# Empirical Comparison of Accuracy and Performance for the MIPSVM classifier with Existing Classifiers

Amund Tveit

Division of Intelligent Systems
Department of Computer and Information Science,
Norwegian University of Science and Technology,
N-7491 Trondheim, Norway
amundt@idi.ntnu.no

**Abstract.** This paper presents an empirical comparison of the Multicategory Incremental Proximal Support Vector Machine Classifier (MIPSVM) against the C4.5, Naive Bayes, Voted Perceptron, SMO, SVM and Logistic Regression classifiers on several datasets. The datasets are from the UCI Machine Learning repository, a Web Usage Log, and game usage logs from the Zereal Massively Multiplayer Online Game Simulator.

MIPSVM is found to be close to 1 order of magnitude (or more) faster than the other classifiers in all experiments. Based on pairwise T-test comparisons of accuracy between MIPSVM and the other classifiers, MIPSVM was found to be significantly more accurate than the Naive Bayes classifier, and not having significantly different accuracy than the other classifiers on the tested data sets.

## 1 Introduction

The objective of this paper is to test both computational performance and classification accuracy of Multicategory Incremental Proximal Support Vector Machine classifier (MIPSVM) [8] against other classifiers on several datasets.

All experiments are performed using the average accuracy from 10-fold cross-validation method, i.e. training on 9/10s of the data and test on the remaining 1/10 for all 1/10s in the dataset (i.e. 10 tests per dataset).For classifation accuracy comparison, graphs of average accuracy percentage from 10-fold cross validation are used, similar for time comparison

Computational performance is measured in average cross-validation wallclock time shown on a logarithmic scale relative to MIPSVM. MIPSVM has a value of 1 (hence not visible!), so if another classifier algorithm has a value of 10 it means it is one order of magnitude (ten times) slower than MIPSVM.

Finally we analyze the results using pairwise T-tests to get a general indication of MIPSVMs accuracy relative to the other classifiers compared with.

## 2 Classification Datasets Overview

Classification Experiments have been performed on 3 types of data:

1. Three datasets from the UCI machine learning repository [1], experiments on these datasets were performed by former MSc student Håvard Engum (co-supervised by the author) [4].
2. Zereal game simulation output has been used as datasets for the section on "Game Player Classification".
3. Web usage logs from www.jfipa.org has been used as datasets for the section on "Web Intelligence".

### 2.1 Classification Tools Overview

The classifiers that is compared to MIPSVM implementation in IncRidge are classifiers in the Weka toolkit [10] and the C-SVM algorithm in the LIBSVM toolkit. These toolkits are described briefly below. (Other classifier tools were considered, but not selected since they didn't support 10-fold cross-validation testing).

**WEKA** WEKA is an akrynom for "Waikato Environment for Knowledge Analysis" and is a software tool that consists of a set of machine earning algorithms including classifiers. Weka is implemented in Java and has been succesfully run on all major computer platforms citewitten:mining. The Weka classifiers used are Naive Bayes, C4.5, Logistic Regression, Voted Perceptron and SMO. Naive Bayes is usually the *default* classifier in many domains, it is simple and gives in general good results. Logistic Regression is considered to be the standard classification method in the domain of medical research [7]. C4.5 has been shown to perform well compared to other classifiers, in fact outperforming Linear Discriminant Analysis and Logistical Analysis for the classification of high performance mutual funds [5]. Sequential Minimal Optimization (SMO) is an approximate and fast method to train Support Vector Machine Classifiers [6].

**LIBSVM** LIBSVM is an integrated software for support vector classification, regression and distribution estimation. It support multicategory classification and different SVM formulations [2].

In order to get optimal results with LIBSVM, there are a few steps that can be done in advance to enhance LIBSVM's performance both in accuracy and computational efficiency [3].

The first step is to scale the features to the range $[-1, 1]$ or $[0, 1]$. This is because you do not want attributes in greater numeric ranges dominate those in smaller numeric ranges. The other advantage of scaling is that you avoid numerical difficulties of large attribute values when calculating the values of kernel functions. LIBSVM features a tool, $svm - scale$m that scales the data.

The second step is to find optimal values for $C$ and $\gamma$, where $C$ is the penalty parameter from the original SVM-formulation and $\gamma$ is the constant in the Radial Basis Function Kernel. LIBSVM features a tool programmed in Python, easy.py, that finds good values for $C$ and $\gamma$ These two steps described above is time-consuming. Especially step two takes from five to ten minutes on a somewhat fast computer (AMD Athlon 1.66GHz), depending on the data set. In the experiments, LIBSVM was tested both with the two preparing steps above and without any preparing steps.

### 2.2 Experiments on UCI datasets

This section presents results from classification experiments performed on three different datasets from the UCI Machine Learning Repository:

1. **Waveform Generator database** - Generated by a C-program. Each class in the dataset is generated from a combination of " of 2 "base" waves with added noise. The 5000 examples have 40 numeric attributes and 3 classes.
2. **Image Segmentation Database** - Data drawn randomly from a database of seven outdoor images. The images were hand segmented to create a classification for every pixel. Each instance has 3x3 region and 19 attributes. There are 7 classes in the dataset named brickface, sky, foliage, cement, window, path and grass. Number of examples are 2310.
3. **Letter Recognition Database** - Data with 20,000 examples of black-and-white rectangular pixels displays of the capital letters in the English alphabet, hence the number of classes are 26, one for each letter. Each example has 16 numerical features.

It was also planned to use the Forest Covertype data in the experiment, but it was too memory intensive for Weka to handle.

For tables with exact figures of classification accuracy and computational performence on UCI datasets, see table **??** and **??**, respectively.

**Classification Accuracy - UCI Datasets** As we can see from the results in figure 1, MIPSVM performs comparably well when it comes to classification accuracy for the Waveform and Image Segment datasets. For the Letter Recognition dataset it performs considerably worse than the other classifiers. This is likely to be caused by that MIPSVM doesn't have any balancing mechanisms one-against-the-rest classifiers may gain from having when there are many classes. Another reason could be that the letter dataset might be not be linearly separable.

**Computational Performance - UCI Datasets** The computational efficiency is measured in seconds of running time. The experiment was run on a AMD Dual Athlon MP 2100+ (1.66GHz) with 2GB ram. Each configuration is run 10 times, and the average running time is used as a result. The figure shows the *runtime* relative to MIPSVM.

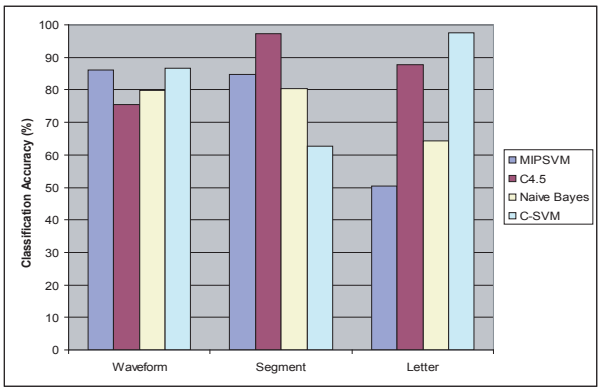


**Fig. 1.** Accuracy - UCI Datasets

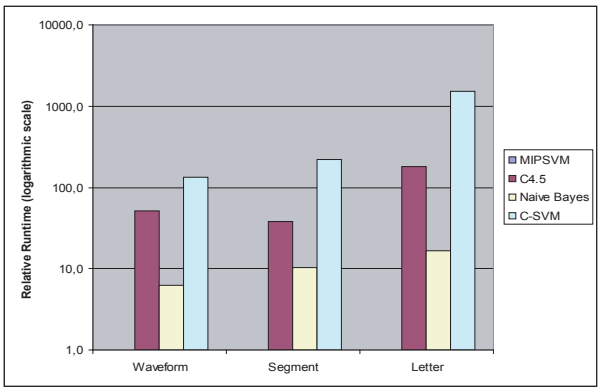


**Fig. 2.** Performance - UCI Datasets

As shown in figure 2 MIPSVM (IncRidge) *outperforms* the other classifiers with respect to computational efficiency.

### 2.3 Experiments on Zereal Datasets

This section presents results from classification experiments performed on three different datasets output from the Zereal Massively Multiplayer Online Game Simulator:

1. **zerClassRnd** - classify agents into the 4 classes PlanAgent, MarkovKiller, Killer or the non-personal character category Monster. It has 160 examples.
2. **PlayerMonster** - classify agents into the 2 classes Player or Monster. It has 160 examples
3. **zerealPlayers** - classify agents into the 3 classes PlanAgent, MarkovKiller or Killer. It has 120 examples.

All datasets have 7 numeric features representing frequencies of occurences per agent (i.e. player or monster):

**pickupfood** - food item frequency (for increased health)
**pickupkey** - key frequency (for unlocking doors)
**pickuppotion** - potion frequency (for increased health)
**pickupsword** - sword frequency (for more powerful combat)
**attack** - attack frequency (aggressitivity)
**leaveworld** - frequency for number of times left a subworld
**walk** - walk frequency (movability)

For exact figures of classification accuracy and computational performence on Zereal datasets, see table **??** and **??**, respectively.

**Classification Accuracy - Zereal Datasets** MIPSVM is the best classifier for the *Player Types* dataset, shared 2nd with C4.5 for the *Player vs NPCs* dataset, and 4th for the *Players and NPCs* dataset (figure 3)

**Computational Performance - Zereal Datasets** Comparison of performance on Zereal datasets were only performed on the *Players and NPCs* dataset since the other datasets were too small to take measurable time with the MIPSVM classifier.

As shown in figure 4 MIPSVM is the fastest classifier, being approximately 2.5 orders of magnitude faster than the SMO classifier and approximately 1.5 orders of magnitude faster than the rest of the classifiers.

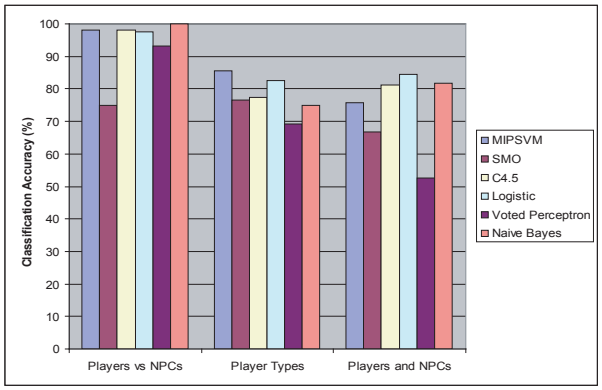


**Fig. 3.** Accuracy - Player and NPC Types

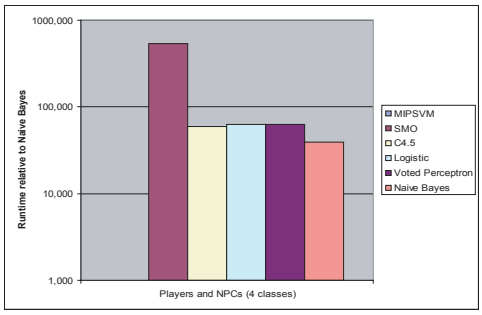


**Fig. 4.** Performance - Player and NPC Types

### 2.4 Experiments on Web Usage Logs Datasets

This section presents results from classification experiments performed on four different datasets based on a web usage log from extracted from www.jfipa.org. This web usage log is then preprocessed in order to create 4 types of data for various classification problems:

1. **Intrapage clickstream prediction** - The 8 datasets - P0 to P7 - are for prediction of the next selected page based on clickstreams. There is one classifier per current page, i.e. the previous pages in a user session and the current page is used to predict the next page. Each dataset has 4 numeric attributes representing the pages in the clickstream, with the class being the 5th and last page going to be predicted.
   **P0** - 1728 examples and 26 classes
   **P1** - 184 examples and 10 classes
   **P2** - 166 examples and 3 classes
   **P3** - 147 examples and 24 classes
   **P4** - 179 examples and 27 classes
   **P5** - 270 examples and 24 classes
   **P6** - 101 examples and 22 classes
   **P7** - 191 examples and 19 classes
2. **Intrasite clickstream prediction** - One dataset for prediction of the next selected page based on clickstreams. This dataset uses one classifier, i.e. does not use the graph structure of links and pages as the *Intrapage clickstream prediction* datasets, but have the same attribute structure. It has 4957 examples and 191 classes.
3. **Search Engine Result Clickstreams** - Two datasets used to answer the questions: 1) *Do people arriving from search engine results differ from those who don't?* and 2) *Do people arriving from results from the Google search engine differ from those from other search engines?* Both datasets have 4957 examples and 2 classes, but different classes for each dataset. These datasets also has the same attribute structure as the previous ones.
4. **Spider and Person Clickstreams** - One dataset used to answer the question: *Are web crawler (agent) clickstreams different from those of users?* This dataset has 9914 examples and 2 classes. This dataset has the same attribute structure as the previous ones.

**Classification Accuracy - Web Datasets** As observed from figure 5 MIPSVM is among the 2 best classifiers for 4 of the 8 datasets (P0,P1,P2 and P4) and only the worst for one dataset (P6) for the prediction of intrapage clickstreams.

Due to OutOfMemory exceptions Logistic Regression and Voted Perceptron failed to be applied to the Intrasite prediction problem. MIPSVM was a lot less accurate than C4.5, but more accurate than Naive Bayes (figure 6).

For the search engine result clickstream datasets all classifiers except Naive Bayes have very similar performance, but with Logistic Regression and MIPSVM

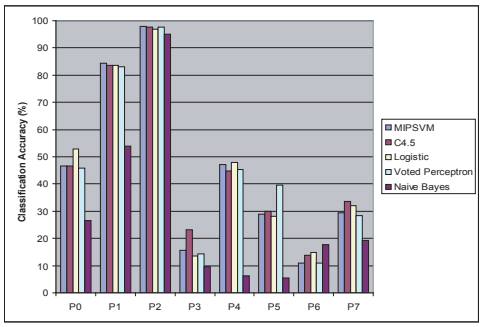


**Fig. 5.** Accuracy - intrapage clickstreams

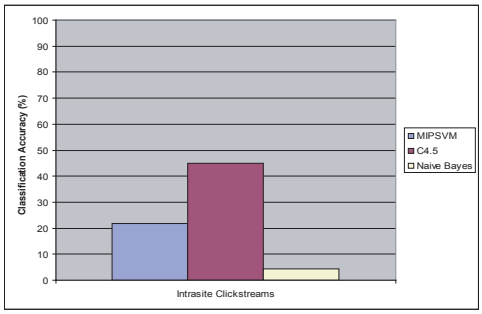


**Fig. 6.** Accuracy - intrasite clickstreams

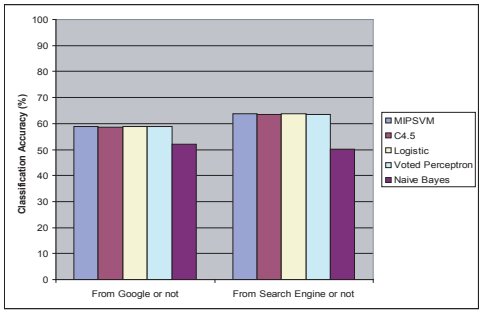


**Fig. 7.** Accuracy - referring URL type

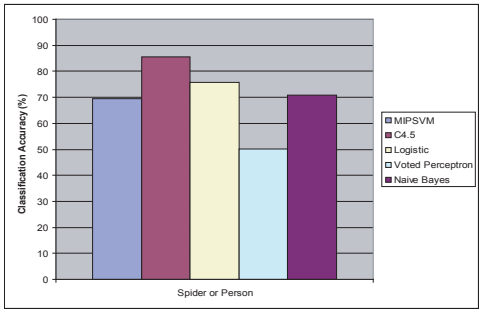


**Fig. 8.** Accuracy - person vs agent clickstreams

being slightly more accurate than the other classifiers on the *From Search Engine or Not* dataset (figure 7).

C4.5 is the most accurate classifier on the *Spider or Person* dataset with Naive Bayes being the third and slightly more accurate than MIPSVM (figure 8).

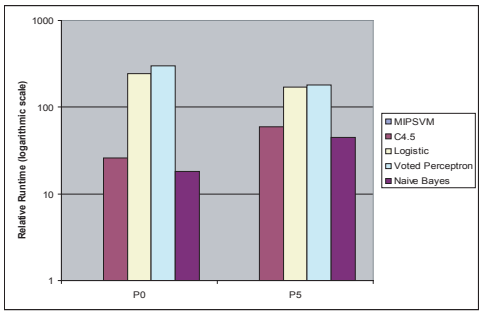


**Fig. 9.** Performance - intrapage clickstreams

**Computational Performance - Web Datasets** Comparison of performance for *Intrapage clickstream prediction* were only done on the P0 and P5 datasets since the other datasets were too small to take measurable time with the MIPSVM classifier. MIPSVM is between 1 and 2 orders of magnitude faster than C4.5 and Naive Bayes, and between 2 and 3 orders of magnitudes faster than Logistic Regression and the Voted Perceptron algorithms (figure 9).

MIPSVM is about 1 order of magnitude faster than Naive Bayes and 1.5 orders of magnitude faster than C4.5 on the *Intrasite clickstream prediction* dataset (figure 10)

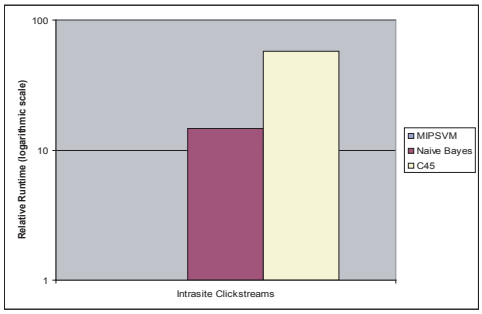


**Fig. 10.** Performance - intrasite clickstreams

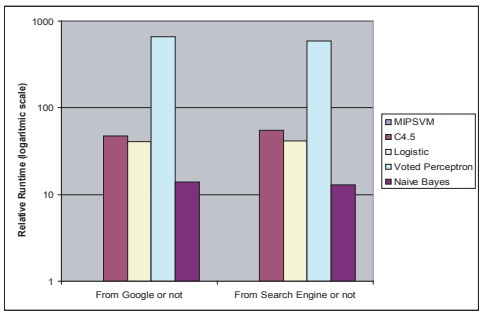


**Fig. 11.** Performance - referring URL type

On the *Search Engine Result Clickstreams* datasets MIPSVM is between 1 and 1.5 orders of magnitude faster than C4.5, Naive Bayes and Logistic Regression. Compared to Voted Perceptron it is about 2.5 orders of magnitude faster (figure 11).

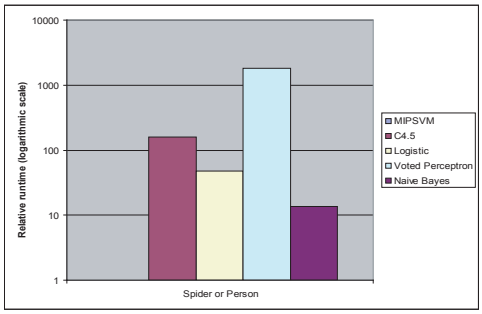


**Fig. 12.** Performance - person vs agent clickstreams

On the *Spider and Person Clickstreams* dataset MIPSVM is approximately 1 order of magnitude faster than Naive Bayes, 2 orders of magnitude faster than C4.5, 1.5 orders of magnitude faster than Logistic Regression and 3 orders of magnitudes faster than Voted Perceptron (figure 12.

## 3 Conclusion

Finally MIPSVM was been compared to other classifiers on three main types of classification data:

1. Classification datasets from the UCI Machine Learning Repository
2. Classicication datasets from the Zereal Massively Multiplayer Online Game Simulator
3. Classification datasets from a Web Usage Log extracted from www.jfipa.org

### 3.1 Computational Performance

MIPSVM is faster than the other classifiers in all experiments. This can be caused by several reasons: the algorithms, implementation overhead, programming language and optimizations in compiler and virtual machines, external libraries used (the ATLAS linear algebra library is used in the qimplementation of MIPSVM [9]).

### 3.2 Classification Accuracy

In order to make some more general conclusions on MIPSVM's accuracy compared to the other classifiers, pairwise T-tests have been used (see chapter **??** for a description of the method). These tests concluded that on the 18 datasets the default configurations of MIPSVM, Naive Bayes and C4.5 were used (appendix C). MIPSVM is *significantly more accurate* than Naive Bayes (5% confidence level, p-value = 0.013), and *not significantly different* from C4.5 (even though C4.5 is more accurate). On the 14 datasets the default configurations of MIPSVM, Logistic Regression and Voted Perceptron were used (appendix C) pairwise T-tests showed that MIPSVM and Logistic Regression have *not significantly different accuracy* (even though Logistic Regression is more accurate), and that MIPSVM and Voted Perceptron have *not significantly different accuracy* (even though MIPSVM is more accurate).

A modestly bold conclusion and recommendation is that MIPSVM is a suitable alternative to Naive Bayes as the *default classifier* when first attacking a classification problem.

### 3.3 Further Work

Opportunities for further work include:

p

- Add support for a variable number of features in examples
- Develop support for parallelized *decremental* PSVM
- Add kernel support
- Add incremental balancing mechanisms to improve accuracy for cases with many, potentially unbalanced classes.
- Investigate the performance effect of altering matrix multiplication algorithms
- Investigate whether the symmetric matrix system ($\mathcal{A}$) can be transformed into a Toeplitz or Hankel matrix system for more efficient computation

## References

1. C. L. Blake and C. J. Merz. UCI Repository of Machine Learning Databases. Online, `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 1998.
2. Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
3. Chih-Chung Chang Chih-Wei Hsu and Chih-Jen Lin. *A practical guide to SVM classification*. Department of Computer Science and Information Technology, National Taiwan University. Paper available at `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.
4. Håvard Engum. IncRidge: A Software Tool for Scalable Incremental and Decremental Classifier Algorithms based on Support Vector Approximations. Technical report, IDI, NTNU, July 2003.

5. Robert C. Norris. Classifying High Performance Mutual Funds: A Comparison C4.5, LDA and Logit. In *Proceedings of the INFORMS Conference on Information Systems and Technology, Session on AI in Business and Industry: Applications and Theory*, May 1996.
6. J. Platt. Sequential Minimal Optimization. Technical Report MSR-TR-98-14, Microsoft Research, 1998.
7. Sloan Rush. Logistic Regression: The Standard Method of Analysis in Medical Research. Technical Report Mathematics #S3, Trinity University, 2001.
8. Amund Tveit and Magnus Lie Hetland. Multicategory Incremental Proximal Support Vector Classifiers. In *Proceedings of the 7th International Conference on Knowledge-Based Information & Engineering Systems (KES'2003)*, number 2773 in Lecture Notes in Artificial Intelligence (LNAI), pages 386–392. Springer-Verlag, 2003.
9. Richard C. Whaley, Antoine Petitet, and Jack J. Dongarra. Automated Empirical Optimization of Software and the ATLAS Project". *Parallel Computing*, 27(1-2):3–25, 2001.
10. Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1st edition, October 1999.