Classification of poisonous mushrooms using High-Performance Support Vector Machines

Shigeo, Hina, Emilio

Summary

Shigeo Kitamura
In this project, we confirmed the usefulness and scalability of the High-Performance Support Vector Machine (HPSVM) algorithm proposed by He et al. by applying it to various datasets, including the mushroom dataset, and by comparing its accuracy with that of other algorithms.

Project Github : GitHub Repository

Motivation

We wanted to see the benefits of training in a distributed environment like Hadoop. We then discovered the paper about HPSVM. We chose the paper because tha dataset used is about mushrooms which are familiar.

Research paper details

He, T., Wang, T., Abbey, R., & Griffin, J. (2019). High-Performance Support Vector Machines and Its Applications. *ArXiv, abs/1905.00331*.

This paper introduces HPSVM, a distributed algorithm for Support Vector Machines (SVM) that is designed for large-scale data processing.

Key aspects of the paper:

1. **Algorithm Design**: The authors propose a distributed SVM algorithm that addresses two main challenges:
    a. Distributing computations to machines without shuffling data
    b. Minimizing inter-machine communications to maximize performance
2. **Technical Approach**:
    a. Uses an interior-point method with the Sherman-Morrison-Woodbury formula to transform large matrices into smaller, more manageable ones
    b. Reduces memory requirements from $O(n^2)$ to $O(n) + O(m^2)$ where n is the number of observations and m is the number of features
    c. Implements a distributed Newton method using Message Passing Interface (MPI)
3. **Implementation Details**:
    a. Works in both Symmetric Multiple Processing (SMP) mode and Massively Parallel Processing (MPP) mode
    b. Includes a Universal Data Feeder (UDF) that supports various platforms including Hadoop, Teradata, Greenplum, and Aster
    c. Data access methods minimize data movement between nodes
4. **Complexity Analysis**:
    a. Memory usage per worker node: $O(mn/p) + O(m^2)$ where p is the number of worker nodes
    b. CPU time per Newton iteration: $O(nm^2/p) + O(m^3)$
5. **Experimental Results**:
    a. Compared with LIBSVM in R on several datasets (Mushroom, Adult, Face)
    b. HPSVM achieved similar accuracy but ran much faster on large datasets
    c. Demonstrated good scalability when increasing the number of computing nodes (20 to 100 nodes)
    d. Outperformed Apache Spark's SVM implementation in both speed and accuracy

The authors demonstrate that HPSVM is particularly effective for large datasets where traditional SVM implementations struggle. The algorithm's design allows it to scale well in distributed environments while maintaining classification accuracy.

Dataset details

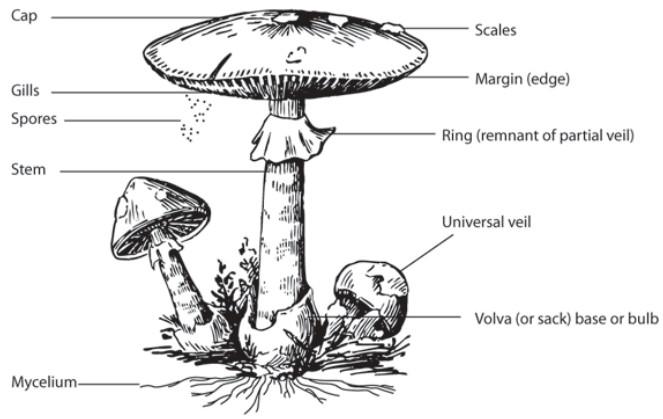
tonomura.hina@gmail.com

From Audubon Society Field Guide; mushrooms described in terms of physical characteristics; classification: poisonous or edible

The dataset shape is
” 8123 rows × 23 columns “ include all object values which describe each column’s labels. Mushrooms are described in terms of physical characteristics such as cap-shape, color, gill-size. It is classified as poisonous as a target feature like poisonous or edible.

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8124 entries, 0 to 8123
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   cap-shape                 8124 non-null   object
 1   cap-surface               8124 non-null   object
 2   cap-color                 8124 non-null   object
 3   bruises                   8124 non-null   object
 4   odor                      8124 non-null   object
 5   gill-attachment           8124 non-null   object
 6   gill-spacing              8124 non-null   object
 7   gill-size                 8124 non-null   object
 8   gill-color                8124 non-null   object
 9   stalk-shape               8124 non-null   object
 10  stalk-root                5644 non-null   object
 11  stalk-surface-above-ring  8124 non-null   object
 12  stalk-surface-below-ring  8124 non-null   object
 13  stalk-color-above-ring    8124 non-null   object
 14  stalk-color-below-ring    8124 non-null   object
 15  veil-type                 8124 non-null   object
 16  veil-color                8124 non-null   object
 17  ring-number               8124 non-null   object
 18  ring-type                 8124 non-null   object
 19  spore-print-color         8124 non-null   object
 20  population                8124 non-null   object
 21  habitat                   8124 non-null   object
 22  poisonous                 8124 non-null   object
dtypes: object(23)
memory usage: 1.4+ MB
```

	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	...
0	5	2	4	1	6	1	0	1	4	0	...
1	5	2	9	1	0	1	0	0	4	0	...
2	0	2	8	1	3	1	0	0	5	0	...
3	5	3	8	1	6	1	0	1	5	0	...
4	5	2	3	0	5	1	1	0	4	1	...



dataset : https://archive.ics.uci.edu/dataset/73/mushroom

Data preprocessing and feature engineering

tonomura.hina@gmail.com

・Steps for Data preprocessing

1. **Read Dataset** : Downloaded mushroom dataset from uci repo
    a. Combined to the one mushroom_dataset.csv since in the original dataset zip, data was divided by features and targets.

2. **Cleaning data**:

    a. Finding the missing values : only the column[‘stalk-root’] has 2480 missing values in 8124 rows × 23 columns.
    b. Handling the missing values :Convert the missing value to ‘m’ which is a new letter because of too much for dropping it.
    c. Finding duplicate : There is no duplicated error

・Feature Engineering

1. **Encoding**
    a. Using Label Encoding to all columns since it includes categorical data and labeled by letters which do not mean the order is irrelevant.

2. **Feature Scaling : Implemented each ML improvement model**
    a. Using Standard Scaler: To use distance-based algorithms such as SVM and PCA, convert to mean 0 and standard deviation 1.

Steps reproduced from the paper

Shigeo Kitamura

We implemented a parallel implementation of Support Vector Machines designed for high-performance computing environments using MPI (Message Passing Interface) for distributed computation. It's specifically built to handle large-scale machine learning tasks by distributing the computational workload across multiple nodes.

hpsvm.py

```
❯ mpirun -n 8 python hpsvm.py
Generated dataset with 8000 training samples and 2000 test samples
2025-03-13 15:27:16,075 - HPSVM - INFO - Initialized HPSVM with 8 nodes
2025-03-13 15:27:16,075 - HPSVM - INFO - Parameters: tau=1.0, tol=0.0001, max_iter=50, kernel=linear
2025-03-13 15:27:16,075 - HPSVM - INFO - Starting HPSVM training with 8000 samples and 20 features
2025-03-13 15:27:16,075 - HPSVM - INFO - Data distributed among 8 nodes
2025-03-13 15:27:16,080 - HPSVM - INFO - Iteration 0: duality gap = 0.599506, step size = 0.126170
2025-03-13 15:27:16,088 - HPSVM - INFO - Iteration 5: duality gap = 0.496357, step size = 0.000185
2025-03-13 15:27:16,096 - HPSVM - INFO - Iteration 10: duality gap = 0.498525, step size = 0.000000
2025-03-13 15:27:16,105 - HPSVM - INFO - Iteration 15: duality gap = 0.498462, step size = 0.000046
2025-03-13 15:27:16,112 - HPSVM - INFO - Iteration 20: duality gap = 0.498854, step size = 0.000013
2025-03-13 15:27:16,122 - HPSVM - INFO - Iteration 25: duality gap = 0.500336, step size = 0.005115
2025-03-13 15:27:16,131 - HPSVM - INFO - Iteration 30: duality gap = 0.500252, step size = 0.000150
2025-03-13 15:27:16,140 - HPSVM - INFO - Iteration 35: duality gap = 0.500968, step size = 0.000071
2025-03-13 15:27:16,149 - HPSVM - INFO - Iteration 40: duality gap = 0.501637, step size = 0.000038
2025-03-13 15:27:16,157 - HPSVM - INFO - Iteration 45: duality gap = 0.502626, step size = 0.000020
2025-03-13 15:27:16,162 - HPSVM - INFO - Iteration 49: duality gap = 0.505756, step size = 0.013602
2025-03-13 15:27:16,164 - HPSVM - INFO - Number of support vectors: 8000 out of 8000 samples
Training completed in 0.09 seconds
Test accuracy: 0.8910
```

Although we were not able to prepare the same environment as in the paper, we did confirm an 89.10% accuracy rate for distributed training on 8 nodes using the sklearn sample dataset.

Contributions (what we did)

Apply HPSVM to Other Datasets

Shigeo Kitamura

We applied HPSVM to create classification models to the following datasets:

- Mushroom
- Wholesale customers
- Hate Speech and Offensive Language Dataset

Comparison of accuracy confirmed that the results are similar for similar data sets.

Dataset	**Accuracy**
Mushroom	0.9532
Wholesale customers	0.9773
Hate Speech and Offensive Language Dataset	0.9443

hpsvm_application.ipynb

evadilloriesco@gmail.com

Our team successfully implemented the HPSVM methodology described in the research paper. To thoroughly evaluate the model's effectiveness, we applied it to multiple datasets, including the original dataset used in the paper and two additional datasets with varying parameters. Furthermore, we conducted a comparative analysis between HPSVM and several other machine learning models using the paper's original dataset. This comprehensive approach allowed us to assess the model's robustness, generalizability, and relative performance across different data contexts and against established machine learning algorithms.

・Trying Different classification model

tonomura.hina@gmail.com

Since this paper uses HPSVM, the following alternative classification model was used and evaluated in this part.(※ Github)

Models :
・Logistic Regression
・Random Forest (＋hard restriction version)

・Decision Tree
・XGboost

	Training Accuracy	Test Accuracy	Recall
Logistic Regression	96.5%	96.0%	95%
Decision Tree	98.0%	97.6%	98%
Random Forest	98.9%	99.1%	100%
XG boost	100%	100%	100%

Other metrics :
・PCA
- Using for dimensional reduction. 10 principal components out of 23 columns explain about 83.8% of the variance.

・Feature Importance
- gill-coor is high score of importances for recognizing poisonous

・Grid Search
- 99.6% accuracy when set up the following parameter

```
{'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 100, 'subsample': 0.7}
```

In this case, the best model was considered to be XGboost and Random Forest, which yielded almost 100% accuracy. The accuracy of HPSVM in the original paper was about 95%, so the accuracy was improved. Although the purpose of the paper was to improve the performance of the original SVM by HPSVM, the application of this model, which is different from SVM, is considered to have provided a higher accuracy than that.

Significant improvements

evadilloriesco@gmail.com

- More rigorous model comparison: Tried additional SVM variations with kernel tuning and experimented with different datasets.
- Optimizing code efficiency and time: Since we are able to work separately multiple contributions can be completed at the same time.
- Comprehensive performance metrics: Beyond accuracy, we're evaluating models using precision, recall and F1-score to provide a more nuanced understanding of performance trade-offs.

Challenges and Learning

- **Paper :**
  Since the paper did not include specific source code, we needed to implement HPSVM based on the formula.

- **Preprocessing :**
  Because the data set contained categorical data all condensed to a single word, it was difficult to convert the data into actual information when creating the histograms.

- **The balance of Data accuracy and quality:**
  It was difficult to strike a balance between creating a highly accurate model and maintaining data quality. The more one tries to interpret the data more deeply and accurately (encoding and selection in PCA), the more likely it is to fall prey to the curse of dimensionality and overlearning. Therefore, we were challenged to determine the balance of what level of quality was acceptable.

- **High accuracy (almost 100%):**
  It is easy for many machine learning algorithms to learn these boundaries.
  Perhaps this data set has the following characteristics

    - Clear feature-category relationship: specific feature patterns clearly indicate classification categories.
    - Discriminating power of features: The features used (e.g., gill color) are considered highly discriminating for the classification task.
    - Data quality: the dataset has been carefully collected and classified by experts and is likely to be low noise and accurate in recording features.

Conclusion and Future Scope

evadilloriesco@gmail.com

- When performing classification on huge datasets, we would like to verify the difference in accuracy and training time between using HPSVM on multiple nodes and using existing algorithms on a single node.

- **Future Scope : Creating interaction features**
  When there are many categorical variables, we want to provide new perspectives to the model by multiplying or combining the features, since the combination of variables may affect the target variable several times.

References

https://archive.ics.uci.edu/dataset/73/mushroom

https://arxiv.org/pdf/1905.00331