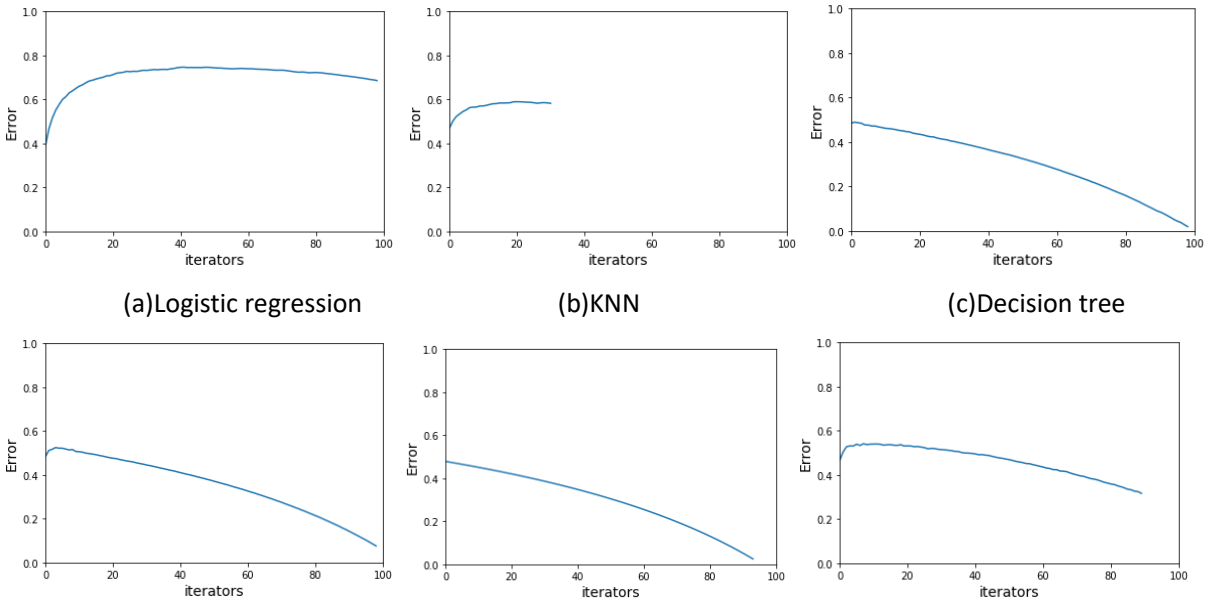
Initialize the classifier weights from labeled data

Repeat the following steps until maximum iteration reached or convergence:

1. Predict the labels of the testset, calculate the error rate and randomly pick n samples from the testset added to the training set
2. Eliminate these samples from testset
3. Train the model to update the classifier weights

In the experiment, the number of unlabeled samples to be added at each iteration is set as 100 and the

maximum iteration is set as 10. So the number of samples in training set is increased by 100, and that in test set is decreased by 100. This setting is to make sure that the final training set has approximately similar size as the test set, preventing the risk of overfitting and non-truthworthy results. The comparisons of learning curves of six different classifiers can be expressed in the following figure. It can be observed that logistic regression and KNN is not converging at all for semi-supervised learning, and the GBDT algorithm converges at a low speed. Generally the SVM has the best classification error rate, while decision tree and random forest classifier are slightly lower. But the computational speed of random forest is much faster than the others algorithms. Therefore random forest, decision tree and SVM can be selected as candidates for the joint training algorithm in the following chapter.



(a)Logistic regression (b)KNN (c)Decision tree

(a)Random forest (b)SVM (c)GBDT

Figure 4.1 The comparisons of learning curves of six different classifiers

5. Semi-supervised joint training algorithm

The final algorithm developed in this case combines the active learning with adding adversarial noise obtained by PSO algorithm and the semi-supervised learning algorithm. Three different classifiers: SVM, RF, DT are selected as the candidates for the final algorithm. The pseudo code for this joint training algorithm can be expressed as follows,

Initialize parameters(pN ,dim ,max_iter) in PSO algorithm
Choose one of the classifier
  Perform optimization and add adversarial noise(R,p) to the features in training data until the loss converging
    Update the training set
  Initialize the number of n at each iteration, the maximum number of iterations iter_max, and the classifier weights

Repeat the following steps until reaching maximum iteration(iter_max) or stopping criteria(samples in training set< samples in test set):

1. Predict the labels of the testset, calculate the error rate and randomly pick n samples from the test set added to the training set
2. Eliminate n samples from test set
3. Train the model to update the classifier weights

The final results in the learning curves can be found as shown in Figure 5.1, and the final classification error rates in Table 5.1. The table listed the R and P obtained for each classifier resptively, to get a better understanding of the source of the noise. The error rate on the test set is 0.049074 with 9,400 samples(200 labeled and 9,200 unlabeled) in the training set and 10800 samples in the test set. Combining the training error, we can obtain the final prediction error on the 20,000 unlabeled samples as 0.0263. The whole comparison among the results can be found in Table 5.1. It can be concluded that the best classification error rate is 2.54% performed by decision tree classifier. Surprisingly, this classifier also has the best computational efficiency as 302 seconds in total. The results indicate that the semi-supervised learning works extremely well in this problem. It can be attributed to the fact that the noise source is fixed so the self-learning algorithm can learn the features in the unlabeled data as more data is fed in. The final stopping criteria is set as the size of training set cannot exceed the size of test size to guarantee the trustworthiness of the results instead of suffering from severe overfitting.

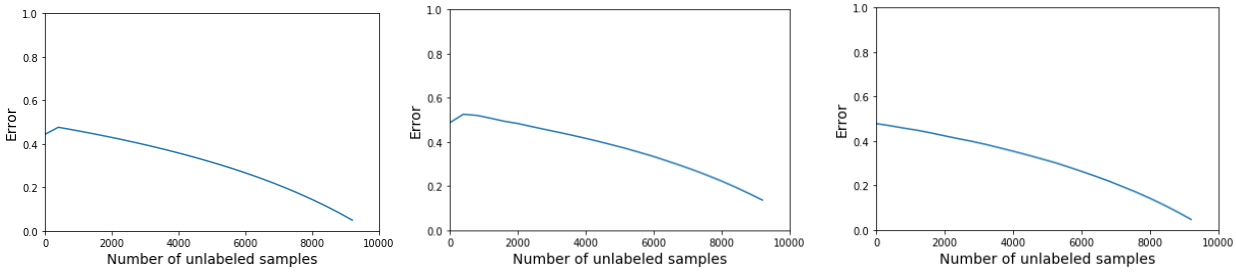


Figure 5.1 Learning curve for Semi-supervised joint training algorithm with SVM, RF and DT, respectively.

Table 5.1 The whole comparison among the results

Classifier	Noise_R	Noise_P	Error rate on training set/%	Error rate on test set/%	Error rate on 20,000 unlabeled samples	Time/s
SVM	19.771	17.382	0%	4.91%	2.63%	1925
RF	6.95701	11.8495	0%	13.66%	7.30%	343
DT	21.7705	28.9641	0%	4.76%	2.54%	302

Reference

[1] Bortolot, Z.J. and Wynne, R.H., 2005. Estimating forest biomass using small footprint LiDAR data: An individual tree-based approach that incorporates training data. *ISPRS Journal of Photogrammetry and Remote Sensing*, *59*(6), pp.342-360.

[2] Keller, J.M., Gray, M.R. and Givens, J.A., 1985. A fuzzy k-nearest neighbor algorithm. *IEEE transactions on systems, man, and cybernetics*, (4), pp.580-585.

[3] Hosmer Jr, D.W., Lemeshow, S. and Sturdivant, R.X., 2013. *Applied logistic regression* (Vol. 398). John Wiley & Sons.

[4] Liaw, A. and Wiener, M., 2002. Classification and regression by randomForest. *R news*, *2*(3), pp.18-22.

[5] Safavian, S.R. and Landgrebe, D., 1991. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, *21*(3), pp.660-674.

[6] Hearst, M.A., Dumais, S.T., Osuna, E., Platt, J. and Scholkopf, B., 1998. Support vector machines. *IEEE Intelligent Systems and their applications*, *13*(4), pp.18-28.

[7] Li, P., Wu, Q. and Burges, C.J., 2008. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in neural information processing systems* (pp. 897-904).

[8] Kim, K., 1996. Face recognition using principle component analysis. In *International Conference on Computer Vision and Pattern Recognition* (Vol. 586, p. 591).

[9] Mika, S., Ratsch, G., Weston, J., Scholkopf, B. and Mullers, K.R., 1999, August. Fisher discriminant analysis with kernels. In *Neural networks for signal processing IX, 1999. Proceedings of the 1999 IEEE signal processing society workshop.* (pp. 41-48). Ieee.

[10] Maaten, L., Chen, M., Tyree, S. and Weinberger, K., 2013, February. Learning with marginalized corrupted features. In *International Conference on Machine Learning* (pp. 410-418).

[11] Calafiore, G., Dabbene, F. and Tempo, R., 1998. Uniform sample generation in l/sub p/balls for probabilistic robustness analysis. In *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on* (Vol. 3, pp. 3335-3340). IEEE.

[12] Solis, F.J. and Wets, R.J.B., 1981. Minimization by random search techniques. *Mathematics of operations research*, *6*(1), pp.19-30.

[13] Trelea, I.C., 2003. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information processing letters*, *85*(6), pp.317-325.

[14] Chapelle, O., Scholkopf, B. and Zien, A., 2009. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, *20*(3), pp.542-542.