Modernizing k-Nearest Neighbor Software

Robin Elizabeth Yancey

Bochao Xin

Norm Matloff

Dept. of Computer Science
University of California, Davis

URL for these slides (repeated on final slide):
`http://heather.cs.ucdavis.edu/SDSSslidesKNN.pdf`

Notation and Acronyms

Notation and Acronyms

- n: number of data points in our training cata

Notation and Acronyms

- n: number of data points in our training cata
- p: number of predictors/features

Notation and Acronyms

- n: number of data points in our training cata
- p: number of predictors/features
- ML: machine learning (= nonparametric regression)

Notation and Acronyms

- n: number of data points in our training cata
- p: number of predictors/features
- ML: machine learning (= nonparametric regression)
- k-NN: k-nearest neighbor method

Notation and Acronyms

- n: number of data points in our training cata
- p: number of predictors/features
- *ML*: machine learning (= nonparametric regression)
- *k-NN*: k-nearest neighbor method
- *RFs*: random forests

Notation and Acronyms

- n: number of data points in our training cata
- p: number of predictors/features
- *ML*: machine learning (= nonparametric regression)
- *k-NN*: k-nearest neighbor method
- *RFs*: random forests
- *SVMs*: Support Vector Machines

Notation and Acronyms

- n: number of data points in our training cata
- p: number of predictors/features
- *ML*: machine learning (= nonparametric regression)
- *k-NN*: k-nearest neighbor method
- *RFs*: random forests
- *SVMs*: Support Vector Machines
- *NNs*: neural networks

Overview of k-NN

Overview of k-NN

- Like all ML methods, does smoothing. $\widehat{E}(Y \mid X = t) =$ average Y among the k-nearest datapoints to t.
- Earliest ML method, e.g. (Fix and Hodges, 1951).
- Later, largely displaced in popularity by RFs, SVMs, NNs.

Overview of k-NN

- Like all ML methods, does smoothing. $\widehat{E}(Y \mid X = t) =$ average Y among the k-nearest datapoints to t.
- Earliest ML method, e.g. (Fix and Hodges, 1951).
- Later, largely displaced in popularity by RFs, SVMs, NNs.
- Still common in some apps., e.g. recommender systems, outlier detection.

Overview of k-NN

- Like all ML methods, does smoothing. $\widehat{E}(Y \mid X = t) =$ average Y among the k-nearest datapoints to t.
- Earliest ML method, e.g. (Fix and Hodges, 1951).
- Later, largely displaced in popularity by RFs, SVMs, NNs.
- Still common in some apps., e.g. recommender systems, outlier detection.
- And has some real advantages:

Comparison of Various ML Methods

Comparison of Various ML Methods

method	tuning pars. (fewer better)	iterative? (no better)	unique sol'n.?(yes better)
k-NN	k	no	yes
RFs	depth, leaf size, split crit. etc.	yes	no
SVM	d, C	yes	yes
NNs	"∞"	yes	no

Improved k-NN

- So, k-NN has the virtues of being simple, e.g. only 1 tuning parameter, and computationally attractive.

Improved k-NN

- So, k-NN has the virtues of being simple, e.g. only 1 tuning parameter, and computationally attractive.
- We believe that, with improvements, k-NN can be quite competitive with other methods.

Improved k-NN

- So, k-NN has the virtues of being simple, e.g. only 1 tuning parameter, and computationally attractive.
- We believe that, with improvements, k-NN can be quite competitive with other methods.
- Two Innovations, one methodological and one diagnostic:

Improved k-NN

- So, k-NN has the virtues of being simple, e.g. only 1 tuning parameter, and computationally attractive.
- We believe that, with improvements, k-NN can be quite competitive with other methods.
- Two Innovations, one methodological and one diagnostic:
    - Assigning different distance weights to different predictors.
    - Exploring locally-determined values of k.
    - This talk will focus on the first innovation.

Different Distance Weigts for Different Predictors

Different Distance Weigts for Different Predictors

- E.g. done in (Han *et al*, 2001) for cosine “distance” for text clasification. Optimization is performed.
- Here we’ll use (weighted) Euclidean distance.

Empirical Examples

Empirical Examples

- Will use the **regtools** package (on CRAN, but latest at *github.com/matloff*).
- Over 50 tools for regression, classification and ML.
- Will use **kNN()** and **fineTuning()**.

The fineTuning() Function

The fineTuning() Function

- Advanced grid search tool for tuning parameter selection.

The fineTuning() Function

- Advanced grid search tool for tuning parameter selection.
- Motivation: The reported "best" parameter combination may not really be best.

The fineTuning() Function

- Advanced grid search tool for tuning parameter selection.
- Motivation: The reported “best” parameter combination may not really be best. Avoid p-hacking problem.

The fineTuning() Function

- Advanced grid search tool for tuning parameter selection.
- Motivation: The reported "best" parameter combination may not really be best. Avoid p-hacking problem.
- The tool allows exploring various good parameter combinations.

The fineTuning() Function

- Advanced grid search tool for tuning parameter selection.
- Motivation: The reported “best” parameter combination may not really be best. Avoid p-hacking problem.
- The tool allows exploring various good parameter combinations. Bonferroni CIs.

The fineTuning() Function

- Advanced grid search tool for tuning parameter selection.
- Motivation: The reported "best" parameter combination may not really be best. Avoid p-hacking problem.
- The tool allows exploring various good parameter combinations. Bonferroni CIs.
- Includes a plotting facility.

Example: Major League Baseball Data

Example: Major League Baseball Data

- For convenience, a very simple example: Predict weight from height, age.
- Dataset from **regtools** package.
- $n = 1023, \; p = 2$ (plus others not used here)

MLB, cont'd.

MLB, cont'd.

```
> data(mlb)  # in regtools pkg
> mlb ← mlb[,c(4,6,5)]
> mlb[1,]
  Height   Age Weight
1     74 22.99    180
> args(kNN)
function (x,y,newx=x,kmax,scaleX=TRUE,
    PCAcomps=0,expandVars=NULL,expandVals=NULL,
    smoothingFtn=mean,allK=FALSE,leave1out=FALSE,
    classif=FALSE)
```

MLB, cont'd

MLB, cont'd

The **fineTuning()** function calls a user-defined function that does the work:

```
# fineTuning() forms current training test sets,
# dtrn and dtst, and current parameter combination
# ‘Mcmbi
knnCall ← function(dtrn,dtst,cmbi) {
   knnOut ← kNN(dtrn[,1:2],dtrn[,3],dtst[,1:2],
      cmbi$k,expandVars=1,expandVals=cmbi$expandHt)
   mean(abs(dtst[,3] − knnOut$regests))
}
```

And the call:

```
ft ← fineTuning(mlb,pars=list(k=c(5,20,50,100),
   expandHt=c(1.8,1.5,1.2,1,0.8,0.5,0.2)),
   regCall=knnCall,nTst=500,nXval=100)
```

MLB Output

MLB Output

```
> ft
$outdf
     k expandHt  meanAcc      seAcc   bonfAcc
1   50      1.8 13.81726 0.03721619 0.11625351
2   20      1.8 13.84013 0.03122950 0.09755266
3  100      1.8 13.87238 0.03471346 0.10843563
4   20      0.8 13.87528 0.03619783 0.11307242
5  100      1.2 13.89429 0.03805532 0.11887472
...
...
24   5      1.2 14.84733 0.03666898 0.11454417
25   5      1.5 14.89271 0.03242414 0.10128441
26   5      0.2 14.89479 0.03801763 0.11875700
27   5      0.5 14.90646 0.04020769 0.12559816
28 100      0.2 15.14842 0.03691466 0.11531160
```

MLB Comments

MLB Comments

- As expected, the largest expansion value for Height seems best; Height is more important than Age.

MLB Comments

- As expected, the largest expansion value for Height seems best; Height is more important than Age.
- Further investigation with even larger expansion seems warranted.

MLB Comments

- As expected, the largest expansion value for Height seems best; Height is more important than Age.
- Further investigation with even larger expansion seems warranted.
- But beware of p-hacking!

MLB Comments

- As expected, the largest expansion value for Height seems best; Height is more important than Age.
- Further investigation with even larger expansion seems warranted.
- But beware of p-hacking!
  - All results subject to sample variation.
  - Thus **fineTuning()** displays radii of Bonferroni CIs.
  - An earlier run with **nXval** (cross val. folds) at 25 had ambiguous results; 100 works well here.

MLB Plot

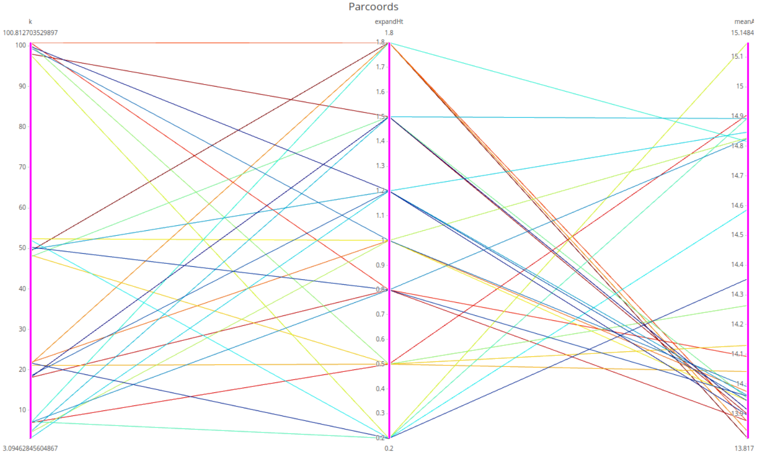
MLB Plot

- The **fineTuning()** function has an associated generic plot function.
- Use the *parallel coordinates* graphical method (Inselberg, 1997).
- View multidimensional data in 2-D.
- Implemented in **cdparcoord** ("categorical and discrete parallel coordinates") package.
- Latter uses Plotly, so can drag columns to change order etc.

Plot

Plot

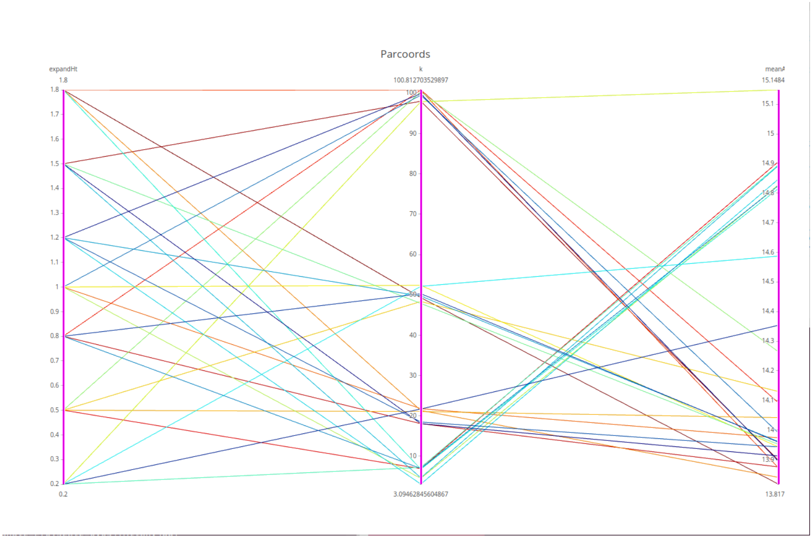
```
> plot(ft)
```


Parcoords
k
100.812703529897
100
90
80
70
60
50
40
30
20
10
3.09462845604867
expandHt
1.8
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.2
meanA
15.1484
15.1
15
14.9
14.8
14.7
14.6
14.5
14.4
14.3
14.2
14.1
14
13.9
13.817

Plot, Column Dragged

Plot, Column Dragged

Can rotate columns by dragging.

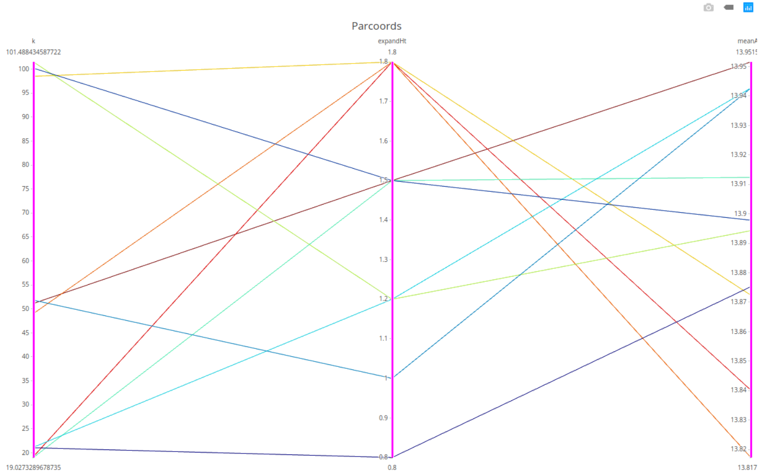


Plot, Zoomed in

Plot, Zoomed in

Can zoom in, isolating only the best combinations.

```
> plot(ft, -10)
```



Example: Prog/Engr Census Data

Example: Prog/Engr Census Data

- Dataset from **regtools** package.
- Predict occupation, among 6 programmer/engineer job titles. X = age, MS indicator, PhD indicator, gender (M), wage income, weeks worked.
- $n = 20070, \ p = 6$

Census cont'd.

Census cont'd.

```
knnCall ← function(dtrn,dtst,cmbi) {
   dtrn ← as.matrix(dtrn)
   dtst ← as.matrix(dtst)
   knnOut ← kNN(
      dtrn[,-(4:9)],dtrn[,4:9],dtst[,-(4:9)],
      cmbi$k,
      expandVars=c(1:6),
      expandVals=c(cmbi$age,cmbi$e14,cmbi$e16,
         cmbi$gend,cmbi$wks,cmbi$wage),
      classif=TRUE)
   preds ← apply(knnOut$regests,1,which.max)
   newy ← apply(dtst[,4:9],1,which.max)
   mean(preds == newy)
}
```

Census cont'd.

Census cont'd.

```
ft ← fineTuning(ped,
   pars=list(k=c(10,50),age=c(0.5,2),
   e14=c(0.5,2),e16=c(0.5,2),gend=c(0.5,2),
   wks=c(0.5,2),wage=c(0.5,2)),
   regCall=knnCall,nTst=500,nXval=100)
```

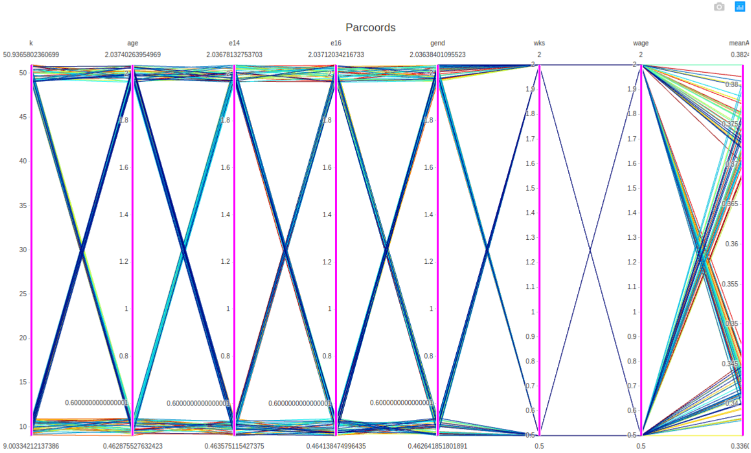
Census cont;d,

Census cont;d,

```
> ft$outdf
     k  age  e14  e16  gend  wks  wage  meanAcc          seAcc
bonfAcc
1    10  0.5  2.0  0.5   2.0  0.5   0.5  0.33602  0.002248141
2    10  0.5  0.5  0.5   0.5  2.0   0.5  0.33792  0.002365906
3    10  0.5  2.0  2.0   2.0  0.5   0.5  0.33810  0.002216809
4    10  2.0  0.5  2.0   0.5  0.5   0.5  0.33812  0.002026455
5    10  0.5  2.0  2.0   0.5  2.0   0.5  0.33820  0.002267647
...
...
124  50  0.5  2.0  0.5   0.5  0.5   2.0  0.37990  0.002038493
125  50  2.0  0.5  2.0   2.0  0.5   0.5  0.38038  0.002260365
126  50  2.0  0.5  0.5   2.0  0.5   2.0  0.38042  0.002094205
127  50  0.5  0.5  0.5   0.5  0.5   2.0  0.38100  0.002340767
128  50  0.5  0.5  2.0   2.0  0.5   2.0  0.38248  0.002202867
```

Census cont'd.

Census cont'd.



Further Comments

Further Comments

- Can be done for any value of p.
- Larger p means: (a) More potential for p-hacking. (b) More columns in plot.
- Optimization not easy in k-NN case, due to lack of derivatives, though could be done for kernel-based smoothing.

Locally-Adaptive Choice of k

Locally-Adaptive Choice of k

- Classic relation:

$$MSE = \text{variance} \ + \ \text{bias}^2 \tag{1}$$

- If $E(Y \mid X = t)$ has a large gradient at a point t, bias may be large, especially on fringes of X.
- It thus may be worth sacrificing on variance, i.e. worth using a smaller k.
- Thus locally-adaptive choice of k.

Locally-Adaptive, cont'd.

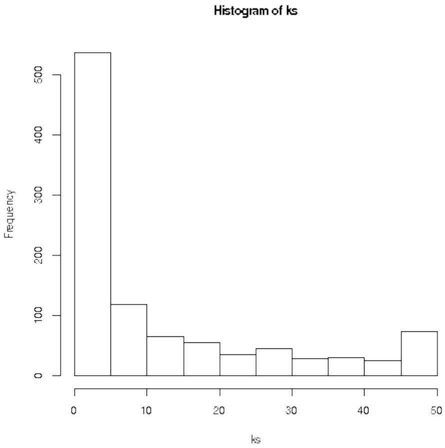
Locally-Adaptive, cont'd.

- There have been a number of theoretical treatments, but they do not appear in common software packages.
- The **regtools** package has the function **bestKperPoint()**
- At each X_i, asks, "Which k would have best predicted Y_i?"

```
> args(regtools:::bestKperPoint)
function (kNNout, y)
```

where **kNNout** is an object returned by **kNN()** and y is the original Y vector.

```
> knnOut ← kNN(mlb[,1:2],mlb[,3],mlb[,1:2],50,
     expandVars=1,expandVals=1.8)
> ks ← bestKperPoint(knnOut,mlb[,3])
```



Just started on this, plan to develop into a diagnostic tool.

Future Work

Future Work

- Comparisons of “improved” k-NN and other ML methods, in accuracy and comp time.
- Development of locally-adaptive approach.