- Did you receive any help whatsoever from anyone in solving this assignment? **No.**
  If you answered `yes', give full details: _____
- Did you give any help whatsoever to anyone in solving this assignment? **No.**
  If you answered `yes', give full details: _____

Report: Tuning Parameters on 12 benchmarks for SMO, DMNBtext, LogitBoost

1. Tuning the SMO classifier learner

1.1. Background of SMO

The SMO (weka class name) classifier implements John Platt's Sequential Minimal Optimization algorithm (literature name) for training a support vector classifier. It's mainly described in 3 papers (Hastie and Tibshirani 1998, Platt 1999, Keerthi, Shevade et al. 2001).

1.2. Parameters varied

In the experiment, most of the time we only vary the value of one parameter while keep others the same (unless otherwise stated).

-C: The complexity parameter (default = 1). It controls how soft the class margins are. We varied the parameter in the range [1 30] with a step size of 1, and found SMO generally has better robust performance with large C value, with min avg and max error reaches at C = 17 and 19, respectively. We decided to fix the value of C = 17, which seemed to strike a good balance between compute time and robustness.

-L: The tolerance parameter (default = 1.0e-3). A smaller tolerance parameter make the learning slower. SMO seems to be quite sensitive to the tolerance parameter, with the most robust performance occurring at relative large values (compared to the default L=1e-3). We first tested it within the range [1e-1, 1e-10], and find avg error rate reaches its minimum at L=1e-1, whereas max error rate keeps the same. We then refined the range to [1e-2, 2e-1], and still found L=1e-1 get the best average error, so we fixed L = 0.1.

-V: The number of folds for the internal cross-validation (default -1, use training data). SMO seems to be rather insensitive to the value of V. We varied the folds from 2 to 10, but no difference was observed compared to the default value. So we decided to keep using the default.

–N: Whether to 0=normalize/1=standardize/2=neither (default 0=normalize). We switched among the 3 modes, and found N = 1 generates more robust results than the default value, whereas N=2 takes too long time (almost forever, force stopped halfway) to compute. So we decided to fix N = 1.

-K: The Kernel to use. We switched among four kernels, PolyKernel (default), NormalizedPolyKernel, RBFKernel and Puk, and found the default to have the best robust performance. So we'll keep using it.

1.3. Final settings

The final SMO classifier is constructed with the parameter set to be "-C 17 –L 0.1 –N 1", other parameters use their default value. With this final setting, the classifier beats the one with

- Did you receive any help whatsoever from anyone in solving this assignment? **No.**
  If you answered `yes', give full details: _____
- Did you give any help whatsoever to anyone in solving this assignment? **No.**
  If you answered `yes', give full details: _____

default parameter with 0.700 v.s. 0.823 on avg $error_C/erro_{NB}$, and 1.087 v.s. 1.260 on max $error_C/error_{NB}$. Elapsed time to train the 12 models: 5193 ms.

2. Tuning the DMNBtext classifier learner

   2.1. Background of DMNBtext

   DMNBtext (weka class name) builds and uses a Discriminative Multinomial Naive Bayes classifier, which is also called Discriminative Parameter Learning for Bayesian Networks in literature. It's mainly described by (Su, Zhang et al. 2008).

   2.2. Parameters varied

   Since DMNBtext cannot handle the dataset directly, we wrap it under a FilteredClassifier with NominalToBinary and ReplaceMissingValues filter, which is itself wrapped under a MultiClassClassifier. We only tune the parameter of the very base classifier, i.e. the DMNBtext classifier.

   -I: The number of iterations that the classifier will scan the training data (default = 1)

   -M (on/off): Use the frequency information in data or not.

   We switched between using/not using frequency information, and varied the iteration number from 1 to 60. We found that generally a large iteration number leads to better robust performance, especially when it's combined with using the frequency information. We found the lowest avg error at I =57 with M on, and lowest max error at I=33 with M on. To compensate for computing time, we decided to fix the value at I = 33, and M on.

   2.3. Final settings

   The final DMNBtext classifier is constructed with the parameter set to be "-I 33 -M", and wrapped under some meta-classifier as described above. With this final setting, the classifier beats the one with default parameter with 0.927 v.s. 1.351 on avg $error_C/erro_{NB}$, and 1.306 v.s. 2.530 on max $error_C/error_{NB}$. Elapsed time to train the 12 models: 1682 ms.

3. Tuning the LogitBoost classifier learner

   3.1. Background of LogitBoost

   The LogitBoost (weka class name) performs classification using a regression scheme as the base learner. It's also called additive logistic regression in literature, and mainly described in (Friedman, Hastie et al. 2000).

   3.2. Parameters varied

   -P: Percentage of weight mass to base training on (default P = 100). Reduce the value can speed up learning process. We varied the parameter from 80 to 100, with a step size of 2, and found the default value has the best robust performance. So we will keep the default value.

- Did you receive any help whatsoever from anyone in solving this assignment? **No.**
  If you answered `yes', give full details: _____
- Did you give any help whatsoever to anyone in solving this assignment? **No.**
  If you answered `yes', give full details: _____

-L: Threshold on the improvement of the likelihood (default L = –Double.MAX_VALUE). We varied the parameter from –Double.MAX_VALUE to –Double.MAX_VALUE/1E10 with a step size of 10-fold decrease in absolute value each step, and found no difference. So we decided to use the default value.

-H: Shrinkage parameter, using a small value can reduce overfitting (default H =1). We varied the parameter from 0.1 to 2 with a step size of 0.1, and found H =1 (default) has the best result. Then we refined the range from 0.9 to 1.1 with a step size of 0.01, and found H =1.02 has the best avg error rate, and H = 0.96 has best max error rate. Overall, H=1.01 is the best result that beats the control (default) on both avg and max error rate. So we decided to fix H = 1.01.

-F: Number of folds for internal cross-validation.

-R: Number of runs for internal cross-validation.

Since these two variables may couple together, we varied F from 2 to 10 with a step size of 1 and N from 1 to 10 with a step size of 1 at the same time, and found no improvement of these parameters on avg and max error rate. So we decided to use the default value (F = 0, R = 1, i.e. no cross-validation).

-I: The number of iterations to be performed (default I = 10). We varied the parameter from 1 to 25 with a step size of 1, and found I = 19 has the best avg error rate, however, the control has the best maximum value. So we decided to use the default value.

-Q (On/off): Use resampling instead of reweighting for boosting. We switched between resampling and reweighting method, and found the default reweighting generally doing better, so we keep the default.

-W: The base classifier to be used. We switched among the default DecisionStump tree classifier, the M5P and the REPTree classifier (we didn't try other base classifiers which need a filter to handle the dataset), and found that DecisionStump has the lowest max error rate, whereas M5P has the lowest average error rate. However, the learning time using M5P is much longer, so we decided to keep using the default DecisionStump as the base classifier.

3.3. Final settings

The final LogitBoost classifier is constructed with the parameter set to be "-H 1.01", with other parameters kept to be the default. With this final setting, the classifier beats the one with default parameter with 0.728 v.s. 0.732 on avg $error_C/erro_{NB}$, and 1.032 v.s. 1.043 on max $error_C/error_{NB}$. Generally, the improvement is slight, which might indicate LogitBoost is not very sensitive to parameter tuning. Elapsed time to train the 12 models: 314 ms.

4. Summary

- Did you receive any help whatsoever from anyone in solving this assignment? **No.**
  If you answered `yes', give full details: _____
- Did you give any help whatsoever to anyone in solving this assignment? **No.**
  If you answered `yes', give full details: _____

As summarized in the table below, among the classifiers we used, LogitBoost has the best robust performance without parameter tuning. However, its robustness only improved a little after tuning, whereas SMO and DMNBtext have improved quite a lot with parameter tuning. After tuning, LogitBoost still has best performance in terms of max error rate, while SMO stands out in terms of the avg error rate. It's a hard decision again to weight between the avg error rate and max error rate. And finally we choose SMO to represent our best tuned classifier to generate 12 models to be submitted onto autolab.

Generally, the robust performance of each individual classifier in terms of avg and max error rate is improved by parameter tuning. However, none of them can exceed the result obtained from Milestone 2, which is not the performance of a single classifier, but the performance of 3 classifier together. This is an interesting result, and may give some hints for the ensemble method.

Table 1 Classifier Robust Performace

	Milestone 3: All the 12 benchmarks use only of the classifier in each column.						**Milestone2: Each benchmark uses the best of the 3 default classifier, and then calculate the avg and max $error_C/error_{NB}$**
	SMO		DMNBtext		LogitBoost		
	default	tuned	default	tuned	default	tuned	
Avg $error_C/error_{NB}$	0.823	0.700	1.351	0.927	0.732	0.728	0.663
Max $error_C/error_{NB}$	1.260	1.087	2.530	1.306	1.043	1.032	0.984

5. References

Friedman, J., et al. (2000). "Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors)." The Annals of Statistics **28**(2): 337-407.

Hastie, T. and R. Tibshirani (1998). "Classification by pairwise coupling." The Annals of Statistics **26**(2): 451-471.

Keerthi, S. S., et al. (2001). "Improvements to Platt's SMO algorithm for SVM classifier design." Neural Computation **13**(3): 637-649.

Platt, J. C. (1999). Fast Training of Support Vector Machines using Sequential Minimal Optimization.

Su, J., et al. (2008). Discriminative parameter learning for Bayesian networks. Proceedings of the 25th international conference on Machine learning, ACM.