**Application Project Final Report**

**1. Motivation & Dataset**

The machine learning algorithms we learned in classes is purely theoretical and the underlying mathematical deductions are rather abstract in terms of minimizing the training loss. In the real-world application of machine learning, the performance of different techniques varies and the variation is specific. Therefore, to study and compare the techniques, we applied different learning algorithms to assess their strengths and weaknesses with respect to a particular dataset.

**1.1 Dataset Source Information**

The dataset for the application project is Parkinsons Telemonitoring Data Set. The dataset was created by Athanasios Tsanas ([tsanasthanasis@gmail.com](mailto:tsanasthanasis@gmail.com)) and Max Little ([littlem@physics.ox.ac.uk](mailto:littlem@physics.ox.ac.uk)) of the University of Oxford, in collaboration with 10 medical centers in the US and Intel Corporation who developed the telemonitoring device to record the speech signals1.

There are 5875 instances in the dataset without any missing data. The actual number of features is 20 and there are two different labesl associated with each instance. The main aim of the data is to predict the motor and total UPDRS scores (Unified Parkinson Disease Rating Scale) from the 16 voice measures (the other four additional features – subject ID, age, sex and testing time – are rather irrelevant).

**1.2 Dataset Application**

This dataset measures the extent to which Parkinson affects the voice in various amplitude. One of the major applications is in the Parkinson diagnosis and assessment field as the voice measurements can be used to predict the clinician's Parkinson's disease symptom score on the UPDRS scale.

**2. Milestones and Results**

**2.1 Milestones Overview and Model Implementation**

In the application project, there are four milestones. The first milestone studies the linear regression and the second milestone studies the Gaussian processes. The third milestone – dimensionality reduction – is categorized as data preprocessing instead of a model training. At the fourth milestone, two optional tasks – support vector machine and decision tree are studied, as well as the performance of each machine learning model.

In linear regression, the loss function is the square error (MSE) and the regularization is L2 (Ridge Regression). To minimize the loss function, the algorithm used is stochastic gradient descent for efficiency.

In Gaussian processes, three models are built using squared exponential kernel, exponential kernel and the Matern 3/2 kernel respectively. The kernel parameters are all set to one to eliminate the effect of hyper-parameters. The selection of the kernel function is rather intuitive – to compare two kernel functions covered in class and another commonly used kernel function with each other. The model is measured by negative log-likelihood and the MSE is also recorded.

The support vector machine model is trained with Gaussian kernel, and the decision tree is trained with CART algorithm without pruning. The model performance is also measure by out of fold MSE.

All models are trained in MATLAB using built-in Machine Learning Toolbox with 10-fold cross validation to decrease overfitting. In terms of the implementation complexity, Kernel SVM is more complicated than linear regression, as demonstrated in project 3 and project 2. The covariance matrix computation and quadratic programming is more time-consuming than gradient descent. Likewise, decision tree is an iterative algorithm and has similar implementation efforts to the linear gradient, while Gaussian processes are more complex. In conclusion, decision tree and linear regression are the easiest to implement, while SVM and Gaussian processes are more difficult.

**2.2 Model Accuracy Result**

The tables below illustrated the 10-rerun out-of-fold loss results of each model trained on full dataset. The average MSE performance is used as the accuracy measurement.

It is obvious that Gaussian processes have the best training accuracy (41.86 for Squared Exponential Kernel and 40.87 for Matern 3/2 Kernel), followed by SVM (53.40), linear regression (61.15) and lastly decision tree (75.92).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Loss with M label** | **GP with Squared Exponential Kernel** | **GP with Matern 3/2 Kernel** | **Linear Regression** | **SVM** | **Decision Tree** |
| Trial 1 | 47.70 | 37.94 | 57.15 | 54.12 | 79.36 |
| Trial 2 | 39.32 | 40.47 | 58.14 | 55.22 | 76.81 |
| Trial 3 | 44.57 | 43.32 | 65.08 | 51.60 | 74.47 |
| Trial 4 | 42.87 | 41.96 | 60.60 | 52.54 | 75.35 |
| Trial 5 | 38.73 | 37.98 | 58.55 | 54.13 | 67.51 |
| Trial 6 | 41.68 | 39.62 | 62.94 | 54.53 | 76.54 |
| Trial 7 | 39.53 | 41.24 | 59.21 | 54.43 | 73.52 |
| Trial 8 | 42.67 | 44.07 | 62.44 | 50.10 | 79.60 |
| Trial 9 | 42.51 | 40.64 | 60.85 | 54.36 | 71.24 |
| Trial 10 | 39.04 | 41.52 | 66.55 | 52.94 | 84.77 |
| **Average** | **41.86** | **40.87** | **61.15** | **53.40** | **75.92** |

Table 1 (above): training losses for M label by different models

Table 2 (below): training losses for T label by different models

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Loss with T label** | **GP with Squared Exponential Kernel** | **GP with Matern 3/2 Kernel** | **Linear Regression** | **SVM** | **Decision Tree** |
| Trial 1 | 64.31 | 70.70 | 96.33 | 98.82 | 113.35 |
| Trial 2 | 73.60 | 63.55 | 92.72 | 96.19 | 126.28 |
| Trial 3 | 64.42 | 71.91 | 107.87 | 88.90 | 140.62 |
| Trial 4 | 62.11 | 64.27 | 99.36 | 104.72 | 133.44 |
| Trial 5 | 72.87 | 79.97 | 102.02 | 95.40 | 150.44 |
| Trial 6 | 76.67 | 71.62 | 112.78 | 102.54 | 116.58 |
| Trial 7 | 71.59 | 72.89 | 102.65 | 101.71 | 119.26 |
| Trial 8 | 65.62 | 78.75 | 97.86 | 93.28 | 132.15 |
| Trial 9 | 71.00 | 69.21 | 98.17 | 97.34 | 116.47 |
| Trial 10 | 68.08 | 64.68 | 109.66 | 98.58 | 147.63 |
| **Average** | **69.03** | **70.75** | **101.94** | **97.75** | **129.62** |

**2.3 Time Efficiency Result**

The tables below illustrated the 10-rerun training time results of each model trained on full dataset. It is obvious that linear regression have the shortest training time and hence best time complexity, followed by decision tree, SVM and lastly Gaussian Processes.

For this particular dataset, it is okay that Gaussian processes have significantly longer training time because overall ~30s are not too long and the model accuracy is much more important than the training time. In any cases where the training data is significantly larger and the training time is at least as important as the model accuracy, Gaussian processes might become a really bad option as the training time grows cubically.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time with M Label** | **GP with Squared Exponential Kernel** | **GP with Matern 3/2 Kernel** | **Linear Regression** | **SVM** | **Decision Tree** |
| Trial 1 | 28.91 | 31.70 | 0.36 | 1.09 | 0.60 |
| Trial 2 | 23.32 | 31.27 | 0.05 | 1.01 | 0.45 |
| Trial 3 | 28.86 | 27.29 | 0.03 | 0.96 | 0.47 |
| Trial 4 | 23.77 | 30.08 | 0.03 | 0.97 | 0.50 |
| Trial 5 | 25.72 | 33.40 | 0.03 | 0.98 | 0.50 |
| Trial 6 | 25.61 | 34.77 | 0.03 | 0.99 | 0.20 |
| Trial 7 | 28.36 | 31.50 | 0.03 | 0.96 | 0.17 |
| Trial 8 | 31.63 | 33.81 | 0.03 | 1.91 | 0.18 |
| Trial 9 | 31.79 | 42.22 | 0.03 | 1.96 | 0.18 |
| Trial 10 | 33.71 | 36.01 | 0.03 | 0.97 | 0.16 |
| **Average** | **28.17** | **33.20** | **0.06** | **1.18** | **0.34** |

Table 3 (above): training time for M label by different models

Table 4 (below): training time for T label by different models

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time with T Label** | **GP with Squared Exponential Kernel** | **GP with Matern 3/2 Kernel** | **Linear Regression** | **SVM** | **Decision Tree** |
| Trial 1 | 30.45 | 47.43 | 0.02 | 0.93 | 0.17 |
| Trial 2 | 28.70 | 38.08 | 0.03 | 0.98 | 0.18 |
| Trial 3 | 32.26 | 33.44 | 0.03 | 1.00 | 0.17 |
| Trial 4 | 30.06 | 33.73 | 0.03 | 0.97 | 0.16 |
| Trial 5 | 31.40 | 35.51 | 0.03 | 0.97 | 0.16 |
| Trial 6 | 26.50 | 35.54 | 0.03 | 1.26 | 0.16 |
| Trial 7 | 29.92 | 37.55 | 0.03 | 2.61 | 0.17 |
| Trial 8 | 30.93 | 34.61 | 0.03 | 0.96 | 0.17 |
| Trial 9 | 33.03 | 35.54 | 0.02 | 0.98 | 0.18 |
| Trial 10 | 32.71 | 30.21 | 0.02 | 0.97 | 0.17 |
| **Average** | **30.60** | **36.16** | **0.03** | **1.16** | **0.17** |

**2.4 Dimensionality Reduced vs Full-Feature Dataset**

PCA analysis is performed on the data in order for dimensionality reduction. Below is the graph of the percentage of variance explained by each feature sorted in descending order. The first three principal components explain 57.04%, 24.04% and 7.6% of the data variance respectively and in total cover the 88.67% of the variance. By discarding other features, the reduced dataset losses approximately 12% of the total data information.

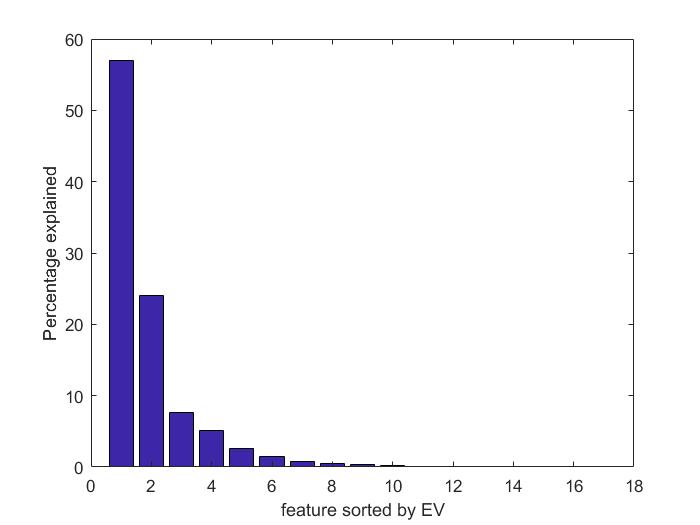


Figure 1: Data Variance-Feature Graph

The effect of dimensionality reduction is consistent across different training techniques. All models suffer from using only the top three features. The training loss increases slightly due to the information loss. For example, the out-of-fold loss for linear regression increases from 61.15 to 65.50 for M label and from 101.94 to 113.85 for T label. The overall performance across different models remains same and the Gaussian processes still have the lowest MSE (Figure 2).

On contrary to the MSE, training time benefits from the dimensionality reduction. Every model trained with reduced dataset takes less time than the models trained with full feature dataset. However, the decrease in training time is not very significant for either linear regression, decision tree or SVM because they are all fast algorithms and decreasing the number of features does not reduce much computation as compared to Gaussian processes. The overall time complexity across different models remains same and the Gaussian processes still have the longest training time (Figure 3).

The results for M label training comparison are illustrated in the graphs below. The results for T label training are similar.

Figure 2 (above): Full-feature vs Reduced Dataset Training Loss for M Label. Result is similar for T label.

Figure 3 (below): Full-feature vs Reduced Dataset Training Time for M Label. Result is similar for T label.

Below is the reduced dataset visualized in PCA coordinates (Figure 4). No linear distribution is directly observed. In fact, the data distribution is much closer to a cluster, which explains why kernel SVM and Gaussian processes have better performance (lower MSE) than linear regression – because they are more flexible models compared to linear regression.

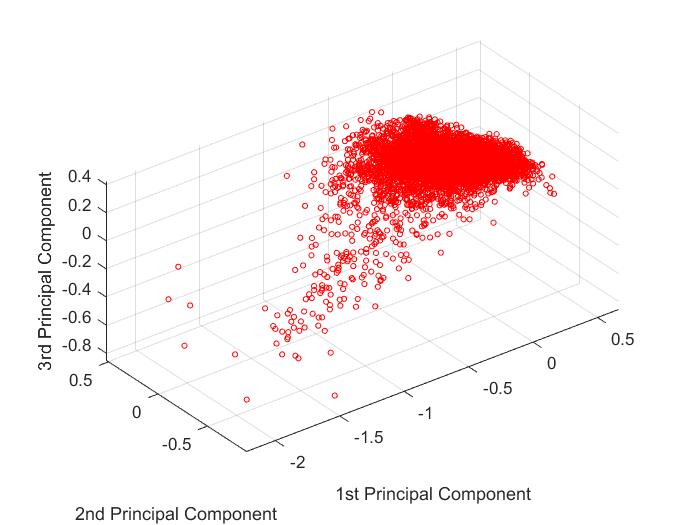


Figure 4: 3-PC Data visualized in PCA coordinates.

**2.5 Model Flexibility**

Theoretically, the model flexibility is measured by how well the model can fit different dataset. Clearly, decision tree is not very flexible as it does not work very well on regression dataset. Linear regression is better but still inflexible because it is limited to linear distributed dataset. SVM and Gaussian processes are flexible as they can fit fancy non-linear functions by applying kernel tricks.

**3. Conclusion**

Among the four machine learning models, Gaussian processes have the best performance for accuracy and worst performance for time complexity. For this particular problem where the model is mapping the voice measurements to the unified Parkinson score, we are particularly interested in understanding and assessing how well the model fits the training data. Even though overfitting can really hurt the prediction performance, we are using cross validation to help reduce overfitting while still aim to fit the training data as much as possible. Therefore from the out-of-fold loss, we conclude that Gaussian process is the best learning model for this Parkinson Score problem.

Both Gaussian processes and SVM apply kernel tricks and hence are capable of finding non-linear patterns in the dataset, while linear regression is limited to the linear model. The advantage of Gaussian Processes is that the kernel hyper-parameters can be automatically learned from the data. Whereas with SVM, this is a big issue and not convincingly solved yet. Additionally, Gaussian processes provide full probabilistic prediction, and an estimate of uncertainty in the prediction, which is more powerful than SVM.

Lastly, decision have the worst accuracy performance because it can create an over-complex model that fits the training data too well to make a good prediction. Pruning the tree or boosting may help the performance but the decision tree is limited as a greedy algorithm – the curse of local optimality.

**3.1 Hypothesis Testing**

The T label has a mean of 29.02 and stand deviation of 10.70. To test our null hypothesis that Gaussian processes and SVM have similar performance, we use the prediction mean as the raw score and the confidence interval is 0.05. The prediction mean for Gaussian process with squared exponential kernel is 27.39 and the Z score is therefore -0.15204. The two-tailed P value equals 0.88 and by the criteria, this difference is considered to be not statistically significant. The prediction mean for SVM with Gaussian kernel is 21.29 and the Z score is therefore -0.72243. The two-tailed P value equals 0.47 and by the criteria, this difference is considered to be not statistically significant either.

Therefore Gaussian processes have similar performance as SVM.

**4. Reflection after Project**

**4.1 Lesson Learned**

Machine learning models are built to solve optimization problems. With different optimizers and approaches, the model trained is of various accuracy. Throughout the milestones, there are three major lessons: 1. Pre-analyzing the data. The dataset I obtained from the UCI database not only contains features such as voice measurements but also other logistics features such as subject ID. My very first linear regression model is trained with all features including four additional logistics ones and the performance is not very satisfying because those logistics features are acting as noise. After eliminating the irrelevant logistics features the model performance improved. Therefore it’s important to pre-analyze your dataset and understand what features your model is trying to study so that the model is trained on correct data; 2. Understand how your data is obtained. PCA and dimensionality are said to be able to remove noise by discarding the least varied features. However, in this particular dataset, voice measurements data is recorded by a high-definition telemonitoring devices that produce minimum noise. Therefore, by understanding how the data is obtained and the device accuracy can help in deciding to what extent to reduce the features to maximize noise removal and minimize information loss in order to train a better model; 3. MSE loss is not the only measurement for model performance. Even though Gaussian processes have a smaller loss than SVM, the difference is not statistically significant. Yet, both models have the same coefficient of determination with , which means they have same fit of the training data. Additionally, there are other performance measures such as training time, which can be very important depends on the cases.

**4.2 Time vs Accuracy Consideration**

Because the dataset selected for this project is rather small, when we compare different algorithms we are more concerned with out-of-fold loss instead of the training time. The conclusion we draw on this project is specific to this particular dataset. In real-world application such as genome sequence classification, gene expression regression, etc., the training data is significantly larger and thus the training time is as an important factor as the model accuracy. There is the tradeoff between accuracy and training time that must be considered when working on a specific problem. Therefore, in future we study how fast the time complexity of each model grows in terms of the training dataset size.

**4.3 Future Work**

Additional to the time-accuracy consideration, there are more aspects to study. If continuing with the same dataset, we can perform hyper-parameter optimization to study the effect of hyper-parameters on Gaussian processes and support vector machine, or train random forest to compare its performance with decision tree. Otherwise, we can retrain all the models on categorical dataset to study how dataset type affects the model performance.