Classification of Body Postures and Movements Data Set

CSCI 4364/6364

Fall 2015

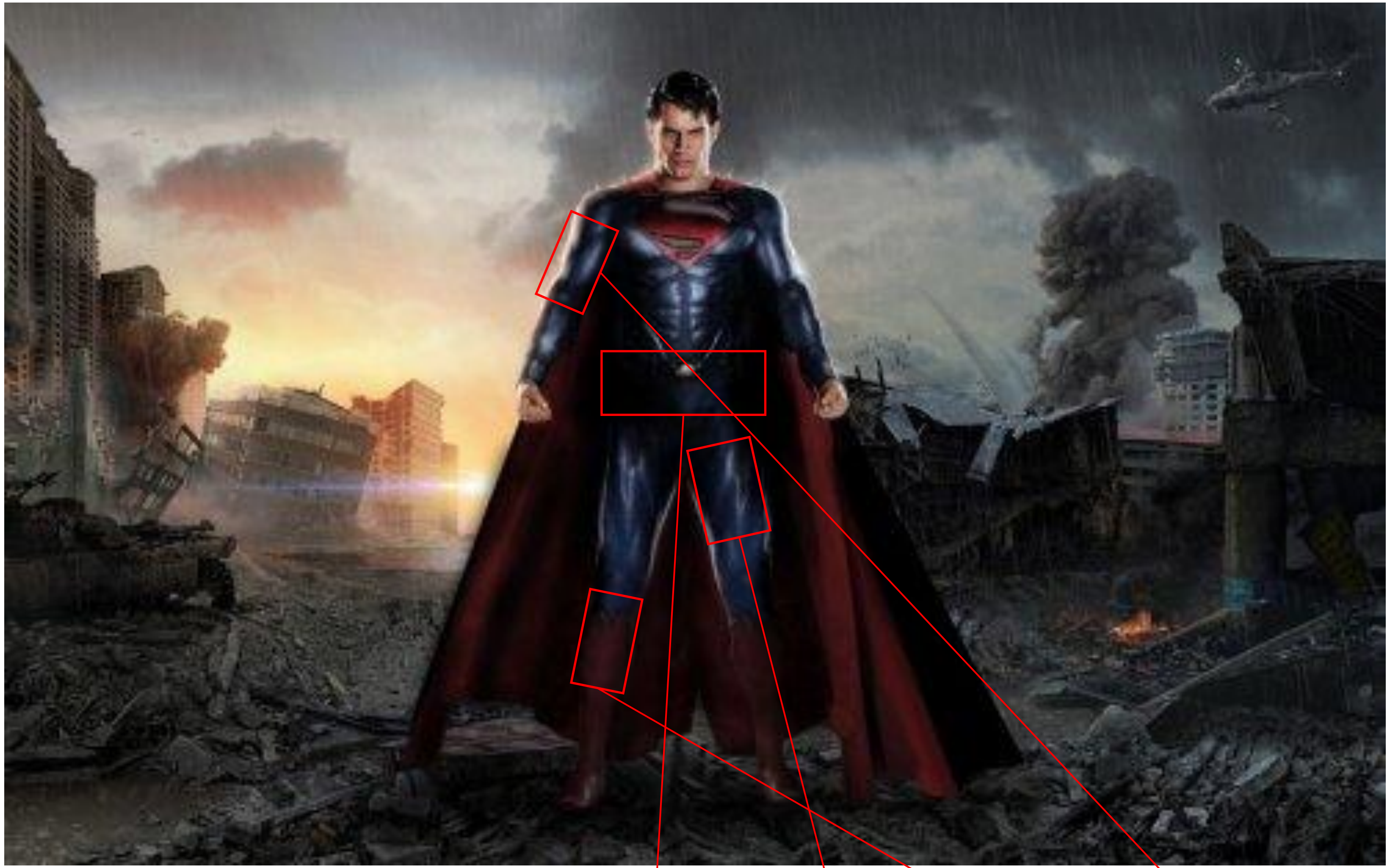
Liwei Gao, Yao Xiao

Purpose of Project

- With the rise of life expectancy and aging of population, the development of new technologies that may enable a more independent and safer life to the elderly and the chronically ill has become a challenge.
- The purpose of the project is to build a model, which uses the data from wearing sensors to predict the body postures and movements of the elder or ill. This would reduce the treatment costs.

Dataset

- Wearable Computing: Classification of Body Postures and Movements (PUC-Rio) Data Set. (UCI Machine Learning Repository)
- The dataset includes 165,632 instances with 18 attributes.
- It collects 5 classes (sitting-down, standing-up, standing, walking, and sitting) on 8 hours of activities of 4 healthy subjects.
- The dataset may be divided into two parts: the information of the subjects (gender, age, tall, weight, body massive index) and data from 4 accelerometers.



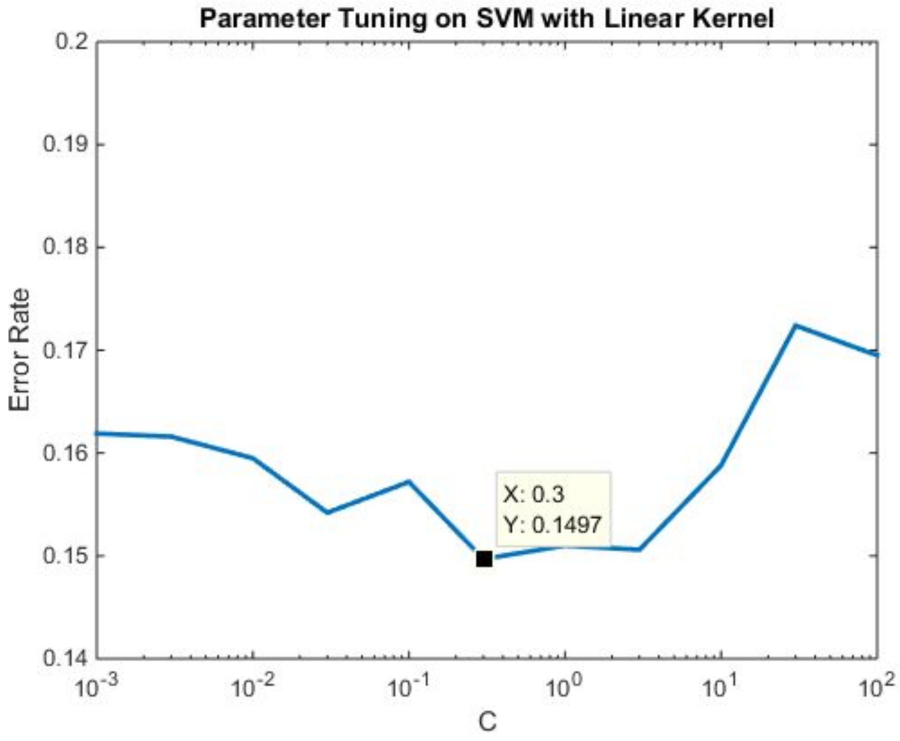
(gender, age, tall, weight, body massive, x1, y1, z1, x2, y2, z2, x3, y3, z3, x4, y4, z4) ------------> class

waist left thigh right ankle right arm

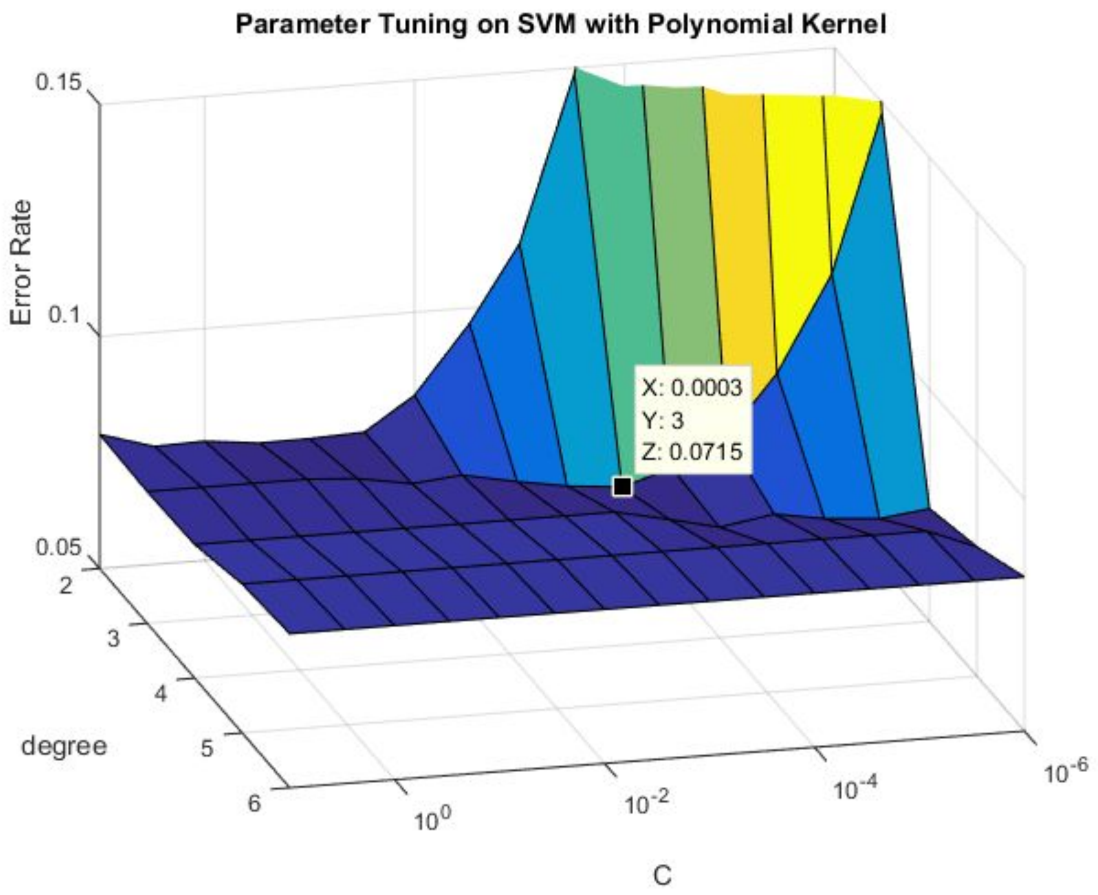
Models of Project

- SVM with Linear Kernel
- SVM with Polynomial Kernel
- SVM with RBF Kernel
- Decision Tree
- Random Forest
- Gradient Boosting (GBDT)
- Neural Networks

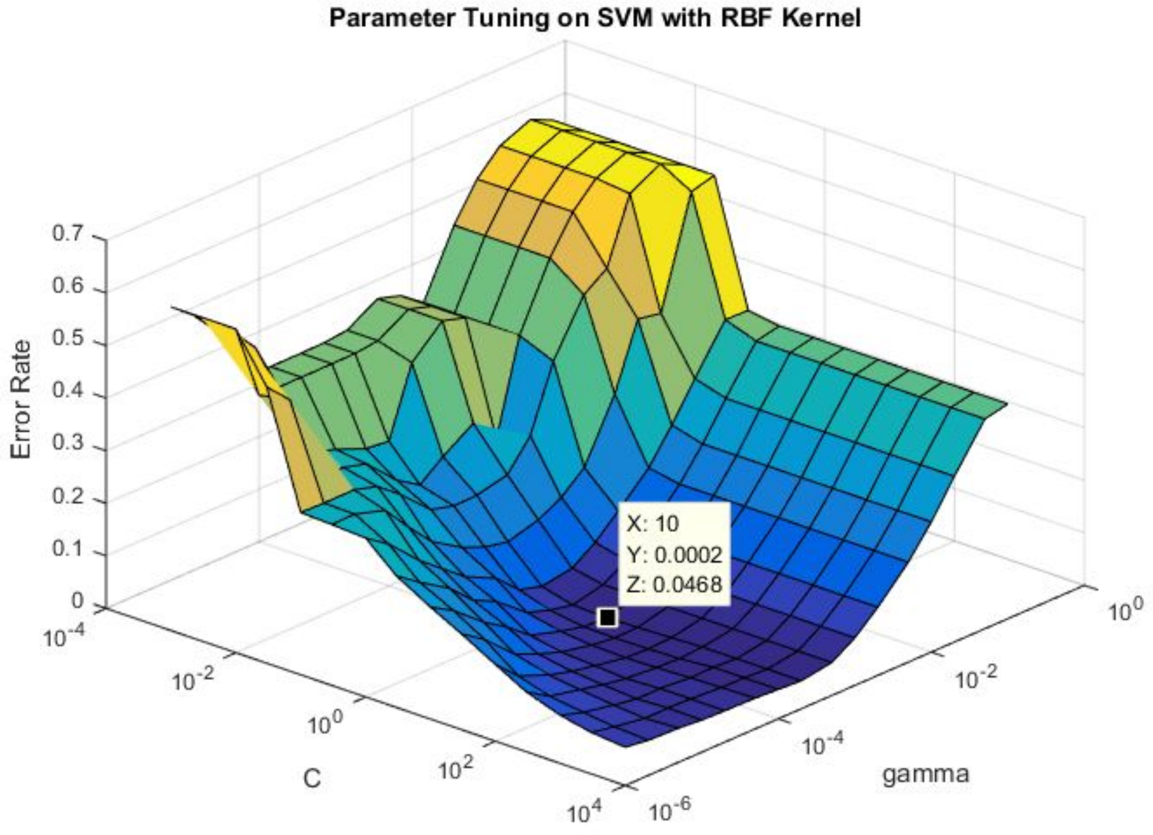
SVM with Linear Kernel



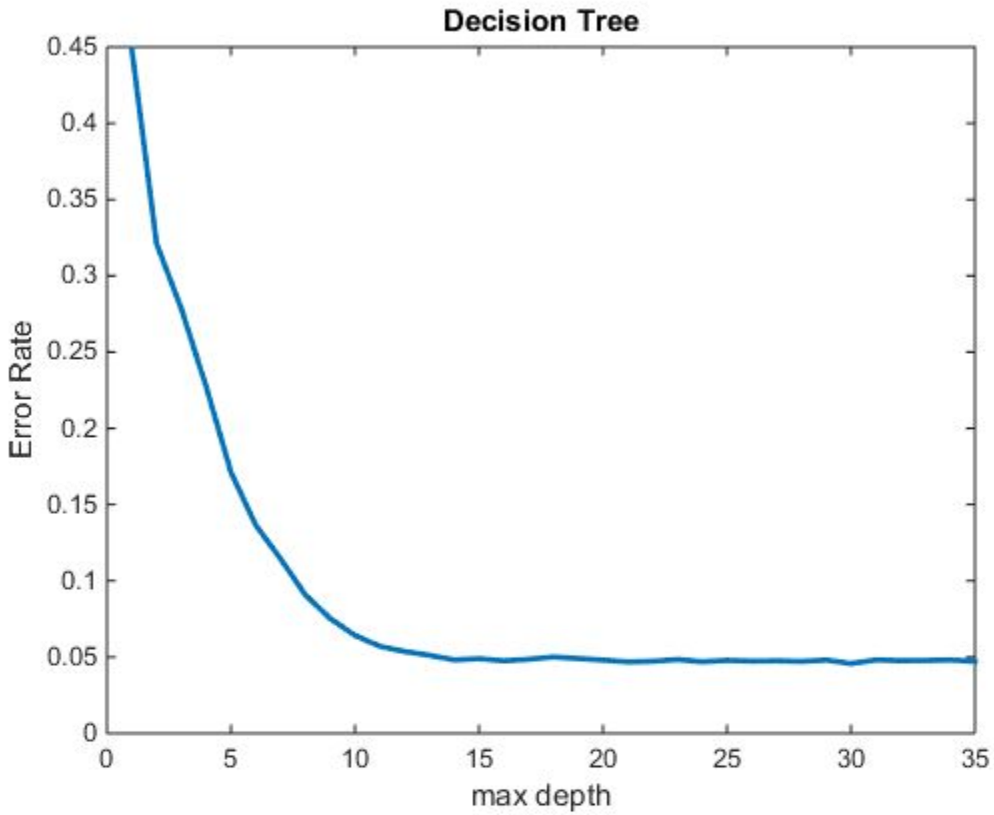
SVM with Polynomial Kernel



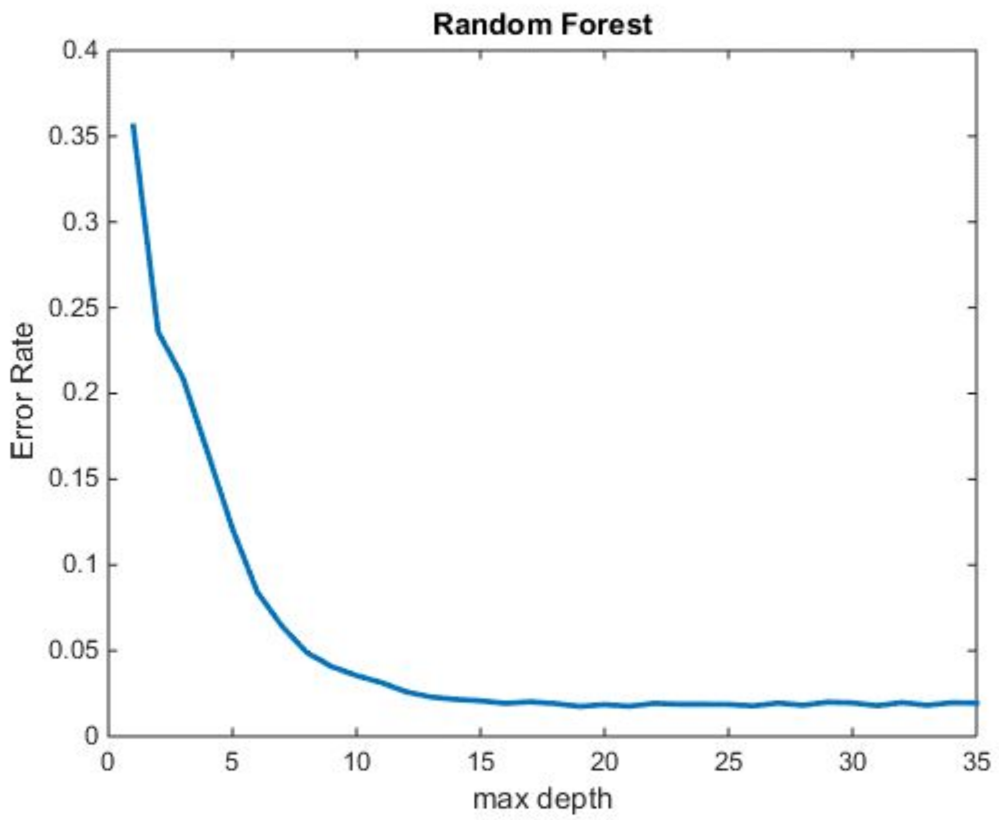
SVM with RBF Kernel



Decision Tree

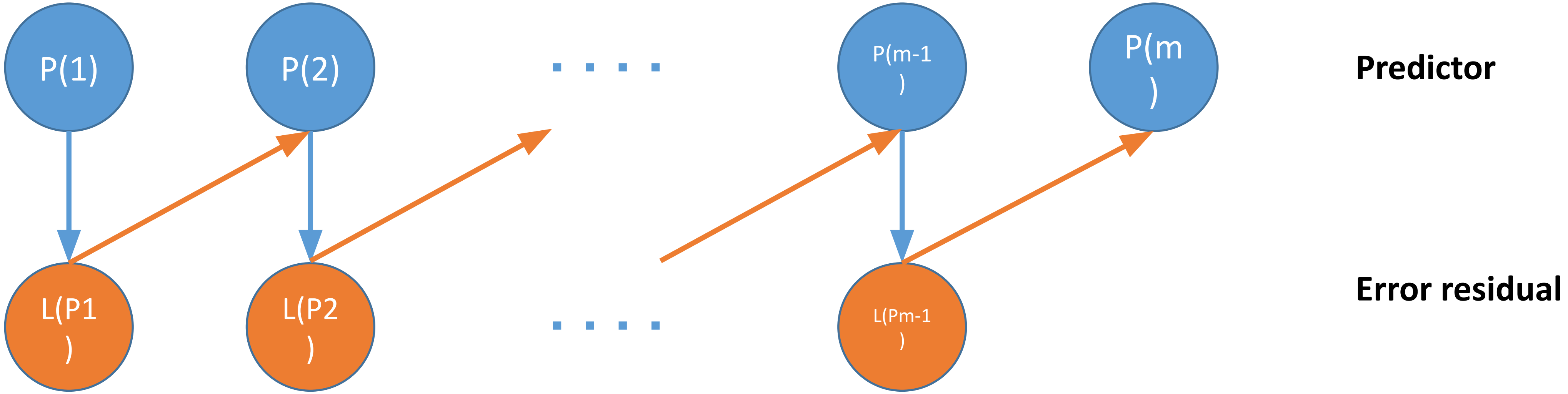


Random Forest



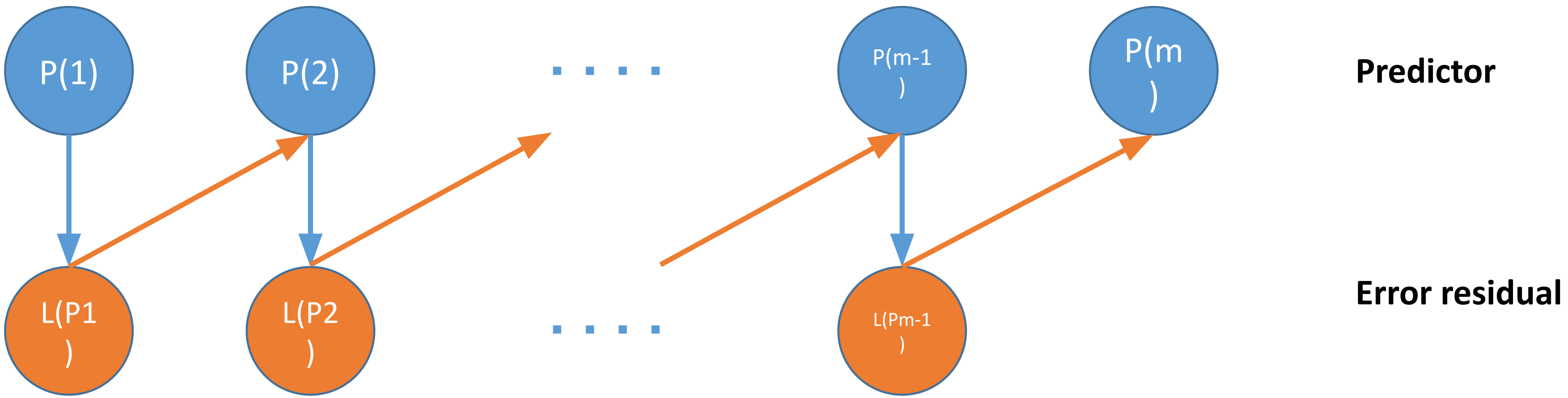
GBDT

- Gradient boosting is a way of boosting, just like Ada boosting.
- However, its idea is that boosting can be interpreted as an optimization algorithm on a suitable cost function.



GBDT

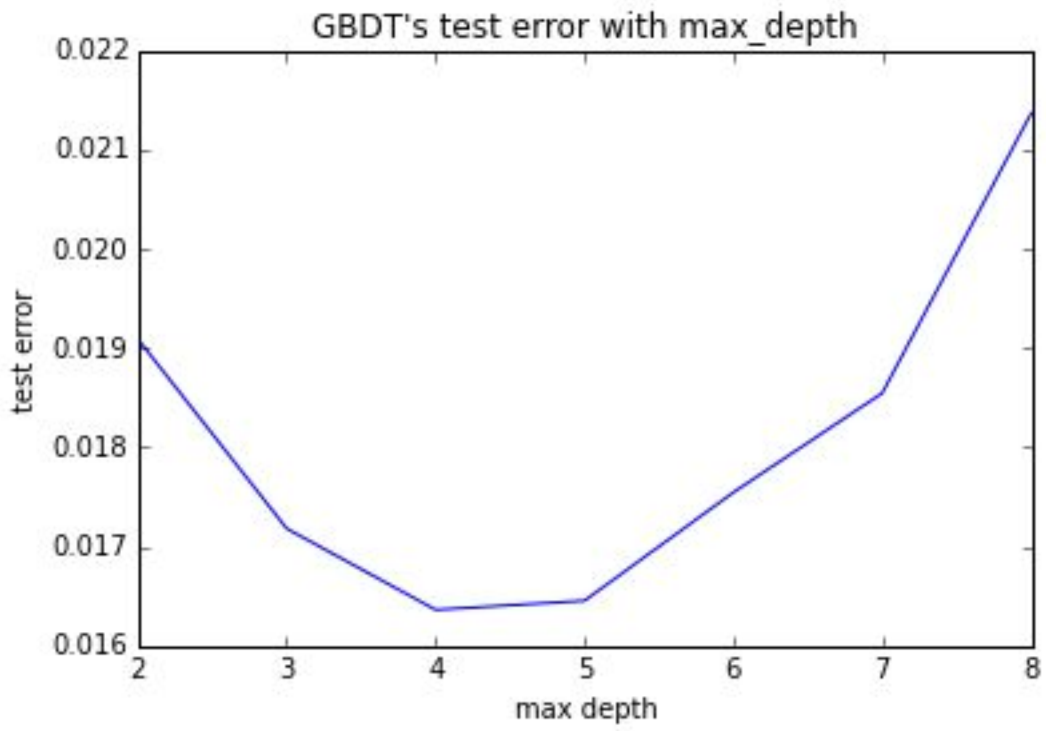
- Review what we learned in Ada boosting. In Ada boosting, we change the weight of points after each training, then we train again.
- In gradient boosting, we compute the loss function(error residual) of each weak learner, which is a function of parameter set P, then do gradient descent for this function and get a better learner. We add these two learner and get new complex learner P2.



GBDT

- There are some important parameters when we use GBDT.
- **n_estimators:** The number of boosting stages to perform. Gradient boosting is fairly robust to over-fitting so a large number usually results in better performance.
- **learning_rate:** It shrinks the contribution of each tree, the bigger the faster (overfit). It is a trade-off with n_estimators.
- **max_depth:** maximum depth of each decision trees. Deep trees are easy to result in overfitting.

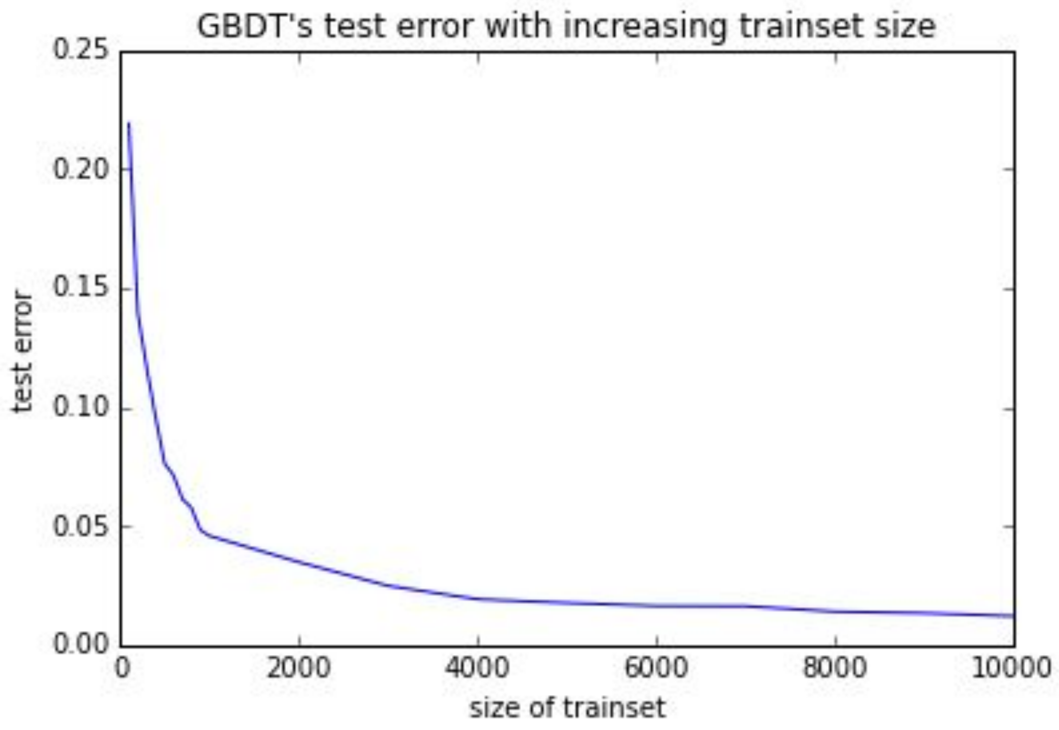
GBDT



We trained the model on different max depths, and we found the best max depth is 4.

GBDT

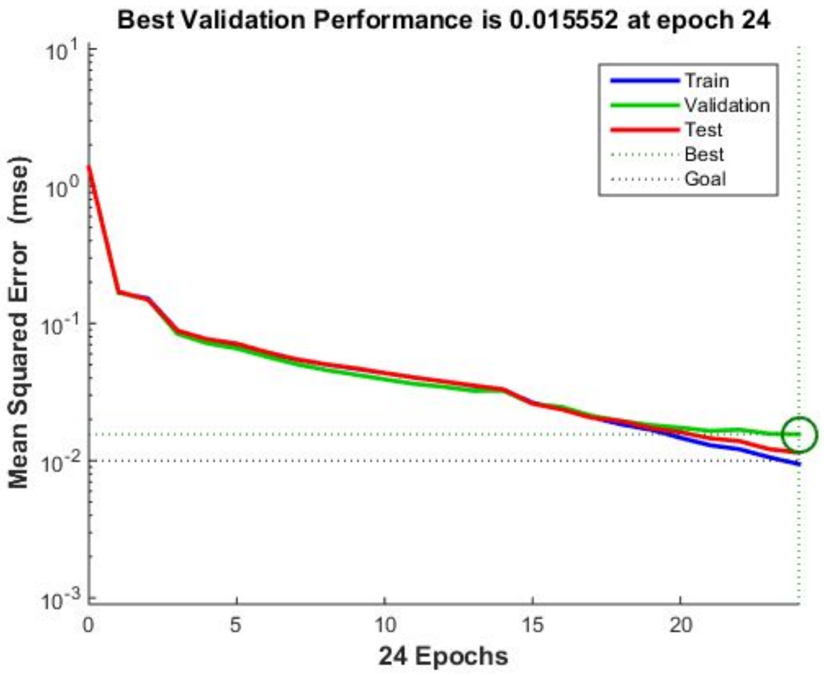
To reduce the time complexity, we also trained the model on different size of train data. And we found the size is over 4500, it doesn't improve the accuracy much.



Neural Network

- Input Layer: 17 features as 17 inputs;
- Output Layer: 5 outputs. (Then take the index of highest output as class);
- Hidden Layer: After several tests, we used three hidden layers (13,11,7).
- Connections: feed forward net.
- Some Advice for Hidden Layer:
  - *The optimal size of a hidden layer is usually between the size of the input and size of the output layers.*
  - *More layers instead of more neurons on each layer.*
  - *1 or 2 hidden layers or use mean of input and output as neuron number can get a decent performance.*

Neural Network

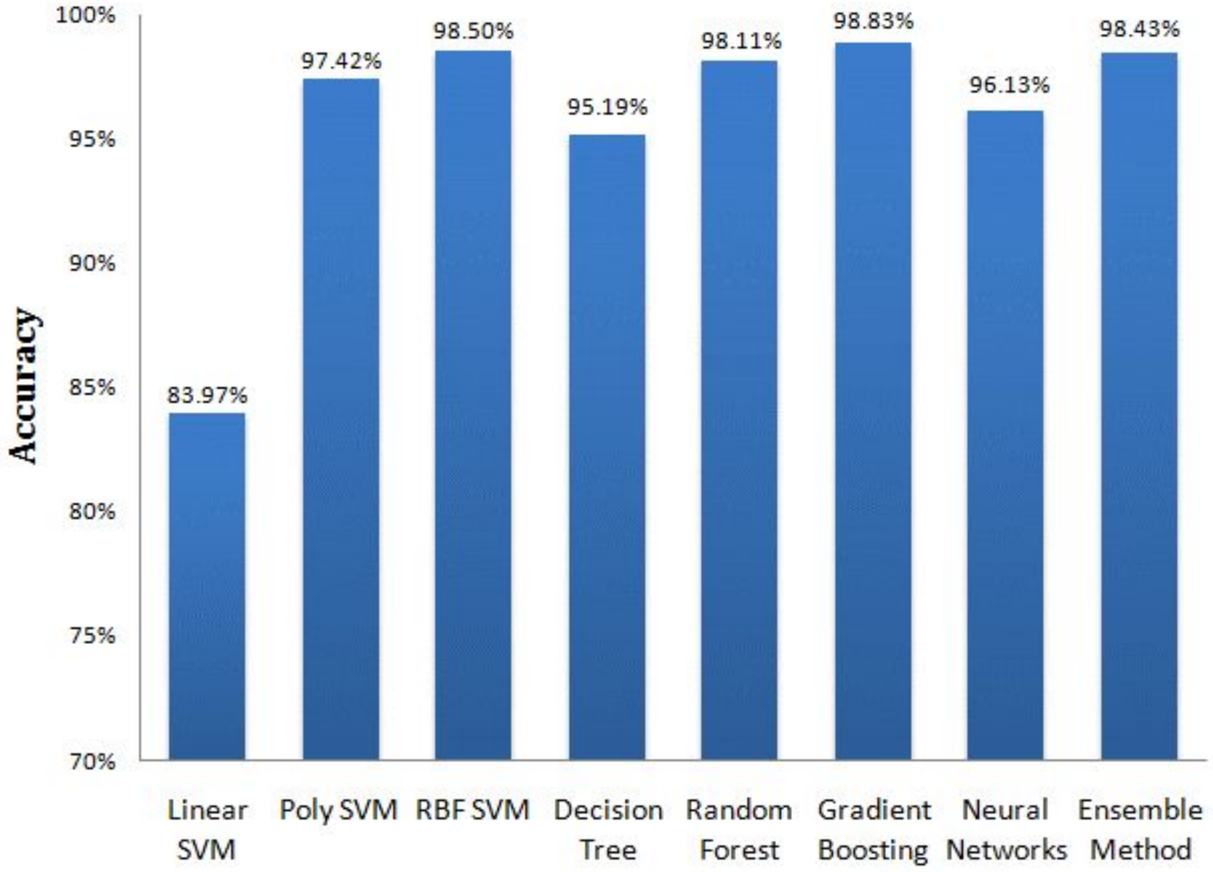


Our NN model has little improvement after 24 epoch in training.

Hidden Layer	Accuracy(%)
11	93.76
14	93.71
(10 9)	95.37
(13 9)	96.06
(13 9 7)	95.86
(14 11 8)	95.93
(13 11 7)	96.35

Some representative hidden layer test shows as table.

Ensemble of Models



- We combined 7 models into a ensemble model.
- The weight of each model is $\frac{1}{\varepsilon^2}$, where ε is the error rate of model.
- We test all three models on same test set (10000 data), the accuracy shows as left histogram.

Thank You!