Random KNN

E. James Harne[1], Shengqiao Li[2], and Donald A. Adjeroh[1]

1. West Virginia University
2. University of Pittsburgh Medical Center

ICDM 2014

Outline

Challenges and Possible Solutions for High-dimensional Data

- Challenges:
  - Small n, large p
  - Irrelevant features
  - Hard to build predictive models directly
- Solutions:
  - Variable (Feature) Selection:
    - Variable filtering (statistic defined over populations): Easy to compute and fast, but not necessarily good for the final predictive model.
    - Wrapper methods (model wrapped in a search algorithm): Slow and not scalable, but the final model might be good,
  - New Modeling Methods
    - Random Forests

Overview

Random KNN consists of an ensemble of base k-nearest neighbor models, each constructed from a random subset of the input variables.

Random KNN can be used to select important features using the RKNN-FS algorithm. RKNN-FS is an innovative feature selection procedure for "small n, large p problems."

Random KNN (no bootstrapping) is fast and stable compared with Random Forests.

The rknn R package implements Random KNN classification, regression and variable selection algorithms.

Advantages of Random KNN

- KNN is stable, no hierarchical structure
- Final model can be a single KNN (vs. many trees)
- Local method: robust for complex data structure
- Automatically re-train, incremental learning
- Easy to implement

Random KNN Properties

Symbol	Definition
r	number of KNN classifers
m	number of features used for each KNN
M	multiplicity of a feature in r KNN's; $M \sim B(r, m/p)$
S	number of silent features (features not drawn); $P(S = s) = \binom{p}{s} \sum_{j=0}^{p-s} (-1)^{p-s-j} \frac{\binom{p-s}{j}\binom{j}{m}^r}{\binom{p}{m}^r}$
R	number of KNN's until $S = 0$
I_f	indicator variable, 1 if feature f is silent; $P(I_f = 1) = P(M = 0) = (1 - \frac{m}{p})^r$
r_f	number of KNN's until f is drawn
ν	$\nu = E(M) = \frac{rm}{p}$
λ	$\lambda = E(S) = p(1 - \frac{m}{p})^r$
η	coverage probability; $P(S = 0) = \sum_{j=0}^{p} (-1)^{p-j} \binom{p}{j} \left[\binom{j}{m} \Big/ \binom{p}{m} \right]^r$

Coverage Probability

If we ignore the dependency among I_f's, and approximate S by a binomial random variable $B(p, (1 - m/p)^r)$, then η_b is an upper bound of η given by:

$$\eta = P(S = 0) < \left[1 - \left(1 - \frac{m}{p}\right)^r\right]^p = \eta_b.$$

If we approximate S by a Poisson random variable, then

$$\eta_p = e^{-\lambda},$$

The value of r may be determined by inverting the binomial or Poisson approximation of η, respectively, as follows:

$$r_b = \frac{\ln(1 - \eta^{1/p})}{\ln(1 - m/p)}$$

or

$$r_p = \frac{\ln(-\ln \eta) - \ln p}{\ln(1 - m/p)}.$$

Time Complexity of Random KNN

- KNN: $O(2^p kn \log n)$
- Random KNN: $O(r2^m kn \log n)$

If $m = \sqrt{p}$, then the complexity is $O(r2^{\sqrt{p}} kn \log n)$. Since the exponent is much smaller than that for the ordinary KNN method, Random KNN is expected to be much faster for high dimensional data. If we use $m = \log p$, we obtain a complexity in $O(rpkn \log n)$,

which is linear in p, the number of variables.

Golub Leukemia Data Classification

- The Leukemia data set is available in package "golubEsets".
- Number of genes: $p = nrow(Golub\_Train) = 7129$.
- We choose $m = 55$ for each KNN.
- The function r can be used to compute r, the number of KNN base classifiers. We set coverage probability $\tilde{\eta} = 0.999$, $r = r(nrow(Golub\_Train), eta = 0.999) = 1332$.

```
> require(rknn)
> require(genefilter)
> require(golubEsets)
```

Classification Results

```
> options(width=40)
> golub.rnn<- rknn(data=golub.train, newdata=golub.test,
+ y=golub.train.cl, r=1332, mtry=55, seed=20081029);
> golub.rnn

Call:

rknn(data = golub.train, newdata =
golub.test, y = golub.train.cl,
r = 1332, mtry = 55, seed = 20081029)

Number of neighbors:  1
Number of knns:  1332
No. of variables used for each knn:  55
Prediction:
 [1] ALL ALL ALL ALL ALL ALL ALL ALL ALL
[10] ALL ALL ALL ALL ALL ALL ALL ALL ALL
[19] ALL ALL AML AML AML AML ALL AML AML
[28] AML AML AML ALL AML AML AML
```

Classification Confusion Matrix

```
> confusion(golub.test.cl, fitted(golub.rnn))

classified as-> ALL AML
            ALL  20   0
            AML   2  12
```

Two cases are misclassified.

Variable Ranking

- A measure, called *support*, is defined to rank the variable importance.
- The *support* of a feature is the average accuracy of the base classifiers containing the feature.
- The R function `rknnSupport` is used to compute feature supports:

Variable Suppot

```
> golub.support<- rknnSupport(golub.train, golub.train.cl,
> golub.support

Call:

rknnSupport(data = golub.train, y =
golub.train.cl, k = 3)

Number of knns:  500
No. of variables used for each knn:  55
Accuracy: 0.9473684
Confusion matrix:

classified as-> ALL AML
            ALL 27   0
            AML  2   9
```

Variable Support Plot

```
> plot(golub.support, main="Support Criterion Plot")
```

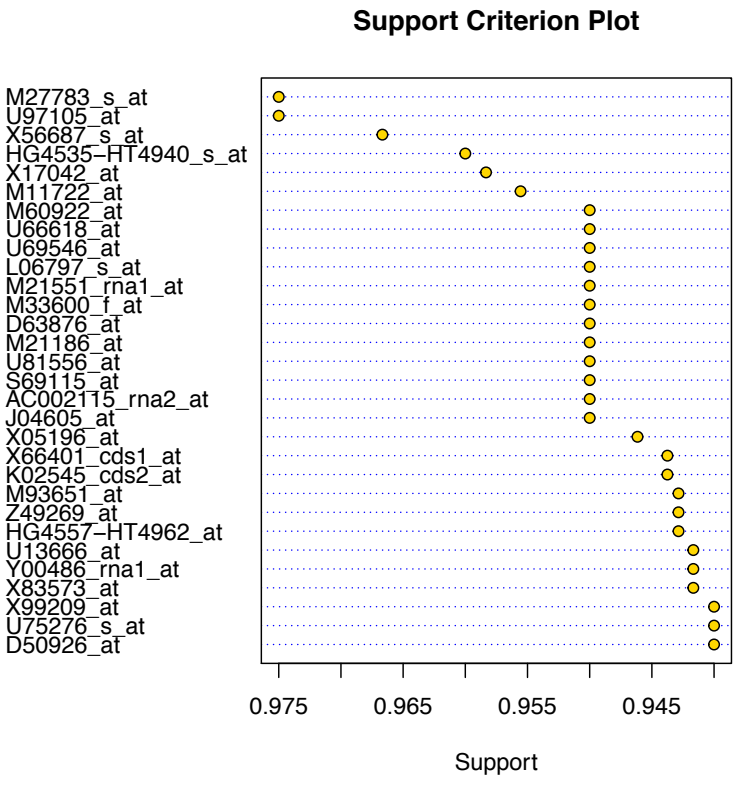


Figure: Support plot for Golub leukemia training data

Two Stage Multi-step Variable Elimination

- Stage I: A fixed portion of the input variables are removed (e.g. 50%) at each step;
- Stage II: A fixed number of variables are removed (e.g. 1) at each step;
- Balance between performance and speed.

Stage I

```
> set.seed(20081031)
> golub.beg<- rknnBeg(golub.train, golub.train.cl);
> plot(golub.beg)
```

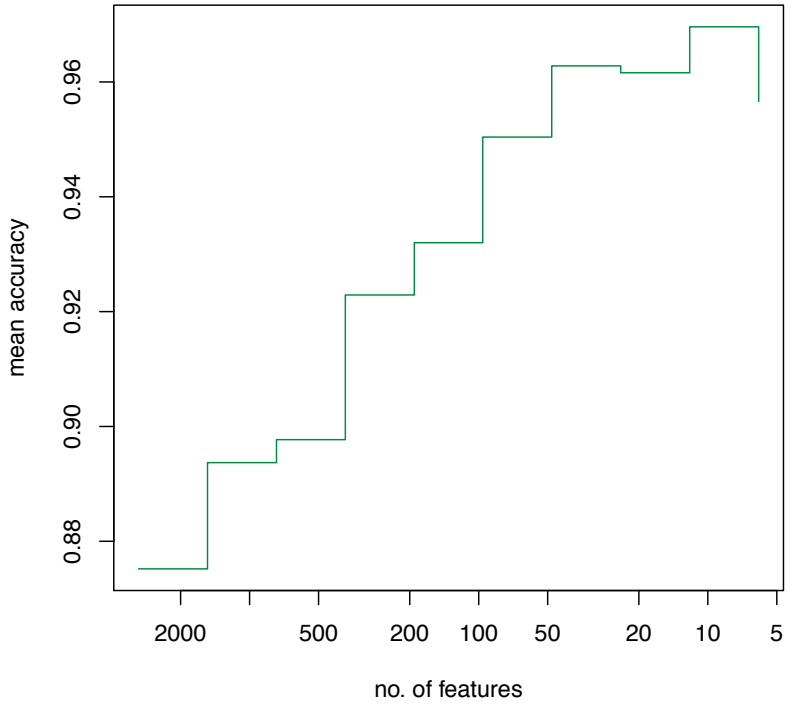


Figure: Mean accuracy change with the number of features for Golub leukemia data in first stage

Stage II

```
> better.set<- prebestset(golub.beg);
> golub.bel<- rknnBel(golub.train[,better.set], golub.trai
> plot(golub.bel, ylim=c(0.88, 1))
```

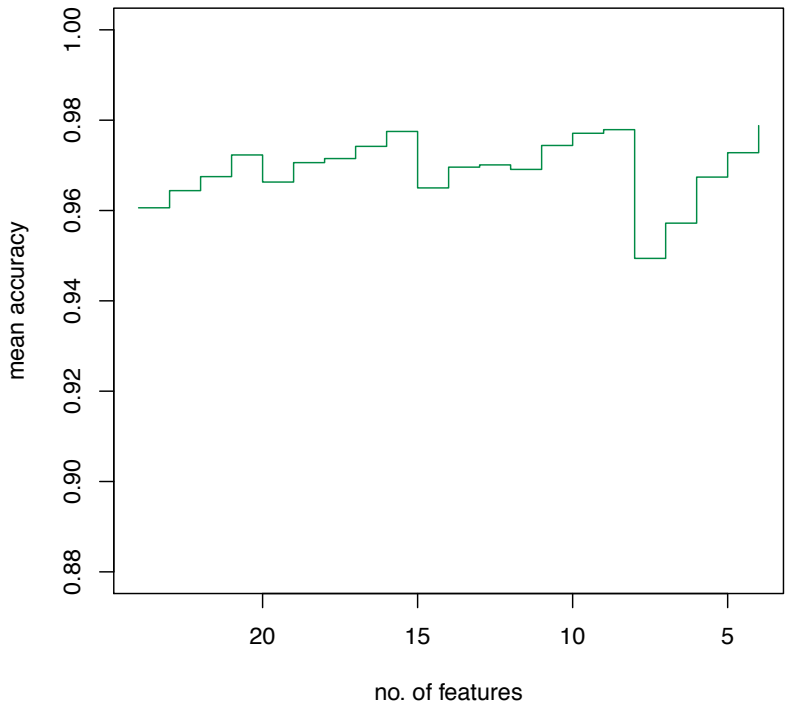
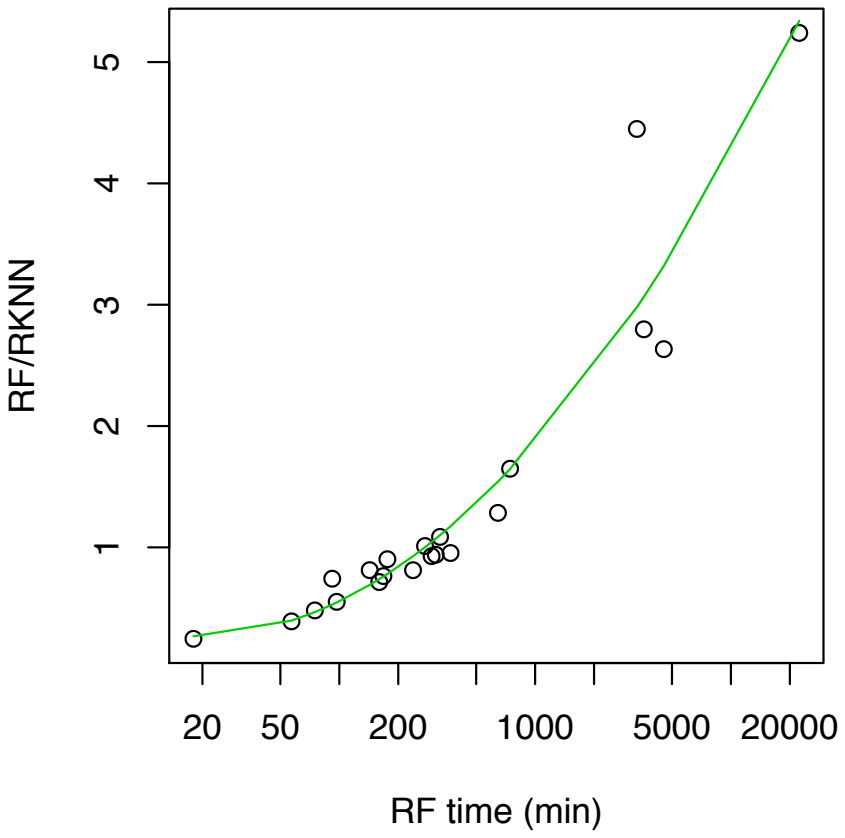


Figure: Mean accuracy change with the number of features for Golub leukemia data in second stage

Speed Comparison with Random Forests

Random KNN approach for feature selection is faster than Random Forests:



Stability Comparison with Random Forests

Random KNN approach for feature selection is more stable than Random Forests:

Table: Average selected gene set size and standard deviation

Dataset	$p * c/n$	Mean Feature Set Size			Standard Deviation		
		RF	R1NN	R3NN	RF	R1NN	R3NN
Ramaswamy	1267	907	336	275	666	34	52
Staunton	859	185	74	60	112	12	11
Nutt	829	146	49	49	85	6	4
Su	792	858	225	216	421	9	26
NCI	688	126	187	163	118	41	33
Brain	666	18	137	120	13	42	42
Armstrong	468	249	76	73	1011	16	12
Pomeroy	329	69	89	82	70	15	13
Bhattacharjee	310	33	148	146	29	15	10
Adenocarcinoma	260	8	38	11	4	20	11
Golub	222	12	27	21	8	5	5
Singh	206	26	25	13	32	6	6
Average		220	118	102	214	18	19

Regression

```
> require(chemometrics)
> data(PAC)
> x<- scale(PAC$X);
> PAC.beg<- rknnBeg(data=x, y=PAC$y, k=3, r=500, pk=0.8)
> plot(PAC.beg)
```

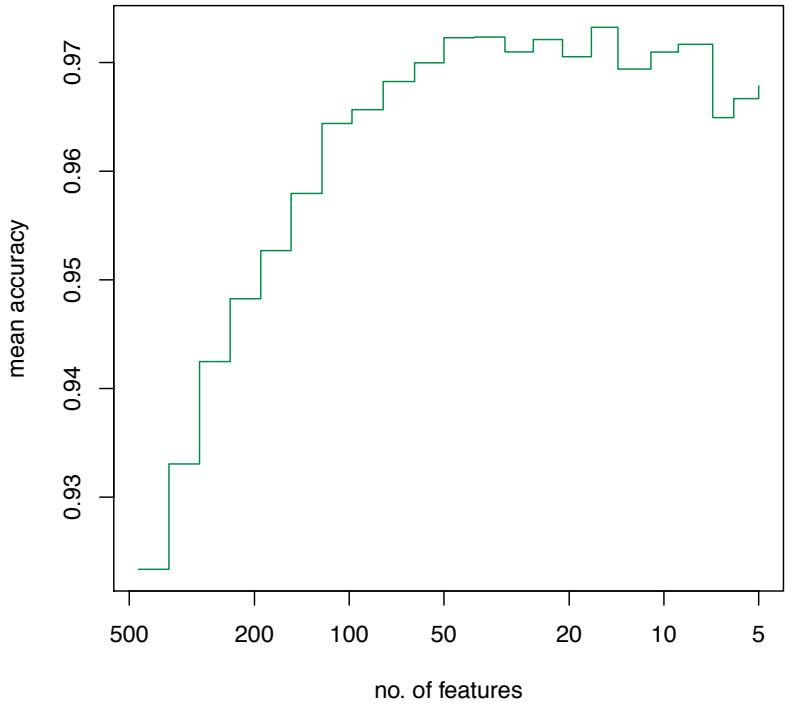


Figure: Mean accuracy change with the number of features for PAC data

Regression Performance

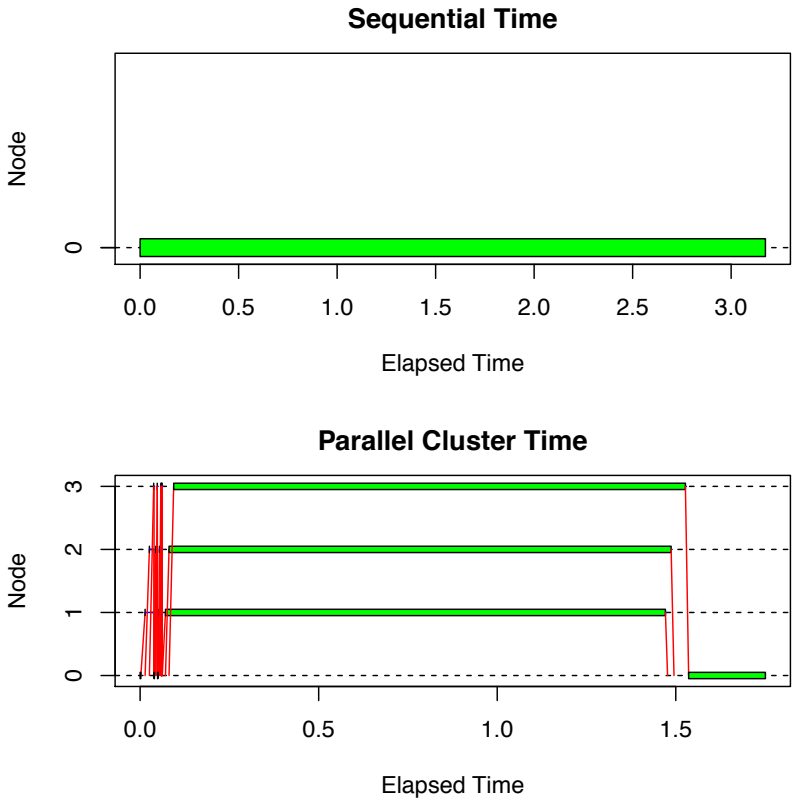
Random KNN can be easily extended to regression problems.

```
> knn.reg(x[,bestset(PAC.beg)],  y=PAC$y, k=3)
$call
knn.reg(train = x[, bestset(PAC.beg)], y = PAC$y, k = 3)

$k
[1] 3

$n
[1] 209

$pred
  [1] 207.6700 206.6733 206.6633
  [4] 215.2267 210.3400 226.4467
  [7] 226.2733 211.9933 210.0233
 [10] 224.6267 219.9700 224.6933
 [13] 208.4367 218.9567 223.6600
```

Parallel vs Sequential

```
> require(snow)
> require(parallel)
> sequential_time<- snow::snow.time(golub.srknn<- rknn(data
> confusion(golub.test.cl, golub.srknn$pred)

classified as-> ALL AML
            ALL  20   0
            AML   2  12

> cluster<- parallel::makeCluster(detectCores()-1);
> parallel_time<- snow::snow.time(golub.prknn<- rknn(data=g
> confusion(golub.test.cl, golub.prknn$pred)

classified as-> ALL AML
            ALL  20   0
            AML   2  12

> stopCluster(cluster)
```

Parallel vs Sequential



Extensions

1. How do we estimate entropy of complex molecules, e.g., proteins? Combine Approximate KNN with Random KNN?
2. How do we estimate a gene interaction network? Combine the Dempster-Shafer induction network algorithm with Random KNN?