Knn classification Notes

Code

Remember our mpg dataset

r r require(ggplot2)

```
Warning message:
In scan(file = file, what = what, sep = sep, quote = quote, dec = dec,  :
  EOF within quoted string
```

r r head(mpg)

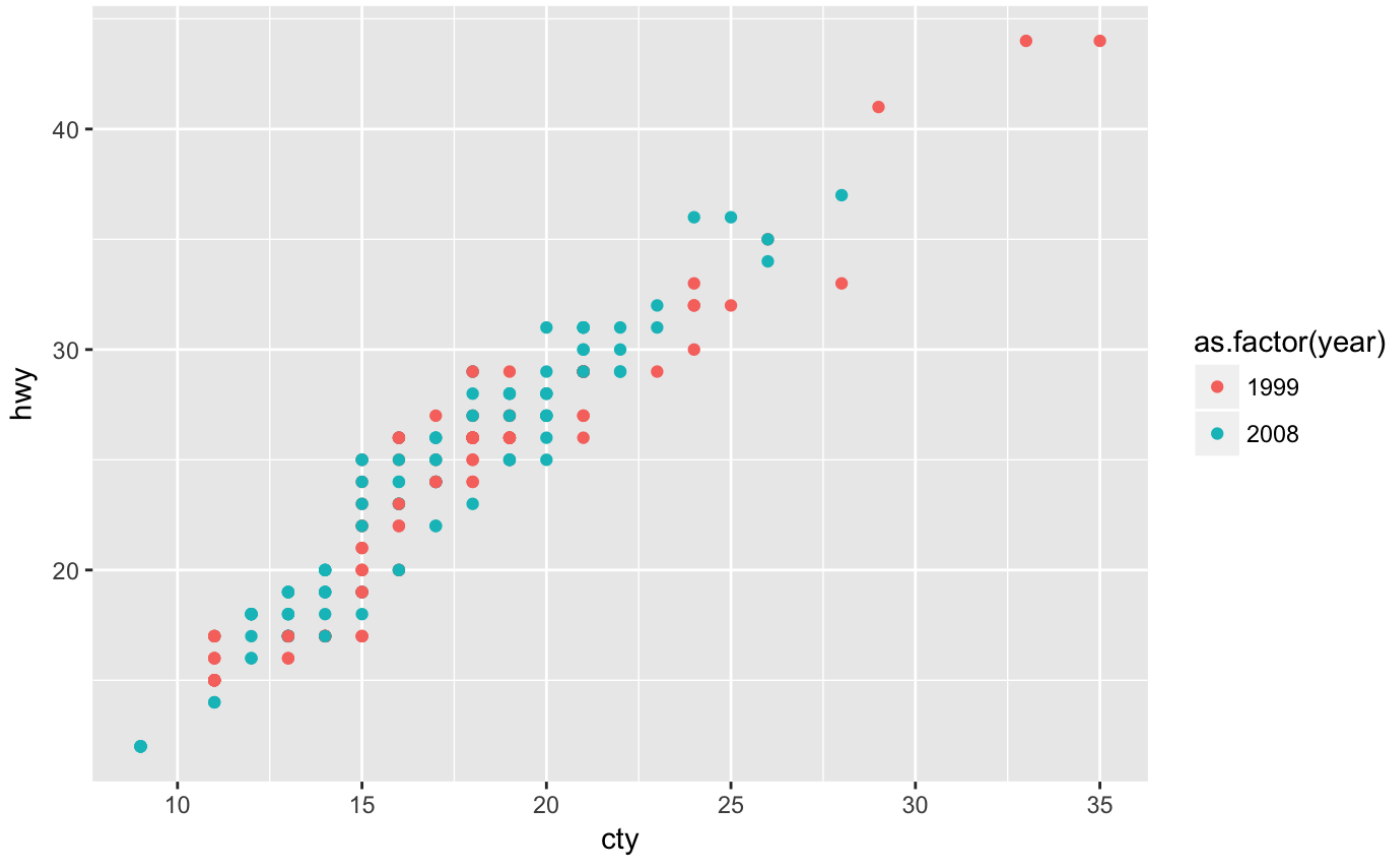
Given the city and highway mileages of a car in our dataset, can we guess which year (1999 or 2008) they were manufactured? Another way of asking this question is can we predict the year variable (which happens to be categorical) in the dataset from the hwy and cty variables.

Question 1: Based on last weeks lecture, can you define a regression model? What type of regression model would it be?

Question 2: Can you `interperet` the results ?

Question 3: Does the graph below back up your interpretation ?

r r ggplot(data=mpg)+geom_point(aes(x=cty, y=hwy, color=as.factor(year)))



Classification with KNN

Next, we will conduct the same classification process with the knn method. The process involves:

- Selecting the variable from the dataset
- Normalize the variables. Note: in this example our input variables happen to be all numerical so we will appply the nomralization process to all the input variables. If any of your input variables are categorical variables, they need to be converted into dummy variables. See the `categorical input variables` section for details.
- Splitting the data into test and training sets.
- Train and prerict.
- Test the accuracy of the prediction.

Select variables

We will use hwy and cty variables as the input variable to `predict` the year variable. The predicted variable is also known as label. Since we are using the year as categorical variable, the numerical values of years need to be treated as factor.

`r` r input= subset(mpg, select=c(hwy, cty)) label=as.factor(mpg$year)

Normalize input variables

KNN requires that each numerical values are normalized, so similarity calculations are not effected by the scale of the variabele.

r r normalization<-function(x){ (x-min(x))/(max(x)-min(x)) } n.input<-sapply(input, normalization)

Splitting the data

As discussed in the last class, we need to split the data `randomly` into two groups: training and testing.

r r set.seed(1234) ## In order to get the same split everytime we run the random split indicies=sample(1:2 , length(mpg$cty) , replace = T, prob=c(.8,.2)) ## creating a vector of 1s and 2s #dim(n.input[indicies==1,]);dim(n.input) training.input=n.input[indicies==1,] ## the rows that match to the 1s in the indices are selected as training testing.input=n.input[indicies==2,] ## the rows that match to the 2s in the indices are selected as training

labels (the year categories) are also split

training.label=label[indicies==1]
testing.label=label[indicies==2]

Train and Predict

You can run prediction by defining training input, testing input and training labels all at once. The prediction (or the outputs) are what knn things the labels of the testing data should be.

r r #install.packages('class') require(class)

```
Loading required package: class
```

r r set.seed(1234) predictions<-knn(train=training.input, test=testing.input, cl=training.label, k=10)

If the model is perfectly accurate then the `predictions` should be the same as the `testing label` . See below if this is the case

r r data.frame(predictions,testing.label)

Calcuating Accuracy

Accurarcy is considered as the percentage of correctly labeled data in the testing set

r r accuracy=sum(predictions==testing.label)/length(predictions) accuracy

```
[1] 0.5909091
```

r r table(predictions, testing.label)

```
          testing.label
predictions 1999 2008
       1999   14    8
       2008   10   12
```

Confussion Matrix:

Another way to study the accuracy is to look at the confusiong matrix. Which can be created as below.

r r table(predictions, testing.label)

```
          testing.label
predictions 1999 2008
       1999   14    8
       2008   10   12
```

In this matrix the columns represent the true values and the rows represent predicted values. For example, of the 24 `1999` vehicles (the sum of the first column is 24), 14 of them were correctly predicted as 1999 (First row of the first column is 14); 10 of them were incorrectly predicted as 2008…
The values on the diagonal are the number of true predictions. Values outside of the diagonals represent the number of false predictions for the corresponding category.

There is also a package for creating confusing matrix with more detailed results

r r results<-data.frame(predictions, testing.label) # install.packages(‘caret’) require(caret)

```
Loading required package: caret
there is no package called ‘caret’
```

r r # install.packages(‘e1071’) require(e1071)

```
Loading required package: e1071
there is no package called ‘e1071’
```

r r confusionMatrix(table(results))

```
Error in confusionMatrix(table(results)) :
  could not find function \confusionMatrix\
```

Categorical Input variables

Categorical variables need to be represented with numbers for similarity calculations. The best way to do this is creating dummy variables for different category values (e.x. hardback, paperback example).

Take the class variable in our dataset:

```
 [1] \compact\ \compact\ \compact\ \compact\ \compact\ \compact\ \compact\ \compact\ \compact\
\compact\ \compact\
[12] \compact\ \compact\ \compact\ \compact\ \midsize\ \midsize\ \midsize\ \suv\     \suv\
```

r r table(mpg$class)

```

   2seater    compact    midsize    minivan     pickup subcompact        suv
         5         47         41         11         33         35         62
```

We can create a dummy variable for each value. The easiest way to do this is using the model.matrix function example is below.

r r class_dummies=model.matrix(~class - 1, data=mpg) head(class_dummies)

```
  class2seater classcompact classmidsize classminivan classpickup classsubcompact classsuv
1            0            1            0            0           0                0        0
2            0            1            0            0           0                0        0
3            0            1            0            0           0                0        0
4            0            1            0            0           0                0        0
5            0            1            0            0           0                0        0
6            0            1            0            0           0                0        0
```

We can then combine input with the dummy variables

r r input= subset(mpg, select=c(hwy, cty)) input = data.frame(input, class_dummies)

Question 4: Try to run another knn classification for the year with the hwy,cty and class variables. Are the results better or worse than with the hwy and cty only ?