Identification of Grasp Quality Based on Learning Method

Zhifei Zhang
School of Mechanical, Industrial & Manufacturing Engineering
zhanzhif@onid.orst.edu

Yuechuan Chen
School of Mechanical, Industrial & Manufacturing Engineering
chenyuec@onid.orst.edu

Abstract

It is easy to generate a series of possible grasps through a state-of-the-art automatic grasp planner, but it is hard to figure out which grasp is better by the robot itself. A learning-based approach is presented in this paper to solve this problem. In order to develop an algorithm that predicts the quality of a robotic grasp before execution, a large grasp sample data set is collected, including human-planned grasps and automatic grasps. This paper first conducts a thorough statistical analysis of the ability of grasp metrics that are commonly used in the robotics literature to discriminate between good grasps and bad grasps. Then, the principle component analysis is employed to obtain those discriminative metrics, as well as reducing the dimension of feature space. Finally, The Gaussian process algorithm is applied to build a classifier that identifies grasp quality. Moreover, through comparing human-planned and automatic grasps, regions of poor exploration and regions of poor performance will be identified, which can speed the automatic searching of grasp planner dramatically.

I. INTRODUCTION

Learning-based control has been an explosion of work in many areas over the last decade. Especially, robotic control plays an important role in mapping from the state of the arm (position, velocities and accelerations). When it comes to robotic control, the most we concern about is how to identify the good action from those bad ones. Consequently, classification of robot grasp is necessary for future robot learning.

The goal of this paper is to classify the grasps into good and bad classes. In order to record the process of grasping, Force sensors are used to generate twelve parameters [1]. However, there are significant difficulties of these methods. Because of the complicate structure of human hand, it is impossible to make a direct observation of its configuration in grasping experiments. Another problem is limitation of human-planned grasps through teleportation. Most robot hand can only have 6 degrees of freedom, but a human hand has over 22 degrees of freedom.

This paper will try to get over the challenges above to identify the key of human grasping. Gaussian process (GP) is employed to collect the data, which can give a straightforward expression of human's behavior [2]. Through the classification of these grasps, roboticist can analyze the optimization of different features. But neither choosing nor producing the best solution will not be discussed in this paper.

In purpose of addressing these problems, two different approaches can be applied in classification. The first one sets up the class-conditional distribution $p(x|y)$ and the initial probabilities for each class, and then calculates the fitted probability for each class. The other approach focuses on model $p(y|x)$ directly. Because classification problem is a binary (two-class, C=2), the main idea is to transfer the output of a regression model into a class distribution.

The contribution of this paper is to use GP combined with function approximation (mean and covariance functions) to implement classification problems. To reduce the dimensionality of grasping, principal component analysis will be applied in data collecting. Also, supervised learning is used to find how adjustable parameters in covariance functions can be inferred from the training dataset.

In Section 2, the two main techniques to transfer the data and classify them are introduced. Section 3 shows some relevant algorithm compare with GP. The approaches that are applied in the grasp prediction are presented in Section 4. In Section 4, the results of classification are discussed.

II. BACKGROUND

Mainly two techniques will be used to build a grasp quality prediction algorithm. This section provides background information on the two main techniques. First, a grasp sample data is an eleven or twelve dimensional vector, and hundreds of grasp data will form a high dimensional space. Obviously, it's time-consuming to analyze those data directly. Therefore, Principal Component Analysis (PCA) is used for data dimensionality reduction. Then it need to classify the grasp data (processed by PCA) into good grasps and bad grasps, where Gaussian Process (GP) algorithm is employed.

It has be known that those grasp metrics have strong correlations between each other, and some metrics have poor predictive ability. Moreover, correlations between the grasp metrics can lead to poor performance and increase computation costs during training process. In order to deal with this problem, Principal Components Analysis (PCA) is used to perform a dimensionality reduction of our data by reducing the data to only the principal components that

preserve the majority of the data's variance. Then different number of principle components is test to search how many PCs that are kept during leaning process will achieve better performance. In this work, the high dimensional grasp sample data is reduced as low as possible, maybe to three or four dimensions, then project the data into those dimensions and observe how it affects the classifier's performance.

A Gaussian Process (GP) is a non-parametric model that can be used for supervised learning. Specifically, given a set of n training samples $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where x_i is a feature vector and y_i is the output value, the algorithm learns a nonlinear function $f(x)$ that generalizes from the training data in order to predict the output value y for some new data instance x. One of the key benefits of using a GP for predicting grasps is its ability to model nonlinear decision surfaces [3]. A GP is a stochastic process that models a distribution over functions $f(x)$, rather than just modeling a single function $f(x)$ [4, 5, 6]. In our work, each data instance x_i is a grasp, which has k features, corresponding to the k grasp metrics used to represent it. The GP is used to predict a continuous output value that represents the quality of grasp.

III. Related Algorithms

There are several machine learning techniques to implement a binary classification by using a kernel function. Support vector machine (SVM) and neural network—two of the most effective ways to recognize the patterns of a given data set.

A) Support Vector Machine

Support vector machine (SVM) can find a hyperplane to separate the data points to different categories. It can also address non-linear classification by using a kernel function to map inputs into high-dimensional spaces. If the space is defined as D dimensions, the hyperplane called classifier would be D-1 dimensions [7]. The aim of SVM is to obtain the largest distance of adjacent data points. SVM became popular and pragmatic because of its extension of soft margin. Soft margin method provides an increment of the distance between data points. The increment makes it easy to choose a hyperplane which can split the samples. As a common kernel, Gaussian radial basis function can be a useful one used in SVM. It is because the space that Gaussian radial basis function maps input into is a Hilbert space of infinite dimensions [8]. The single parameter γ of Gaussian kernel is the key to regularize the margin classifiers. Various combinations of γ and soft margin parameter can generate different model of classification, which is also the main drawback of SVM [9]. Even though cross validation is applied during processing, it is still hard to get a satisfied result. Some attempt on SVM will be performed in this paper, as well as a compare between SVM and Gaussian process

B) Neural Network

Neural network is a little similar to biological neural networks to some extent. However, modern version of neural network is totally different from biology to adjust itself to a more practical system on signal processing. It builds the connection between inputs and outputs by updating network weights to minimize error. The most difficulty of neural network is the complicate parameters which is hard to interpret. The association of different layers of neurons and the learning process for weights are both challenges for most classification [10].

C) Attempt on SVM

Since the neural network needs to determine many parameters, such as architecture, activation functions and learning rate, it does not a practical solution for the problem in this paper. So, only SVM will be discussed here. The distribution of collected grasp data is shown in Fig. 3.1, where the data are all processed by PCA, and only first two and three principle components are presented.

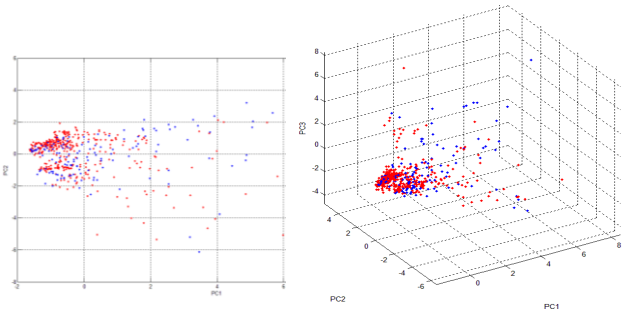


Fig. 3.1 Data distribution in 2D and 3D

(red points are good grasps, blue points are bad ones)

Applying SVM on the data and modifying its parameter, a ROC curve can be obtained to present the performance of SVM, as shown in Fig. 3.2.

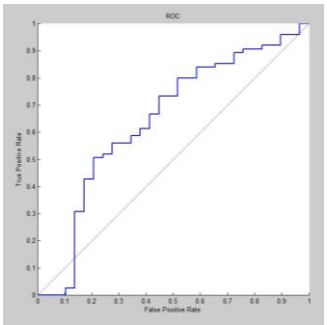


Fig. 3.2 ROC curve of SVM

Usually, a classifier's predictions are used to create a Receiver Operating Curve (ROC) to analyze performance trade-offs. ROC is a common tool used in the machine

learning community for evaluating the performance of a classifier. The shape of the ROC curve indicates how good the classifier is at keeping False Positive Rates (FPR) low and True Positive Rate (TPR) high.

The area under the curve (AUC) value predicts the robustness of a classifier by showing its probability to correctly classify a grasp. An AUC value of 1 indicates perfect performance, and an AUC value of 0.5 indicates that the classifier behaves similar to random guessing.

IV. APPROACH

A) Data Preparation

First of all, grasp data is generated. It can be used for obtain features vector, based on which good and bad grasps are identified. Here, the software named GraspIt, shown in Fig 4.1 is applied to generate grasps automatically through a state-of-the-art grasp planner. Given a common object, such as water bottle, soda can, remoter, and wineglass, the planner can generate tens of ways to grasp it.

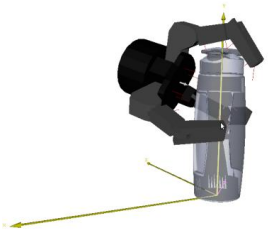


Fig. 4.1 A grasp generated by GraspIt

Then, features of each available grasp is computed according to those raw data, including position and rotation of hand and object, contact point of finger tips, spread of fingers. These features will be the metrics to evaluate the quality of the grasps. Totally, twelve metric are involve in this project, shown by Table 4.1. They have been proposed in prior literature to infer grasp quality [11].

Table 4.1 Metrics to infer grasp quality

Metric	Description
Point Arrangement	proximity of fingertips being in a plane parallel to palm
Triangle Size	Volume of the triangular prism consisting of the finger tips and the palm
Finger Extension	Average finger flexion
Finger Spread	Amount of spread of the fingers
Finger Limit	Extent of finger extensions
Grasp Energy	Distance of hand and object
Parallel Symmetry	Distance between center of mass and contact point centroid perpendicular to object principal axis
Skewness	Alignment of the hand principle axis parallel to the object principle axis
Volume of Object Enclosed	Object volume enclosed by hand normalized by object volume
Wrench Space Volume	volume of grasp wrench space
Perpendicular Symmetry	Distance between center of mass and contact point centroid along object principle axis
Epsilon	Minimum disturbance wrench that can be resisted

So, a grasp is presented by a 12-D vector, which means all input will be displayed in a 12-D space. And hundreds or more grasps need to be collected in the future. Those high dimensional data will make learning slowly no matter which kind of learning algorithm is employed. Moreover, analyzing from the raw data, some metrics have relatively narrow variance, which is obviously disadvantage for identification. Thus, Principle Component Analysis (PCA) is used here to achieve dimensional reduction purpose. After PCA, the first one or more principle components may be selected as input matrix $\mathbf{X} = \{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_n\}$, where $\vec{x}_n = \{x_{n1}, x_{n2}, \cdots, x_{nm}\}^{\mathrm{T}}$ is m inputs for the nth metric. How many components would achieve a better classification performance depends on the experiment result. In this project, maximum n should be 12, but in experiment will only use less than four components.

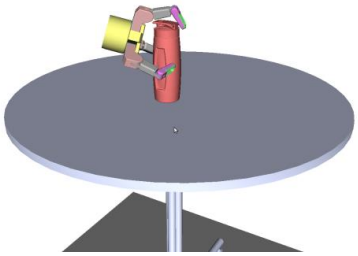


Fig. 4.2 Simulation environment

In order to build a supervised learning loop, quality of those generated grasps need to be given first. The method of obtaining those outputs is shaking test, an experiment from simulation world to real world. Specifically, the grasps will be verified first in a simulation environment to ensure that they are safe and available when executed on a real robot hand. Fig. 4.2 shows the verification in simulation environment (ROS and OpenRAVE).

Finally, the real robot hand and object can be calibrated into the same posture and position as in the simulation environment. The scenario of shaking test is shown in Fig. 4.3, which looks unmatched with Fig. 4.2 because they are not the same grasp exactly. Here just want to illustrate how the prior qualification data is obtained.

Fig. 4.3 Scenario of shaking test (quoted from lab source)

In shaking test, the object will be lifted after grasped by the robot hand, and the robot hand will shake with certain velocity and acceleration. If the hand can hold the object during shaking, the grasp is success. Otherwise, if the object falls down, the grasp is failed. From ten times of repeat test for each signal grasp, a success rate will be calculated.

B) Gaussian Process for machine learning

Gaussian process can be seen as building a distribution model for input features. In a simple word, Gaussian process can present the distribution of a given input's result. Consider $f \in \mathbb{R}^n$ is normally distributed if

$$p(f) = (2\pi)^{-\frac{n}{2}} |K|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(f-m)^T K^{-1}(f-m) \right]$$

for mean vector $m \in \mathbb{R}^n$ and positive semi-definite covariance matrix $K \in \mathbb{R}^{n \times n}$. A multivariate Gaussian, $f(x) = [f(x_1), f(x_2), \cdots, f(x_n)]$, has a multivariate normal distribution for $x = [x_1, x_2, \cdots, x_n]$.

$$f(x) \sim N(m(x), K(x,x))$$

where $m(x)$ is the mean function, which is always set as $m(x) = 0$. And $K(x,x) = [k(x_i, x_j)]_{i,j=1,2,\cdots,n}$ is the covariance function. Usually, squared exponential (SE) kernel is applied here to calculate $k(x_i, x_j)$,

$$k(x_i, x_j) = \sigma_f^2 \exp\left[-\frac{1}{2\ell^2}(x_i - x_j)^2 \right]$$

where amplitude σ_f and lengthscale ℓ are both called hyperparameters, which can be modified to affect learning performance. Now, assume the prior or latent is $f(x) \sim N(m_f, K_{ff})$ and additive noise is $n \sim N(0, \sigma_n^2 I)$, then let $y = f + n$. So it can get

$$p(y, f) = p(y \mid f) p(f) = N\left(\begin{bmatrix} f \\ y \end{bmatrix}; \begin{bmatrix} m_f \\ m_y \end{bmatrix}, \begin{bmatrix} K_{ff} & K_{fy} \\ K_{fy}^T & K_{yy} \end{bmatrix} \right)$$

Then,

$$f \mid y \sim N\left(K_{fy} K_{yy}^{-1} (y - m_y) + m_f, K_{ff} - K_{fy} K_{yy}^{-1} K_{fy}^T \right)$$

The ultimate goal is to predict the output $y*$ when given an input. The function of $y* \mid y$ is similar with that of $f \mid y$, the only difference is that f is replaced by $y*$.

$$y* \mid y \sim N\left(K_{y*y} K_{y*y*}^{-1} (y - m_y) + m_{y*}, K_{y*y*} - K_{y*y} K_{yy}^{-1} K_{y*y}^T \right)$$

During training process, data is gathered and f is updated according to observations. Exactly, what need update is the mean and covariance of f.

According to the multi-dimensional input problem in this project, the Automatic Relevance Determination (ARD) SE covariance function is employed.

$$k(x, x') = \sigma_f^2 \exp\left[-\frac{1}{2} \sum_{d=1}^{D} \left(\frac{x_d - x'_d}{\ell_d} \right)^2 \right]$$

where ℓ_d determines the relevancy of input features. If ℓ_d is very large, the feature is irrelevant.

In this project, GP is used as a classification, so the output should be 0 or 1,

$$p(y = 1 \mid f) = \sigma(f)$$

where $\sigma(f)$ is sigmoid function. In order of obtain the estimation of $y*$ under the condition of y, two steps need to implement. First, it needs to compute the distribution of the latent variable corresponding to test data,

$$p(f* \mid y) = \int p(f* \mid f) p(f \mid y) df$$

where $p(f \mid y)$ is the posterior over f. Secondly, the above distribution is used to produce a probabilistic prediction of $y*$.

$$p(y* = 1 \mid y) = \int \sigma(f*) p(f* \mid y) df*$$

However, $p(f \mid y)$ is intractable. Therefore, Laplace's method is employed here to utilize a Gaussian approximation $q(f \mid y)$ to posterior. Finally, a method to update mean and covariance will be obtained, which is similar to the main idea in regression problem.

As illustrated in Fig. 4.4, the gray area stands for variance. Initially a non-parametric prior over functions is given, which are shown in the upside figure. Then, training data is used to update the mean and covariance of the output corresponding to each input, and obtain the posterior like the downside figure. Therefore, prediction of output can be processed when a new input is given.

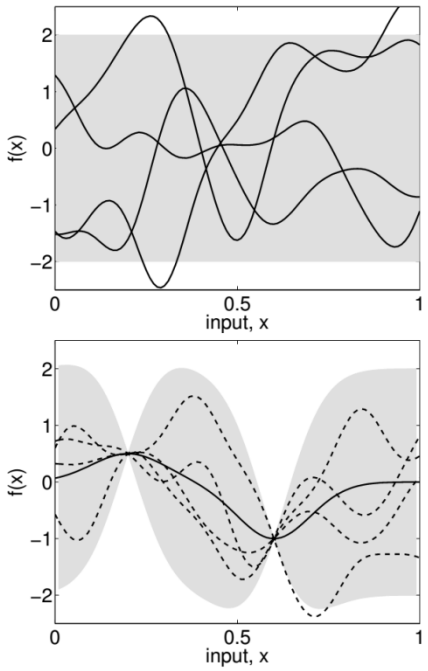


Fig. 4.4 Gaussian process for machine learning

C) Training and Testing

Assume that a Gaussian process classifier has already been built, and all parameters have been set with a right value. The next step is to find a way to training the classifier and verify its performance using the limited data set. Since the process of generating and testing grasps is time-consuming, only hundreds of grasp data is accumulated. Therefore, these data must be fully utilized to achieve a better classification perform.

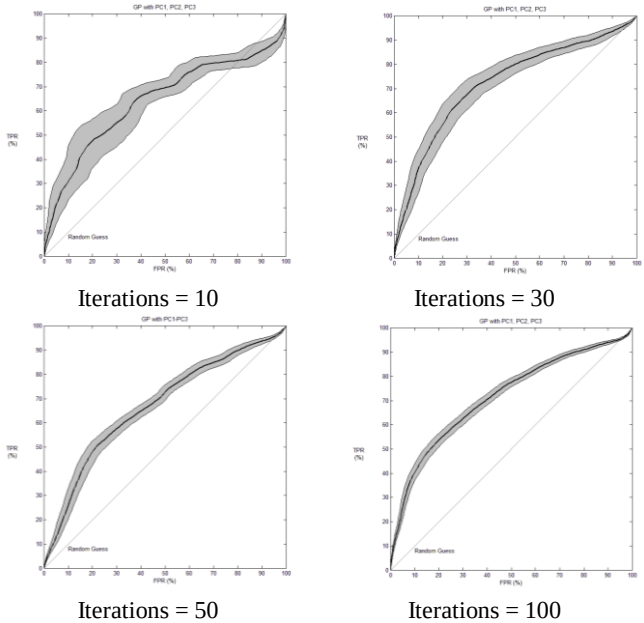
V. EXPERIMENT AND RESULT

In experiment step, all of practical problems need to be considered in, such as how to determine the hyperparameters, how to verify the performance of the classifier, how many PCs should be used as input, computing time, ect.

First, the hyperparameters, which is similar with the kernel parameters in SVM, should be fixed. Unlike SVM, these parameters are much easier to be determined in GP. By minimizing the GP function using conjugate gradients, the hyperparameters will be obtained easily. Then, the only things that needed to be considered are training time and number of input. For almost all classifier, longer training time equals to better performance; however, over-fitting may break this expect at the same time. In order to avoid over-fitting, cross validation is applied to verify the classifier. Then, training time and number of input can be determined through experiments.

A) Training Time

In experiment, PCA is processed on the whole data first, and only the first three principle components (PC) are used to run GP. Currently, only PC1, PC2 and PC3 are processing in GP since the first three PCs will achieve a more performance which will be demonstrated in the following experiment results. Moreover, 80% observations are selected randomly from the raw data as training data, and the left 20% are used as test data. In each iteration, training data will be reselected using the same random rule. More iterations will result in lower variance in prediction, but the mean value will not be improved significantly. As is shown in Fig. 5.1, variance of ROC curve will be narrowed as the iterations increasing; the mean, however, does not change obviously expect becoming smoother.



Iterations = 10 Iterations = 30

Iterations = 50 Iterations = 100

Fig. 5.1 Classification results of different training time

From Fig. 5.1, a significant improvement on variance of ROC curve can be achieved by increasing iterations. However, there is a trade-off relation between iterations and computation time, which is shown in Table 5.1.

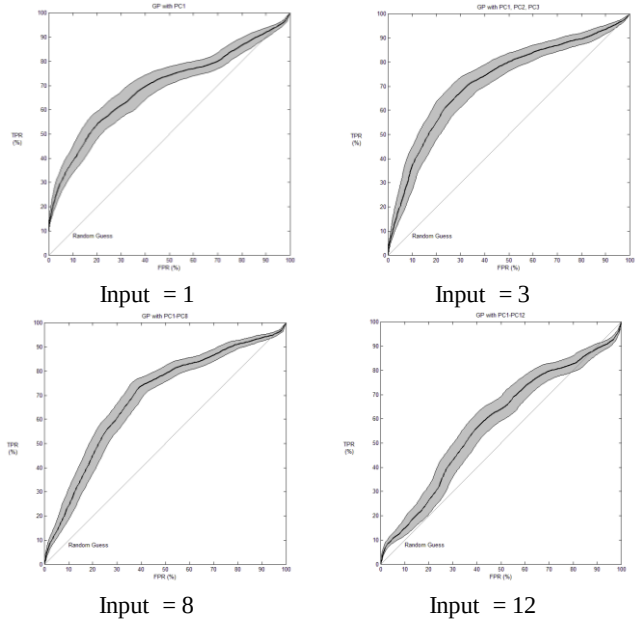
Table 5.1 Compare of different iterations

Iteration	10	30	50	100
Time (s)	50.342	199.822	300.974	620.373
FPR	0.1862	0.2926	0.2178	0.2172
TPR	0.4643	0.6690	0.5034	0.5323
AUC	0.6413	0.7224	0.6689	0.6817
Error	0.0458	0.0353	0.0202	0.0161

Through the table, it is easy to figure out that the AUC will not be improved by increasing iteration. And more iteration will dramatically increase computation time. Therefore, there is a trade-off relation between iteration and computation time. In this project, iteration of 30 is selected according the experiment result.

B) Number of Input

Similarly, experiments on different number of input are processed. For a direct view, some figures are shown in Fig. 5.2. After PCA, all PCs are sorted in descending order according to the eigenvalue.



Input = 1 Input = 3

Input = 8 Input = 12

Fig. 5.2 Classification results of different number of input

The detailed results are shown in Table 5.2. From the experiment result, it can be confidently concluded that more input does not mean better performance. So, three inputs (PC1, PC2 and PC3) should be an optimal option for this project.

Table 5.2 Compare of different number of input

Input	Time (s)	FPR	TPR	AUC	Error
1	68.890	0.2046	0.5449	0.6936	0.0336
2	68.406	0.1992	0.5220	0.6966	0.0404
3	199.822	0.2926	0.6690	0.7224	0.0353
4	247.559	0.1829	0.5525	0.7082	0.0235
5	355.653	0.1453	0.4911	0.6754	0.0212
8	520.016	0.3884	0.7288	0.6895	0.0260
12	754.245	0.4061	0.5697	0.5803	0.0341

C) T-test in Metric Selection

In order to chase a better performance, t-test is employed here to select which three metrics should be used, rather than directly using the first three PCs after PCA.

T-test is a method to detect whether two groups of data obey similar distribution or not. In binary classification problem, more difference between the data of two classes is expected. In this project, t-test is applied on each metrics separately. Here, assume that all metrics are independent. After t-test, each metric will obtain a p-value which present the probability of that two classes within a metric belongs to a similar distribution (hard to be separated). In this project, smaller p-value is expected. Fig. 5.3 shows the p-value of all twelve metrics.

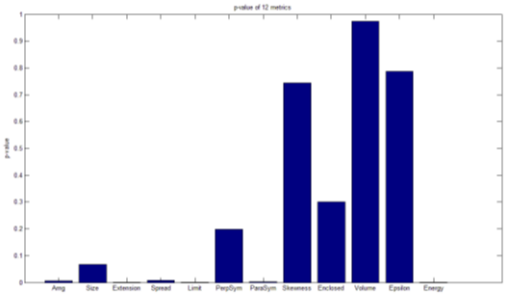
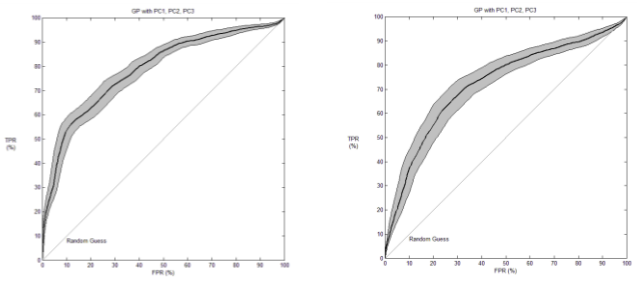


Fig. 5.3 p-value of all metrics

Three metrics with the smallest p-value are selected as the input of GP. But these three metrics still need to be processed by PCA before training the GP classifier. In this case, Extension, Limit and Energy are selected according to p-value. The effect of t-test on the performance is shown in Fig. 5.4.



After t-test Without t-test

Fig. 5.4 Compare of t-test effect

After t-test, AUC can reach 0.7872, which is higher than that without t-test (AUC=0.7224). And TPR reaches 57.58%, when FPR is 13.13%. This result is much better than that without t-test.

VI. CONCLUSION

This paper presented Gaussian Process for classification and a little algorithm about SVM. Due to the complex combination of radial basis function parameter γ and soft margin parameter C, SVM seem not to perform well as GP. Though four different combinations of γ and C need to be check, it is not enough to find an acceptable solution to implement the proper classification. On the other hand, GP get better results with proper training time after t-test. Also, it is not necessary to worry about how to determine and modify parameters when data set changes. So it makes GP more stable and easier to implement. However, GP have to deal with how to handle large matrices [2]. In other words, it will take quite a long time to complete the classification problem when the size of dimension of data is high enough. As a result, a significant limitation of GP is the computational constraints. Another essential issue is the choice of the covariance function. In this paper, Gaussian kernel is the prior selection. Nevertheless, the choice is to some level arbitrary. In fact, a lot of families of covariance kernel functions with different features are applicable to GP.

Further work for this paper would be test more kernel functions to adapt GP giving good predictions. Understanding how other covariance functions with hyperparameters work will provide a potential opportunity for us to learn more about the data classification.

REFERENCE

[1] V. M. Zatsiorsky and M. L. Latash, "Multifinger-prehension: Anoverview," J. Motor Behav, vol. 40, no. 5, pp. 446–475, 2008.

[2] Rasmussen, C. E. "Gaussian Processes in Machine Learning," Advanced Lectures on Machine Learning, Lecture Notes in Computer Science 3176. pp. 63–71. doi:10.1007/978-3-540-28650-9_4, 2004

[3] Alex K. Goins, "Evaluating the Efficacy of Grasp Metrics For Grasp Prediction," ACRA, 2014

[4] C. E. Rasmussen, "Gaussian processes for machine learning," MIT Press, 2006.

[5] Y. Altun, T. Hofmann, and A. J. Smola. "Gaussian process classification for segmenting and annotating sequences," ACM, 2004.

[6] Ed Snelson, "Tutorial: Gaussian process models for machine learning," Gatsby Computational Neuroscience Unit, UCL, 2006.

[7] Aizerman, Mark A. Braverman, Emmanuel M. and Rozonoer, "Theoretical foundations of the potential function method in pattern recognition learning," Automation and Remote Control 25: 821–837.

[8] Lee, Y. Lin, Y. and Wahba, G. (2001). "Multicategory Support Vector Machines," Computing Science and Statistics 33.

[9] Hsu, Chih-Wei, Chang, Chih-Chung, "A Practical Guide to Support Vector Classification (Technical report)," Department of Computer Science and Information Engineering, National Taiwan University.

[10] D. C. Ciresan, U. Meier, J. Schmidhuber, "Multi-column Deep Neural Networks for Image Classification," IEEE Conf. on Computer Vision and Pattern Recognition CVPR 2012.

[11] Ravi Balasubramanian, "Physical Human Interactive Guidance: Identifying Grasping Principles From Human-Planned Grasps," IEEE Transactions on Robotics, vol. 28, No. 4, 2012.8