CS 181 Machine Learning
Practical 1 Report, Team *Advanced Representation*

Jing Wen[1], (Jeremiah) Zhe Liu[1], and (Vivian) Wenwan Yang[2]

[1]Department of Biostatistics, Harvard School of Public Health
[2]Department of Computer Science, Harvard School of Engineering and Applied Sciences

March 3, 2015

1 Exploratory Analysis

The training data set features the gap value (continuous) and 256 pre-extracted features regarding molecular strucuture (binary) for 1,000,000 molecules.

The empirical distribution of energy gap is displayed in Figure 1 (left). As shown, the distribution is unimodel and roughly symmetric, with slightly heavier tail in the positive direction. We thus conclude that the marginal distribution of `gap` approximates a regular Gaussian distribution. Additionally, three negative records were observed. Since energy gap is defined as the difference between highest and lowest occupied molecular orbital, the negative values were considered illogical values and were hence removed from subsequent analysis.

The distribution of extracted features were explored by considering $\mathtt{P}(\mathtt{feature_i} = 1)$ (shown in the textbox below). As shown, 225 out of 256 features have $\mathtt{P}(\mathtt{feature_i} = 1) = 0$ or 1, rendering them completely non-informative in prediction. Among the 31 remaining binary features, we explored the unique combination of feature levels among all subjects. As a result, there exists only 4546 unique combinations among all subjects, with the frequency (i.e. number of subjects with that specific combination of features) of the 400 most frequent combinations visualized in Figure 1 (right). As shown, combination frequency decreased quickly, with the 100 most frequent categories comprising of around 60% of the total data, indicating a sparse input space with highly redundant information.

```
P(feature) = 0.000       Feature Freq: 225
P(feature) = 0.001       Feature Freq: 2
P(feature) = 0.005       Feature Freq: 1
P(feature) = 0.015       Feature Freq: 1
...
P(feature) = 0.976       Feature Freq: 1
P(feature) = 0.996       Feature Freq: 1
P(feature) = 1.000       Feature Freq: 1
```

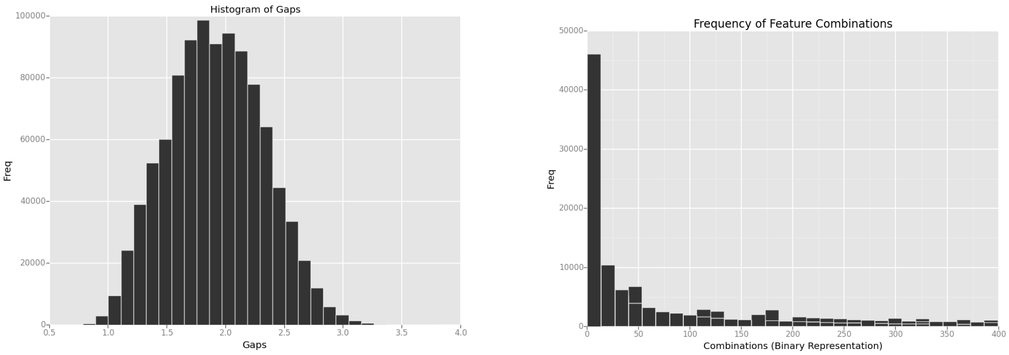

Figure 1: Empirical Distribution of Gaps (left) and the Frequency of unique combination of input features (right)

2 Model Choice and Justifications

Based on observations from previous section, we realize that current learning task possesses below characteristics:

1. Large Sample Size
2. High dimensional, binary Input Space
3. Complicated and unknown Feature-Outcome relation
4. Prediction as the Learning Goal

Based on above characteristics, we considered four classes of popular machine learning strategies as our potential model: regularized linear model (Ridge/LASSO), kernel methods (SVM/Gaussian Process), Bayesian methods, and random forest.

We decided against using **regularized linear models** due to the biased nature of the estimators. For example, recall the form of the likelihood function and solution to Ridge regression:

$$\begin{aligned} L(\mathbf{w}) &= \frac{1}{2}||\mathbf{T} - \mathbf{X}\mathbf{w}||^2 + \frac{\lambda}{2}||\mathbf{w}||_p \\ \hat{\mathbf{w}} &= (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{t} \end{aligned}$$

In order to specify a reasonable linear model, we have to include all possible n-level interactions between the 31 non-trivial features, which will necessarily result in a non-trivial λ during the process of penalization. However, if the target values truely follow linear form, i.e. $\mathbf{t}_{True} = \mathbf{X}\mathbf{w}_{True}$, we have:

$$\hat{\mathbf{t}} = \mathbf{X}\hat{\mathbf{w}} = \mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T * (\mathbf{X}\mathbf{w}_{True}) \neq \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T * (\mathbf{X}\mathbf{w}_{True}) = \mathbf{t}_{True}$$

A simple eigen decomposition will illustrate that such bias increase with the magnitude of λ. We hence decide not to use regularized linear models since the interest lies in unbiased prediction of a potentially complex relationship. In practise, Ridge estimator gives Average Mean square Error (AME) around 0.29, offering inferior performance compared to the other algorithms.

We also decided against using **Kernel-based methods** due to the large sample size. This is because instead of working with the design matrix, Kernel methods working with a $n \times n$ Kernel matrix of the form $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^T$, where $\mathbf{\Phi}$ is a $n \times p$ kernel-transformed matrix of input features. Applying kernel matrix in current scenario implies working with a $1,000,000 \times 1,000,000$ potentially non-sparse matrix, whose computation cost is prohibitive for obvious reasons.

We also chose not to use **Bayesian Methods** due to our lack of prior knowledge with respect to the effect of each features. As a result, Bayesian regression doesn't seem to provide any benefit over regular models.

As a result, we are left to choose **Random Forest** as our algorithm of choice. Introduced by Breiman and Cutler in 2001, random forest is essentially a ensemble method for regression trees. It offers a relatively scalable algorithm which is robust against noisy observations / covariates and powerful in capturing complex interaction between features. This method is traditionally considered having superb performance in terms of prediction accuracy.

2.1 Technical Detail of Random Forest

In this practical, we adopted the implementation `RandomForestRegressor` from `scikit-learn 0.15.1`, which offers a paralleled implementation of the original algorithm described in Breiman (2001). We describe the pseudo code for this algorithm in **Appendix**, and discuss in this section several important parameters in the algorithm:

B (Number of Trees in Forest).
One of the important features of random forest is it does not overfit, i.e. increasing the number of trees in forest does not overfit the data. Indeed, random forest estimate approximate the expectation of "real tree" conditional on the space of bootstrape samples $\hat{f}_{rf}(x) = E_{\Theta(\mathbf{Z})}T(x|\Theta(\mathbf{Z})) = \lim_{B\rightarrow\infty} \hat{f}(x)^{B}_{rf}$ (Hastie et al, 2009). Increase B will lead to a less biased and less noisy approximation of such value.

m (Number of Sampled Features) & $\mathbf{d_{max}}$ (Maximum Tree Depth).
Although increase **B** will only bring estimator toward a constant limit, this limit itself, however, may overfit depending on d_{max} and m. This is because intuitively, too deep a tree indicates a overly rich model, and too many considered feature increase correlation between generated trees. Both may incur unnecessary variance. It is thus crucial to perform parameter selection for **m** and $\mathbf{d_{max}}$ during model fitting.

3 Parameter Selection

3.1 Important Features

As discussed in Section 1 (Exploratory Analysis). Only 31 features were considered important for the purpose of prediction. To validate this point, we first ran a naive random forest with default setting ($m = p, d_{max} = \infty$) and 100 trees, and consider the variable importance metric produced by this fit. Since the variable important for a specific feature is constructed by considering in every tree the change in out-of-bag prediction error by setting the "effect" of this feature to 0, it is considered a cross-validated metric of the contribution of each features in terms of prediction. As a result, 227 features have variable importance $\leqslant 1e-5$ and were thus removed from the subsequent analysis. The 10-fold CV AME of the naive model is 0.272452, and we seek to improve our random forest model based upon this baseline.

3.2 Tunning Parameters

In the next step, we consider the effect of **m** and d_{max} in our prediction performance through 10-fold cross validation. With AME as the outcome metric, we seek to search over a grid of ($\mathbf{m}$, d_{max}) since the convexity of the current problem is unclear. For regression problems, Breiman recommended setting $\mathbf{m} = \frac{p}{3}$ and node size equal to 5, which equals to $\mathbf{m} = 28/3 \approx 10$ and $d_{max} = log_2(1000000/5) \approx 17.6$. In practice, however, larger **m** usually works better for more complex problems. We hence decide to search over the grid $\left[\mathbf{m} \in \{14, 16, ..., 28\}\right] \times \left[d_{max} \in \{16, ..., 24\}\right]$ so that it covers a reasonable range of the theoretically sound values. The AME surface over candidate parameter values are visualized in Figure 2 and Table 1. As a conclusion, we selected $(\mathbf{m} = 19, d_{max} = 28)$. With the selected parameters, we ran a random forest with size **B** = 2000 in order to achieve closest approximation toward the theoretical limit $E_{\Theta(\mathbf{Z})}T(x|\Theta(\mathbf{Z}))$ within reasonable machine computing time (5 hours). The final 10-fold CV AME within training dataset is calculated to be 0.271845.

d_{max}	**m**	14	16	18	20	22	24	26	28
15		0.272426	0.272423	0.272411	0.272421	0.272421	0.272419	0.272420	0.272427
16		0.272404	0.272410	0.272409	0.272410	0.272410	0.272409	0.272411	0.272411
17		0.272408	0.272404	0.272409	0.272405	0.272409	0.272408	0.272404	0.272415
18		0.272407	0.272407	0.272408	0.272410	0.272408	0.272408	0.272409	0.272408
19		0.272416	0.272405	0.272409	0.272412	0.272410	0.272413	0.272411	0.272403
20		0.272406	0.272410	0.272411	0.272408	0.272408	0.272407	0.272414	0.272415
21		0.272410	0.272408	0.272407	0.272409	0.272407	0.272414	0.272417	0.272411
22		0.272407	0.272406	0.272411	0.272407	0.272403	0.272409	0.272418	0.272407
23		0.272412	0.272405	0.272411	0.272409	0.272411	0.272407	0.272412	0.272415
24		0.272407	0.272411	0.272414	0.272410	0.272411	0.272407	0.272410	0.272409

Table 1: 10-fold CV Average Mean Square Error over candidate $(\mathbf{m}, d_{max})$ grids

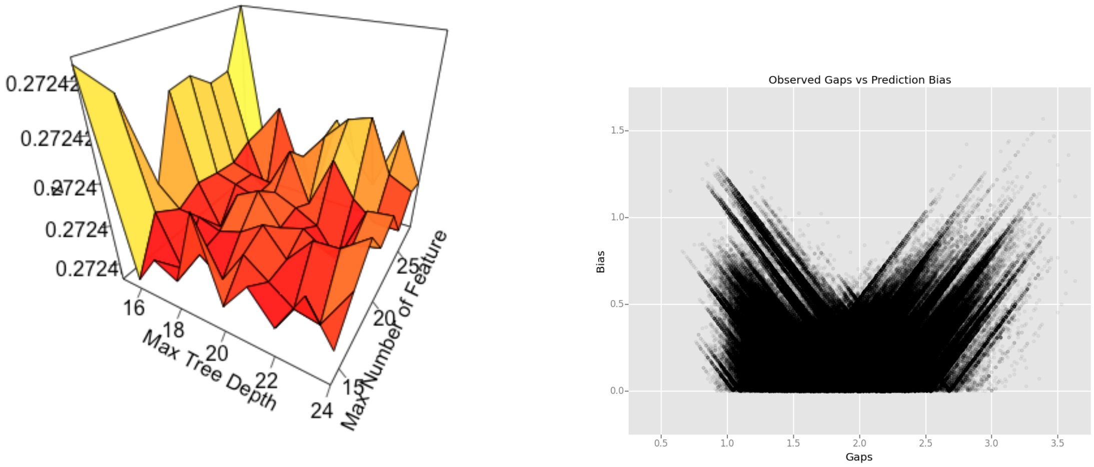

Figure 2: **Left**: 10-fold CV Average Mean Square Error over candidate $(\mathbf{m}, d_{max})$ grids
Right: Observed Gap value v.s. Prediction Bias

4 Model Assessment and Discussion

The observed gap value v.s. prediction bias (absolute value of residual) is visualized in Figure 2 (right). As shown, the prediction bias increases linearly when gap values are far from its mean, which is expected given the limited number of category combinations (4546 unique combinations compared to 1,000,000 observations) of extracted features, which only allowed the random forest to divide the input space into as many as 4546 subspaces and predict using local mean of each subspace, which is why clusters of linearly increasing prediction bias are observed, indicating underfit due to limitation of the input space.

Given the pre-extracted feature provided by this dataset, authors believe that our current model should offer best possible performance in terms of prediction. However, prediction performance may be further improved by extracting more energy-relevant, and preferably continous features from the molecular structure. (e.g. using **RDKit** to extract from the SMILES string), and explore feature-outcome relationship using again random forest to achieve a more satisfactory prediction result. Unfortunately, limited by the time and computing environment available to authors, above procedure was not carried out.

Reference

1. L Breiman, Random Forests, Machine Learning, 45(1), 5-32, 2001.
2. T Hastie. R Tibshirani. J Friedman, ”Elements of Statistical Learning”, Springer, 2009.

Appendix: Algorithm for Random Forest

Input: $\mathbf{Z} = (\mathbf{t}_{N\times 1}, \mathbf{X}_{N\times p})$

Parameter:
- B: The number of trees in the forest
- m: The number of features to sample when splitng nodes.
- d_{max}: The maximal allowed depth of each tree.
- n_{min}: The minimal allowed size the of the splitted node.

Algorithm:

1. For b = 1 to B
    - (a) Draw a bootstrap sample $\mathbf{Z}^*$ of $\mathbf{Z}$
    - (b) (Grow a regression tree T_b using $\mathbf{Z}^*$)
      Untile maximal depth of tree (d_{max})/minimal size of node (n_{min}) is reached, for every node in current tree:
        - i. Randomly sample m features from the p features
        - ii. Find optimal split (in the sense of minimizing RMSE) of current node with respect to select features.
        - iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}$ $b \in \{1, ..., B\}$.
3. Predict for a new point $\mathbf{x}$ as: $\hat{f}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} T_b(\mathbf{x})$.