Human Hand Tracking Regressors

Minjoo Choi*
Department of Industrial Engineering
UNIST
Ulsan, South Korea
minculee@unist.ac.kr

Abstract

A hand pose estimation task was solved by six regressors which were machine learning algorithms, neural network models and a Gaussian prcocess. We will see results of varing hyperparameters in each models and decide the best regressor for the hand pose estimation task.

1 Introduction

A regression problem is to find a continous target value from an input. Many real world problems belong to the regression such as a housing price prediction, a demand forecasting and etc. Machine learning algorithms have regression models and neural networks also do. Here, we want to see the best model or algorithm for Hand pose estimation. Every models has its own inductive bias, so it has a different performance for each data. Deciding the best regressor for Hand pose dataset, we can explore a data space and its representation space.

1.1 Dataset and Problem

We use the ICVL dataset which has a single depth image of a hand and corresponding 63 joint locations. The training dataset has 16,008 images and locations. The testing dataset has 1,596 images and locations. The problem in this project is to predict 63 points describing the hand pose from the single image. Since a location point is float, we need to use regressors.

1.2 Regressors

We introduce six regression models where two are a neural network and four are a machine learning model plus a Gaussian process. Neural Network model contains a multi-layer perceptron (MLP) and a convolutional neural network (CNN). They are all trained by solving the optimisation problem of objective function and stochastic gradient descent. Here, the objective function is a mean-squared error loss (MSELoss). Machine Learning model contains a random forest (RF), a support vector regressor (SVR) and k-nearest neighbour (kNN). The RF is split toward the direction of maximising an information gain. SVR tries to find a hyperplane for maximising the margin of data points. kNN finds the similar data group in trainig data and combine the target data from them. Gaussian Process (GP) is a non-parametric Bayesian regression, which provides a prediction function and a covariance showing confidence of prediction.

*Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Table 1: Data description

Dataset	Number	mean	std
Train	3000		
Validation	150	1881.42	12.29
Test	300		

Table 2: Best Regressors

Model	Specification	Runtime[s]	MSE
MLP	hidden_size=77, n_layers=3, lr=0.1	469.7	0.1018
CNN	channels=[3,3], kernel_size=5, lr=0.1	59.15	0.006178
RF	max _depth=50, max _features=sqrt, min _samples _split=10, n _estimators=3	0.8110	**0.00416**
GP	kernel=DotProduct+Constant, $\alpha = e + 5$	55.69	0.01044
SVR	kernel=rbf, C=0.01, gamma=0.01	59.88	0.006013
kNN	n_neighbors=20	7min	0.005115

2 Experiments

Due to time limitations of our project, we reduce the size of dataset. 3,000 for training, 150 for validation and 300 for testing. This selection made validation MSE lower than training MSE, but we could see the tendency of decreasing training and validation MSE in neural networks model.

We standardise the image pixels by their mean and standard deviation. It was calculated by one mini-batch (batch size 32) of training images. This procedure is needed because the depth map image pixel has a maximum 2,100 value. You can see data description in Table 1.

We test six regressors to predict values. Neural networks are a multi-layer perceptron (MLP) and convolutional neural network (CNN). Machine learning models are a random forest (RF), a Gaussian process, a support vector regressor (SVR) and a k-nearest neighbour (kNN). We select the best regressor for hand pose estimation among models first. And we will see a individual model performence in terms of test mean-squared error (test MSE) and Run-time. In this project, run-time means training time but kNN is testing time. Hyperparameter optimisation will be discussed simultaneously.

2.1 The Best Regressor

Table 2 shows the best performance regressors in each model. The RF regressor is the best regressor to handle hand pose estimation task. And the second one is kNN. It is suprising that the comparatively simple models win a advanced model like CNN. We conjecture that joint locations of hand is a bit simplely related to the specific location of the depth image, so complicated model CNN undefits and requires more training data.

2.2 Regressors Analysis

In this section, we will explain the hyperparmeter optimisation process and see the model performance in terms of test MSE. We used Pytorch 1.10 and scikit-learn 1.1.1 for constructing neural networks and machine learning models respectively. MLP and CNN were trained in setting of 32 batch size and 30 epochs.

2.2.1 Multi-Layer Perceptron

Our basic MLP consists of 3 linear layers. Every hidden size of linear layers has the same for simplicity. See Table 3. You can see run-time decreases when you reduce the hidden size. And the big hidden size which is 144 has sparse space to represent the image if you compare the test MSE of hidden size of 77, 39 and 20 with same setting. But too small hidden size of 39 and 20 also leads to the poor performance prediction because of underfitting. If we increase the number of layers of MLP then test MSE goes up. It also indicates sparse space.

Table 3: Multi-Layer Perceptron

Model	hidden _size	n _layers	lr	Runtime[s]	MSE
	144	0.2	3	980.4	1.957
	77	0.1	3	469.7	**0.1018**
	77	0.2	3	548.4	0.6885
MLP	77	1.0	3	286.5	18.54
	39	0.2	3	286.5	0.1424
	39	0.2	5	286.18	0.379
	20	0.2	3	152.7	0.1601

Table 4: Convolutional Neural Network

Model	channels	kernel _size	lr	Runtime[s]	MSE
	[3,3]	5	0.1	59.15	**0.006178**
	[3,15]	5	0.1	70.39	0.006252
CNN	[15,3]	5	0.1	103.5	0.006225
	[15,15]	5	0.1	115.6	0.007127
	[3,3]	5	0.01	59.44	0.006287
	[3,3]	5	0.5	60.11	0.1753

The very big learning rate like 1.0 disrupts the training process. So the proper selection of learning rate is important in neural network training.

2.2.2 Convolutional Neural Network

Our CNN consitst of 2 convolutional layer blocks which have a convolutional layer, a 2d batchnorm and a 2d maxpool and 3 linear layers to flatten the output. The linear layers are fixed for all experiments for validating a convolutional layer performance.

During the training, CNNs were not trained well after 5 10 epochs.

We check the performance of CNN on hand pose estimation by varing channels size, kernel size and learning rate. First, we fixed the kernel size on 5 and tested the relation of channel size and test MSE. Table 4 shows this situation. Like the MLP, the bigger channel size means sparse representation space so it ruins the performace of CNN. And run-time increases when we increase the size of channels. If you see the last row of Table 4, then the big learning rate also destroys the training.

Next, Table 5 shows the kernel size variation with fixed channel size [3, 3]. The kernel size means filter size of the convolutional layer. Bigger kernel size means a broad field of view in camera. We can notice that the inappropriate kernel size like 3 or 10 affect the test MSE. Obviously, it is due to sparse space or underfitting. The run-time becomes larger as kernel size increases, expectedly.

Since CNN was devised to use in image data, performace is reasonably good. It seems that capturing a contour of hand is important in the depth map image task, but it has no complexity in background image unlike a conventional RGB image task. Hence bigger channel size makes data representation space sparse and leads to uncertainty in prediction.

Table 5: Convolutional Neural Network

Model	channels	kernel _size	lr	Runtime[s]	MSE
	[3,3]	3	0.1	56.20	0.006379
CNN	[3,3]	5	0.1	59.15	**0.006178**
	[3,3]	10	0.1	84.60	0.006243

Table 6: Hyperparameter search range in Random Forest

max _depth	max _features	min_samples _split	n _estimators
[10, 50, 100]	["sqrt", "log2"]	[2, 10]	[3, 5, 10]

Table 7: Random Forest

Model	max _depth	max _features	min_samples _split	n _estimators	Runtime[s]	MSE
RF	50	sqrt	10	3	0.8110	**0.00416**
	10	sqrt	2	3	1.257	0.004534
	50	sqrt	10	5	1.201	0.004698
	50	sqrt	10	10	2.176	0.004725
	50	log2	2	10	0.6759	0.004866
	50	log2	2	5	0.4751	0.004919
	10	sqrt	2	10	2.657	0.005066

2.2.3 Random Forest

We used a scikit-learn random forest regressor. We decided to choose max depth, max features, min samples split and number of estimators as four adjusting hyperparameters. Total 36 models were tested and we selected top 7 MSE models for comparison. Refer to Table 6 for hyperparameter search condition. Before we analyse the result, we alert you that max depth 50 and max depth 100 have same test MSE because of image pixel number boundary (limitation). So we deleted max depth 100 models from analysis.

See Table 7. We conclude that min samples split should be small when the max features parameter is small. This means RF is gauranteed its performce by branching off more precisely if the search interval is small. Here max depth is the same with max features case. The test MSE of RF is the lowest in the other models and run-time is.

2.2.4 Gaussian Process

We chose two parameters which were kernel and α in GP of scikit-learn package. There are two types of kernel, RBF and Dot Product. But Dot Product had experienced a numerical error which could not make a positive definite matrix. So we combined Dot Product kernel with Constant kernel and set a very big α.

If you see Table 8, RBF kernel requires a lot of trainining time. However, Dot Product kernel requires relatively low training time and is good at prediction. It does much better than MLP.

2.2.5 Support Vector Regressor

SVR is a Support Vector Machine (SVM). We selected two or three parameters to adjust. Kernel, C and γ. We changed the parameters but it showed a saturated test MSE. See Table 9. If we want to see the variation of test MSE, then I think we have a more broad search region.

Table 8: Gaussian Process

Model	kernel	α	Runtime[s]	MSE
GP	RBF	1e-5	1023	0.0552
	RBF	1e-2	1036	0.0552
	RBF	1e-1	1044	0.0552
	Dot Product + Constant	1e+0	57.09	0.02918
	Dot Product + Constant	1e+2	45.93	0.01046
	Dot Product + Constant	1e+3	57.83	0.01047
	Dot Product + Constant	1e+4	52.88	0.01046
	Dot Product + Constant	1e+5	55.69	**0.01044**

Table 9: Support Vector Regressor

Model	kernel	C or γ	Runtime[s]	MSE
SVM	linear	C=0.01,0.1,1,10,100	100.9-102.3	**0.006013**
	rbf	C=0.01, $\gamma = 0.01, 0.001, 0.0001$	59.88-61.03	**0.006013**

Table 10: k-Nearest Neighbour

Model	n_neighbors	Runtime[s]	MSE
	5	about 7 min	0.005179
kNN	10	about 7 min	0.005177
	20	about 7 min	**0.005115**

SVR is comparable to CNN. We can think that joint locations of hand has a simple linear relationsip with a specific image pixel to some extent from this result.

2.2.6 k-Nearest Neighbour

The kNN is the best regressor in the hand pose estimation task. Hyperparameter is only one, the number of neighbour. We doubled it. See Table 10. The test MSE is improved when we consider more neighbour but not significant. We missed tracking the testing time so recorded an estimation time.

From the kNN results, we conjectures joint locations are similar among the similar hand poses.

3 Conclusion

In this project, we predicted joint locations from hand depth map images using six regressors. We conclude that the RF is the best regressors to capture the hand pose and relate the representation to locations. The kNN and SVM follows two and third. These produces some conjectures in hand image space.

References

[1] Christopher M. Bishop (2006) *Pattern Recognition and Machine Learning*