Parameter Estimation with **MOCK** algorithm

Bowen Deng (bdeng2)

November 28, 2015

1 Introduction

Wine quality has been considered a difficult task that only few human expert could evaluate. However, it is infeasible to give all wine produced a quality label by one of those experts.
In this project, we focus on the task of predicting wine quality from physicochemical properties via data mining. We propose a new algorithm for this task, called **MO**nte **C**arlo **K**ernel (**MOCK**), and compared its performance with ordinary linear square (**OLS**).

1.1 Related work

[2] proposed a neural network approach for predicting human wine taste preferences. [4] proposed a regression tree model for the same problem. Both models obtained pretty good accuracy: 84.7% by neural net, and 88.29% by regression tree.
Regression tree seems a better fit not only because the accuracy is higher, but also because neural network is prone to overfitting given only thousands of examples. On the contrarary, regression tree is not that sensitive to noise, and it can avoid overfitting by cross validation and pruning. However, both models are painful to train, and computationally expensive.

1.2 Data Overview

The data we use is **Wine Quality Data Set** from [1]. The dataset has 4898 examples, with 11 features for each example.
The output feature is quality, which is an integer score between 0 and 10. The input features

Feature	Min	Max	Mean
Fixed acidity (g(tartaric acid)/dm^3)	3.8	14.2	6.9
Volatile acidity (g(acetic acid)/dm^3)	0.1	1.1	0.3
Citric acid (g/dm^3)	0.0	1.7	0.3
Residual sugar (g/dm^3)	0.6	65.8	6.4
Free sulfur dioxide (mg/dm^3)	2	289	35
Total sulfur dioxide (mg/dm^3)	9	440	138
Density (g/cm^3)	0.987	1.039	0.994
pH	2.7	3.8	3.1
Sulphates ((potassium sulphate)/dm^3)	0.2	1.1	0.5
Alcohol (vol.%)	8.0	14.2	10.4

Table 1: The physicochemical data statistics for input features

are fixed acidity, volatile acidity, citric acid, residual sugar, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol. All values are numeric. No missing values is present.

Notice the range of the score is large, we view this task as regression rather than classification. The correlation matrix for the dataset is shown in figure 1.

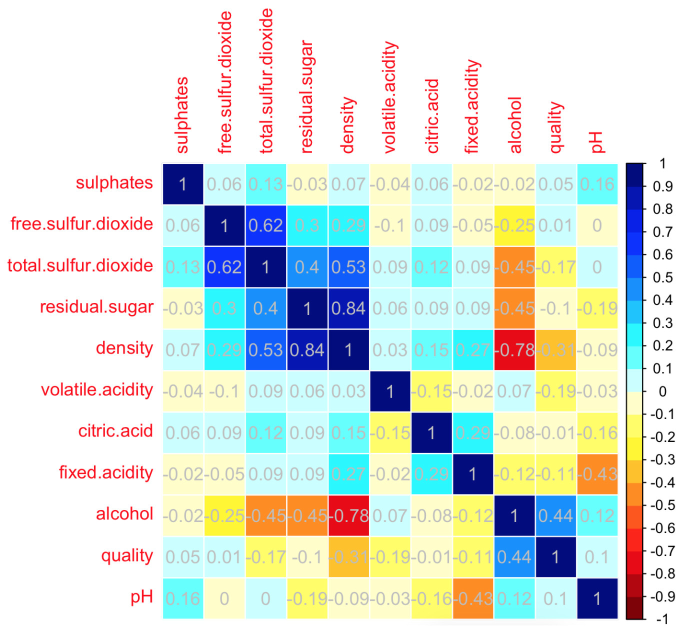


Figure 1: Heatmap for correlation matrix of wine quality data

2 Methods

We first present the general **MOCK** algorithm, then focus on the wine quality prediction problem.
Take Θ' a subset of Θ, and a kernel function $K_\tau : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$.

Algorithm 1 MOnte Carlo Kernel algorithm

```
for $j = 1, \cdots, B$ do
    Sample $\theta(j) \sim \text{Unif}(\Theta')$
    for $i = 1, \cdots, m$ do
        $z_i \leftarrow f_{\theta(j)}(x^{(i)})$
    end for
    $w(j) \leftarrow K_\tau(\begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}, \begin{pmatrix} z_1 \\ \vdots \\ z_m \end{pmatrix})$
end for
$\hat{\theta} = \frac{\sum_{j=1}^B w(j)\theta(j)}{\sum_{j=1}^B w(j)}$
```

Theorem 1. *Assume $\beta \in [-M/2, M/2]^n$, for each $x^{(i)}$, $y^{(i)}$ sampled from $N(\beta^T x^{(i)}, \sigma^2)$. And we use a symmetric kernel function $K_\tau(y, z) = K(\frac{y-z}{\tau})$ satisfying*

$$\int K(x) < \infty$$

then as $M, m \to \infty$,

$$\hat{\beta} \to \beta$$

The proof will be presented in supplementary material.

Remark. These properties hold for common kernel functions including Gaussian kernel and uniform kernel.

For linear model, which we will be using for the regression on wine quality, the algorithm will have the following degenerations.

$$\begin{aligned} f_{\theta(j)}(x^{(i)}) &= \theta(j)^T x^{(i)} \\ K_\tau(y,z) &= \exp\left(-\|y-z\|^2/\tau\right)(\text{Gaussian kernel}) \end{aligned}$$

We split the data into 80% of training examples (3918 examples) and 20% of training examples (980 examples), and we use root of mean square error (**RMSE**) as the evaluation criteria.

3 Results

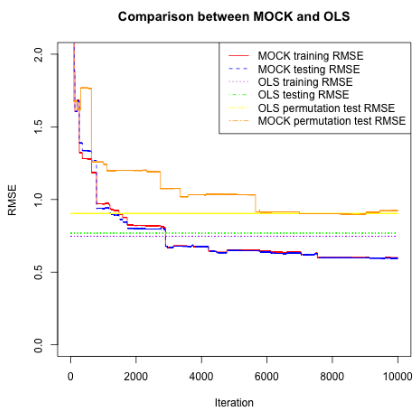
Set $M = 2.0$, $\tau = m$. We implement the **MOCK** algorithm in `R`.
The β we obtained by **MOCK** algorithm is:

Intercept	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	free.sulfur.dioxide
5.89	-0.03	0.01	-0.06	0.02	-0.01
total.sulfur.dioxide	density	pH	suphates	alcohol	
-0.00	0.14	0.12	0.00	-0.02	

Table 2: β obtained by **MOCK** algorithm with $\tau = m = 3918$, $M = 2.0$.

3.1 Comparison

We compare the **RMSE** and runtime of **MOCK** with those of **OLS**.
To prove the accuracy is not by pure chance. We apply permutation test[3]: shuffle the order



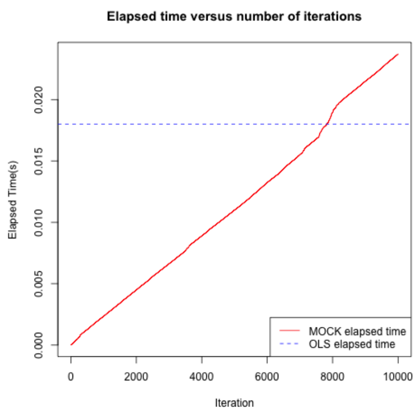


Figure 2: Comparison of **RMSE** (left) and runtime (right) for textbfMOCK and **OLS**

of values for training data, train the parameter, and predict with the perturbed parameter. The result is also shown on the figure.

The runnng time of **MOCK** versus number of iterations is also shown. The running time of **OLS** is fixed as we are using the built-in `stats::lm` function.
For the task of predicting wine quality, the loss of **MOCK** is less than that of **OLS** when iteration number is more than 4000. On the other hand, **MOCK** has an almost linear runtime, which exceeds the time for **OLS** when number of iteration is more than 7500.
There is a clear tradeoff between the accuracy (**RMSE**) and the runtime. However, 6000 is a sweet spot where accuracy is almost converged and it's 18% lower than that of **OLS**, and the runtime is 25% smaller than that of **OLS**.
By looking at the error for perturbed dataset, the **RMSE** obtained by both **MOCK** and **OLS** are significantly lower than that of perturbed dataset. Hence, we can safely draw the conclusion that both algorithms are capturing the signal rather than noise.
We don't have the detailed accuracy result of neural net and regression tree. Therefore, we cannot compare the results with that of theirs. However, the focus of this project is to evaluate the **MOCK** algorithm rather than proposing models that beat the existing work. We believe **MOCK** also work if we modify the model to neural network model, though with more parameter tuning.

4 Conclusion

4.1 Discussion

The **MOCK** algorithm is easy to implement. It converges quickly, and the accuracy is relatively high.
The drawback of **MOCK** compared with **OLS**, is that it requires more parameter tuning. In the wine quality example, we tuned τ (the bandwidth), M (the bound of parameters) for several times to obtain good training error. Another issue is that **MOCK** doesn't provide a p value indicating the significance of each variable as **OLS** does.

4.2 Future work

In this project, we only focus on parameter estimation with linear model. And due to this limitation, we only compared the **MOCK** estimator with **OLS** estimator.
Another extension is to study heuristics to setup τ and M, so users don't have to tune the parameter for a long time. One possible way is to use cross validation. This approach isn't used in this project due to the limited size of the training examples.
As mentioned in Methods section, this algorithm is valid for a very general class of problems, including classification, confidence interval estimation, etc. As a future work, we can apply the **MOCK** algorithm on other interesting datasets, and compare the performance with other approach, including Naive Bayes, logistic regression, SVM, even neural network.
On the theoretical side, we only proved the consistency for linear model, and have not provide probability bound which gives the minimum m required to obtain ε-accuracy. Apart from giving an exact bounds for m, we can also exploit the mathematical proof for the consistency for classification models.

5 Supplementary Material

Proof for theorem 1.

Proof. By law of large numbers,

$$\frac{\sum_{i=1}^m K(z(i), y(i))}{m} \to \mathbb{E}[K(z, y)]$$

Similarly,

$$\begin{aligned}
\frac{\sum_{i=1}^m K(z(i), y(i))\hat{\beta}(i)}{m} &\to \mathbb{E}K(z, y)\hat{\beta} \\
&= \mathbb{E}K(z, y)\beta + X^{-1}\mathbb{E}K(z, y)[X\hat{\beta} - X\beta] \\
&= \mathbb{E}K(z, y)\beta + X^{-1}\mathbb{E}K(X\hat{\beta}, X\beta + \varepsilon)[X\hat{\beta} - X\beta] \\
&\approx \mathbb{E}K(z, y)\beta + X^{-1}\mathbb{E}[(X\hat{\beta} - X\beta)K(X\hat{\beta}, X\beta) + (X\hat{\beta} - X\beta)K_2'(X\hat{\beta}, X\beta)\varepsilon] \\
&= \mathbb{E}K(z, y)\beta + X^{-1}\mathbb{E}[(X\hat{\beta} - X\beta)K(X\hat{\beta}, X\beta)] \\
&= \mathbb{E}K(z, y)\beta + X^{-1}\int_{s = X\hat{\beta} \in X[-M,M]^n} (s - X\beta)K(s - X\beta, 0) \\
&= \mathbb{E}K(z, y)\beta - X^{-1}\int_{s \notin X[-M,M]^n} (s - X\beta)K(s - X\beta, 0)
\end{aligned}$$

The last equality holds because

$$\int_{s\in\mathbb{R}^m} (s - t)K(s - t, 0) = \int_{s\in\mathbb{R}^m} sK(s, 0) = \int_{s\in\mathbb{R}^m} (-s)K(-s, 0) = -\int_s sK(s, 0) = 0$$

The integral diminishes as $\int K < \infty$, and by Cauchy's rule.

$$\begin{aligned}
\hat{\beta} &= \frac{\sum_{i=1}^m K(z(i), y(i))\hat{\beta}(i)/m}{\sum_{i=1}^m K(z(i), y(i))/m} \\
&\to \frac{\mathbb{E}K(z, y)\beta}{\mathbb{E}K(z, y)} \\
&= \beta
\end{aligned}$$

□

References

[1] A. Asuncion, D. Newman, UCI Machine Learning Repository, University of California, Irvine, 2007 `http://archive.ics.uci.edu/ml/datasets/Wine+Quality`

[2] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. *Modeling wine preferences by data mining from physicochemical properties.* Decision Support Systems, Elsevier, 47(4):547-553, 2009.

[3] Fisher, R.A. *The Design of Experiments.* New York: Hafner. 1935.

[4] Michal Horak. *Prediction of wine quality from physicochemical properties.* Data mining, Czech Technical University in Prague. 2009/10.