Beijing PM2.5 Data

Hourly data of PM2.5 from the US Embassy in Beijing. PM2.5 is a measure of particulate matter that have a diameter of less than 2.5 micrometers. They are an important measure of air quality for humans.

The data was downloaded from the UCI Machine Learning Repository and is calleed the Beijing PM2.5 Data Set

Load data

Loading the data and restricting to complete cases leaves about 41K observations. We will remove the No, day, and hour columns. Remaining columns are the year and month, pm2.5, temperature, pressure, combined wind direction. cumulated wind speed, cumulated hours of snow and cumulated hours of rain.

```
df <- read.csv("PRSA_data.csv", header=TRUE)
df <- df[complete.cases(df), c(3, 6:13)]
head(df)
```

```
##    month pm2.5 DEWP TEMP PRES cbwd  Iws Is Ir
## 25     1   129  -16   -4 1020   SE 1.79  0  0
## 26     1   148  -15   -4 1020   SE 2.68  0  0
## 27     1   159  -11   -5 1021   SE 3.57  0  0
## 28     1   181   -7   -5 1022   SE 5.36  1  0
## 29     1   138   -7   -5 1022   SE 6.25  2  0
## 30     1   109   -7   -6 1022   SE 7.14  3  0
```

```
str(df)
```

```
## 'data.frame':    41757 obs. of  9 variables:
##  $ month: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ pm2.5: int  129 148 159 181 138 109 105 124 120 132 ...
##  $ DEWP : int  -16 -15 -11 -7 -7 -7 -7 -7 -8 -7 ...
##  $ TEMP : num  -4 -4 -5 -5 -5 -6 -6 -5 -6 -5 ...
##  $ PRES : num  1020 1020 1021 1022 1022 ...
##  $ cbwd : Factor w/ 4 levels "cv","NE","NW",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ Iws  : num  1.79 2.68 3.57 5.36 6.25 ...
##  $ Is   : int  0 0 0 1 2 3 4 0 0 0 ...
##  $ Ir   : int  0 0 0 0 0 0 0 0 0 0 ...
```

Train/test split

```
set.seed(1234)
i <- sample(1:nrow(df), 0.8*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Try linear regression

Correlation is about 0.5, not bad but not impressive either. The mse is 6232. The rmse is about 79. Since the range of pm2.5 in the test data is 0:886 this is about 10%. Not really that bad.

```r
# build a model
lm1 <- lm(pm2.5~., data=train)
summary(lm1)
```

```
##
## Call:
## lm(formula = pm2.5 ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -183.20  -51.37  -15.65   30.75  881.23
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.715e+03  8.192e+01  20.931  < 2e-16 ***
## month       -9.156e-01  1.340e-01  -6.831 8.58e-12 ***
## DEWP         4.144e+00  6.141e-02  67.487  < 2e-16 ***
## TEMP        -6.216e+00  7.626e-02 -81.505  < 2e-16 ***
## PRES        -1.498e+00  8.025e-02 -18.663  < 2e-16 ***
## cbwdNE      -2.679e+01  1.590e+00 -16.856  < 2e-16 ***
## cbwdNW      -2.959e+01  1.310e+00 -22.581  < 2e-16 ***
## cbwdSE       2.524e+00  1.227e+00   2.057   0.0397 *
## Iws         -1.948e-01  9.717e-03 -20.045  < 2e-16 ***
## Is          -3.524e+00  5.663e-01  -6.224 4.90e-10 ***
## Ir          -6.154e+00  3.122e-01 -19.714  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 79.14 on 33394 degrees of freedom
## Multiple R-squared:  0.2613, Adjusted R-squared:  0.2611
## F-statistic:  1181 on 10 and 33394 DF,  p-value: < 2.2e-16
```

```r
# evaluate
pred <- predict(lm1, newdata=test)
cor_lm <- cor(pred, test$pm2.5)
mse_lm <- mean((pred-test$pm2.5)^2)
```

normalize data

```r
train$cbwd <- as.integer(train$cbwd)
test$cbwd <- as.integer(test$cbwd)
normalize <- function(x){
  return ((x - min(x)) / (max(x) - min(x)))
}
train_scaled <- as.data.frame(lapply(train, normalize))
test_scaled <- as.data.frame(lapply(test, normalize))
```

Try knn

Correlation is up to 0.7. MSE is down to 4302. Much better, even without scaling and just randomly picking a k.

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.3
## Loading required package: lattice
## Loading required package: ggplot2
```

```r
fit <- knnreg(train_scaled[,-2], train_scaled[,2], k=7)
# evaluate
pred <- predict(fit, test_scaled[,-2])
pred <- pred * (max(test$pm2.5) - min(test$pm2.5)) + min(test$pm2.5)
cor_knn <- cor(pred, test$pm2.5)
mse_knn <- mean((pred-test$pm2.5)^2)
```

Try neural network

Ran in to a series of problems: - turned on verbose output with lifesign="full" and realized threshold too high - raised threshold to 0.04, only 10K train; performance between linear and knn - the following is a run on a smaller training set to test the architecture

```r
library(neuralnet)
n <- names(train_scaled)
f <- as.formula(paste("pm2.5 ~", paste(n[!n %in% "pm2.5"], collapse = " + ")))
set.seed(1234)
j <- sample(1:nrow(train), 10000, replace=FALSE)
train_small <- train_scaled[j,]
set.seed(1234)  # this should not be necessary but it is
nn1 <- neuralnet(f, data=train_small, hidden=c(6,3), threshold = 0.05, linear.output=TRUE)
plot(nn1)
# evaluate
pred <- compute(nn1, test_scaled[,-2])
pred <- pred$net.result
pred_unscale <- pred * (max(test$pm2.5) - min(test$pm2.5)) + min(test$pm2.5)
cor_nn1 <- cor(pred_unscale, test$pm2.5)
mse_nn1 <- mean((pred_unscale - test$pm2.5)^2)
nn1$result.matrix
```

```
##                                       1
## error                    22.12521198080
## reached.threshold         0.04535879807
## steps                  7764.00000000000
## Intercept.to.1layhid1    -3.13790041841
## month.to.1layhid1         2.66791062317
## DEWP.to.1layhid1          2.32867347928
## TEMP.to.1layhid1         -2.37029689658
## PRES.to.1layhid1         -1.21156276149
## cbwd.to.1layhid1          0.86673689445
## Iws.to.1layhid1           6.01792109941
## Is.to.1layhid1            8.92529736677
## Ir.to.1layhid1           -3.62685527044
## Intercept.to.1layhid2    -0.60291125447
## month.to.1layhid2        -0.26404266925
## DEWP.to.1layhid2         -5.73670726993
## TEMP.to.1layhid2          2.38279627270
```

```
## PRES.to.1layhid2         1.84555893043
## cbwd.to.1layhid2         0.43864559965
## Iws.to.1layhid2         10.88925035169
## Is.to.1layhid2           0.08465862336
## Ir.to.1layhid2           5.35630993630
## Intercept.to.1layhid3   -1.07084586652
## month.to.1layhid3        1.07375906930
## DEWP.to.1layhid3        -0.72137282294
## TEMP.to.1layhid3        -0.44215861181
## PRES.to.1layhid3        -0.57182953688
## cbwd.to.1layhid3         1.65764249369
## Iws.to.1layhid3         10.80619918850
## Is.to.1layhid3          -1.76432789976
## Ir.to.1layhid3           1.32186035813
## Intercept.to.1layhid4   -0.51233406939
## month.to.1layhid4        0.51546933133
## DEWP.to.1layhid4        -1.28760775887
## TEMP.to.1layhid4         0.37628141321
## PRES.to.1layhid4        -0.48885895271
## cbwd.to.1layhid4        -0.39042320179
## Iws.to.1layhid4         -3.23439604553
## Is.to.1layhid4          -2.54720501398
## Ir.to.1layhid4           1.83979728170
## Intercept.to.1layhid5    0.30090059464
## month.to.1layhid5       -9.48242621756
## DEWP.to.1layhid5         1.90849736702
## TEMP.to.1layhid5         1.65067697521
## PRES.to.1layhid5         0.82881924863
## cbwd.to.1layhid5        -0.72556930648
## Iws.to.1layhid5        -10.89942157268
## Is.to.1layhid5          -8.00218754165
## Ir.to.1layhid5           0.30569890335
## Intercept.to.1layhid6   -3.43417734495
## month.to.1layhid6        4.26210584325
## DEWP.to.1layhid6         0.56338090930
## TEMP.to.1layhid6        -0.02237076607
## PRES.to.1layhid6        -0.53148028787
## cbwd.to.1layhid6        -1.21000213372
## Iws.to.1layhid6        -12.93053074206
## Is.to.1layhid6          -5.47643194596
## Ir.to.1layhid6           1.27668821268
## Intercept.to.2layhid1   -0.47622934851
## 1layhid.1.to.2layhid1   -0.52207175419
## 1layhid.2.to.2layhid1    1.02573803133
## 1layhid.3.to.2layhid1   -1.08070466569
## 1layhid.4.to.2layhid1   -2.68639669472
## 1layhid.5.to.2layhid1    0.06982320323
## 1layhid.6.to.2layhid1    0.14287779866
## Intercept.to.2layhid2    2.47751850440
## 1layhid.1.to.2layhid2   -6.26479658798
## 1layhid.2.to.2layhid2    6.46489304949
## 1layhid.3.to.2layhid2   -0.52843383490
## 1layhid.4.to.2layhid2   12.17933479365
## 1layhid.5.to.2layhid2   -6.38412709166
```

```
## 1layhid.6.to.2layhid2   -7.01559330210
## Intercept.to.2layhid3    1.65256267340
## 1layhid.1.to.2layhid3   -2.24430046798
## 1layhid.2.to.2layhid3    1.54529318095
## 1layhid.3.to.2layhid3    0.91262711135
## 1layhid.4.to.2layhid3   -7.70801743705
## 1layhid.5.to.2layhid3    3.72619371560
## 1layhid.6.to.2layhid3    3.10313158958
## Intercept.to.pm2.5       0.16366245000
## 2layhid.1.to.pm2.5      -1.01781841937
## 2layhid.2.to.pm2.5      -0.40655943537
## 2layhid.3.to.pm2.5       0.59437833138
```

```r
# look at weights
par(mfrow=c(2,2))
gwplot(nn1, selected.covariate = "month", min=-2.5, max=5)
gwplot(nn1, selected.covariate = "DEWP", min=-2.5, max=5)
gwplot(nn1, selected.covariate = "TEMP", min=-2.5, max=5)
gwplot(nn1, selected.covariate = "PRES", min=-2.5, max=5)
```

For month, the weights are negative for the first few months and positive for the latter ones. DEWP weights are positive while TEMP and PRES weights are negative. If you see a predictor for which the weights are consistently 0, that tells you that the predictor is not useful.

Try on full data

```r
set.seed(1234)  # this should not be necessary but it is
nn2 <- neuralnet(f, data=train_scaled, hidden=c(6,3), threshold = 0.05,  linear.output=TRUE)
# evaluate
pred <- compute(nn2, test_scaled[,-2])
pred <- pred$net.result
pred_unscale <- pred * (max(test$pm2.5) - min(test$pm2.5)) + min(test$pm2.5)
cor_nn2 <- cor(pred_unscale, test$pm2.5)
mse_nn2 <- mean((pred_unscale - test$pm2.5)^2)
nn2$result.matrix
```

```
##                                      1
## error                   70.76442641065
## reached.threshold        0.03856239618
## steps                97322.00000000000
## Intercept.to.1layhid1   -5.05842031561
## month.to.1layhid1        3.13081347182
## DEWP.to.1layhid1         2.96014961079
## TEMP.to.1layhid1        -1.05014984989
## PRES.to.1layhid1        -1.11780302323
## cbwd.to.1layhid1         0.59630481330
## Iws.to.1layhid1          4.07172456222
## Is.to.1layhid1           2.64806453192
## Ir.to.1layhid1          -2.12638136640
## Intercept.to.1layhid2    0.40962163774
## month.to.1layhid2      -13.10035573716
## DEWP.to.1layhid2       -11.07967382025
## TEMP.to.1layhid2         6.96154551676
## PRES.to.1layhid2         3.84997704381
```

```
## cbwd.to.1layhid2         -0.41480439601
## Iws.to.1layhid2          14.34312238613
## Is.to.1layhid2           -2.07433102844
## Ir.to.1layhid2            9.72398464475
## Intercept.to.1layhid3    -0.75449180263
## month.to.1layhid3         2.50070044277
## DEWP.to.1layhid3         -3.07389970103
## TEMP.to.1layhid3          1.90094750432
## PRES.to.1layhid3          2.60821440764
## cbwd.to.1layhid3          0.62675319813
## Iws.to.1layhid3           9.35642147060
## Is.to.1layhid3            1.47804674231
## Ir.to.1layhid3            5.28269276048
## Intercept.to.1layhid4    -2.08860418066
## month.to.1layhid4         1.77580894899
## DEWP.to.1layhid4         -0.96487701120
## TEMP.to.1layhid4          0.92808466331
## PRES.to.1layhid4         -0.66888373828
## cbwd.to.1layhid4         -0.41885821275
## Iws.to.1layhid4           2.02771982323
## Is.to.1layhid4            1.00729664350
## Ir.to.1layhid4            0.34167356761
## Intercept.to.1layhid5     0.25851060245
## month.to.1layhid5        -9.52227054549
## DEWP.to.1layhid5          2.04428490655
## TEMP.to.1layhid5          1.52465090663
## PRES.to.1layhid5         -0.12417714522
## cbwd.to.1layhid5         -0.41455913120
## Iws.to.1layhid5          -4.67621624707
## Is.to.1layhid5           -3.18435396634
## Ir.to.1layhid5           -0.95455261802
## Intercept.to.1layhid6    -3.90597397699
## month.to.1layhid6         4.26302080569
## DEWP.to.1layhid6          1.42032291079
## TEMP.to.1layhid6         -0.55045482864
## PRES.to.1layhid6         -1.11682315888
## cbwd.to.1layhid6         -1.15427146256
## Iws.to.1layhid6         -12.70401273853
## Is.to.1layhid6           -3.07661268675
## Ir.to.1layhid6           -1.99704329142
## Intercept.to.2layhid1    -0.26338118457
## 1layhid.1.to.2layhid1    -0.52251870863
## 1layhid.2.to.2layhid1     0.11072058682
## 1layhid.3.to.2layhid1    -0.44405652402
## 1layhid.4.to.2layhid1     0.17718323619
## 1layhid.5.to.2layhid1    -0.07277716709
## 1layhid.6.to.2layhid1     0.13396056306
## Intercept.to.2layhid2     2.46601677519
## 1layhid.1.to.2layhid2   -12.25387843665
## 1layhid.2.to.2layhid2     3.44921198268
## 1layhid.3.to.2layhid2     0.81972853834
## 1layhid.4.to.2layhid2    14.72161957954
## 1layhid.5.to.2layhid2    -6.50853613540
## 1layhid.6.to.2layhid2    -9.21490149260
```

```
## Intercept.to.2layhid3    -5.18190469715
## 1layhid.1.to.2layhid3    -2.46675320735
## 1layhid.2.to.2layhid3     1.21952235738
## 1layhid.3.to.2layhid3     7.96085027314
## 1layhid.4.to.2layhid3     0.74987749332
## 1layhid.5.to.2layhid3    27.63316127978
## 1layhid.6.to.2layhid3     4.69678355056
## Intercept.to.pm2.5        0.16305400762
## 2layhid.1.to.pm2.5       -1.06802722979
## 2layhid.2.to.pm2.5       -0.40689107446
## 2layhid.3.to.pm2.5        0.63976225761
```

Results so far

```r
print(paste("cor and mse for linear regression: ", cor_lm, mse_lm))
```

```
## [1] "cor and mse for linear regression:  0.513420623615538 6232.51207615418"
```

```r
print(paste("cor and mse for nn on 10K training data: ", cor_nn1, mse_nn1))
```

```
## [1] "cor and mse for nn on 10K training data:  0.683424467601116 4897.58871907957"
```

```r
print(paste("cor and mse for nn on 33K training data: ", cor_nn2, mse_nn2))
```

```
## [1] "cor and mse for nn on 33K training data:  0.699485001864613 4738.66481420322"
```

Takeaways: - comparing linear, knn, and nn: knn is the clear favorite - the fact that linear did not do well and knn did may have been an indication that there is not a linear relationship between the predictors and the response, perhaps we need a more complex architecture

try a more complex model on all the training data

```r
set.seed(1234)  # this should not be necessary but it is
nn3 <- neuralnet(f, data=train_scaled, hidden=c(6,4,2), threshold = 0.12, linear.output=TRUE)
# evaluate
pred <- compute(nn3, test_scaled[,-2])
pred <- pred$net.result
pred_unscale <- pred * (max(test$pm2.5) - min(test$pm2.5)) + min(test$pm2.5)
cor_nn3 <- cor(pred_unscale, test$pm2.5)
mse_nn3 <- mean((pred_unscale - test$pm2.5)^2)
# print results
print(paste("cor and mse for nn with 3 hidden layers on 33K training data: ", cor_nn3, mse_nn3))
```

```
## [1] "cor and mse for nn with 3 hidden layers on 33K training data:  0.696470794870237 4792.1443672050
```

```r
nn3$result.matrix
```

```
##                                      1
## error                   71.18368994754
## reached.threshold        0.11346036781
## steps                 6892.00000000000
## Intercept.to.1layhid1   -1.55182025599
## month.to.1layhid1        0.51235739498
## DEWP.to.1layhid1         1.06819750548
## TEMP.to.1layhid1        -1.82118549681
```

```
## PRES.to.1layhid1         0.62247995457
## cbwd.to.1layhid1         0.89863847263
## Iws.to.1layhid1         11.28483926495
## Is.to.1layhid1           1.69427636071
## Ir.to.1layhid1          -0.89661050413
## Intercept.to.1layhid2   -8.13186616360
## month.to.1layhid2       16.08734451298
## DEWP.to.1layhid2         9.02249098626
## TEMP.to.1layhid2        -5.01208717997
## PRES.to.1layhid2         2.39532601361
## cbwd.to.1layhid2         0.39430756492
## Iws.to.1layhid2         10.71777134553
## Is.to.1layhid2           9.89412907983
## Ir.to.1layhid2          -5.91736002960
## Intercept.to.1layhid3   -0.34737189424
## month.to.1layhid3       14.56616713089
## DEWP.to.1layhid3        -5.11234053228
## TEMP.to.1layhid3        -3.58936680730
## PRES.to.1layhid3        -1.14368193475
## cbwd.to.1layhid3         0.07447489672
## Iws.to.1layhid3          8.30368060466
## Is.to.1layhid3           4.58277423027
## Ir.to.1layhid3          -0.94003709447
## Intercept.to.1layhid4   -4.43743852413
## month.to.1layhid4        5.72561550679
## DEWP.to.1layhid4        -0.11117223436
## TEMP.to.1layhid4         0.63847293795
## PRES.to.1layhid4        -1.40598104636
## cbwd.to.1layhid4        -0.48779909222
## Iws.to.1layhid4        -13.67473667885
## Is.to.1layhid4           2.17584520967
## Ir.to.1layhid4           0.47791175569
## Intercept.to.1layhid5   -2.76130915927
## month.to.1layhid5        1.50692690267
## DEWP.to.1layhid5        -3.43720263144
## TEMP.to.1layhid5         0.23896286279
## PRES.to.1layhid5         2.67318361449
## cbwd.to.1layhid5         0.25941969303
## Iws.to.1layhid5         14.59393754688
## Is.to.1layhid5           2.90684856688
## Ir.to.1layhid5           2.20090759053
## Intercept.to.1layhid6    2.17435117104
## month.to.1layhid6       -1.01359464670
## DEWP.to.1layhid6        -4.94788215094
## TEMP.to.1layhid6         2.22015357105
## PRES.to.1layhid6        -1.49519097810
## cbwd.to.1layhid6         0.07376456897
## Iws.to.1layhid6         -4.09190880784
## Is.to.1layhid6          -3.04814365248
## Ir.to.1layhid6           3.90701238041
## Intercept.to.2layhid1    0.51469367003
## 1layhid.1.to.2layhid1   -8.09764628240
## 1layhid.2.to.2layhid1    1.21808251540
## 1layhid.3.to.2layhid1    1.94577531651
```

```
## 1layhid.4.to.2layhid1  -12.70653452098
## 1layhid.5.to.2layhid1    4.60596037048
## 1layhid.6.to.2layhid1   22.97196572018
## Intercept.to.2layhid2    1.86934364633
## 1layhid.1.to.2layhid2    0.87612744561
## 1layhid.2.to.2layhid2   -1.86816089920
## 1layhid.3.to.2layhid2   -2.18627107853
## 1layhid.4.to.2layhid2    3.68643502544
## 1layhid.5.to.2layhid2  -10.01219577588
## 1layhid.6.to.2layhid2   -3.96095259019
## Intercept.to.2layhid3    1.13412644497
## 1layhid.1.to.2layhid3   -2.24946071219
## 1layhid.2.to.2layhid3   -0.10821621856
## 1layhid.3.to.2layhid3   -1.60541240614
## 1layhid.4.to.2layhid3   -0.33352562879
## 1layhid.5.to.2layhid3    2.96474255220
## 1layhid.6.to.2layhid3    0.56230814723
## Intercept.to.2layhid4    0.01234775590
## 1layhid.1.to.2layhid4   -2.73928992846
## 1layhid.2.to.2layhid4   -0.05629421124
## 1layhid.3.to.2layhid4    1.56558061044
## 1layhid.4.to.2layhid4   -2.41333511364
## 1layhid.5.to.2layhid4    0.53498856719
## 1layhid.6.to.2layhid4   -1.79670359099
## Intercept.to.3layhid1   -2.06124560512
## 2layhid.1.to.3layhid1   -1.11530412782
## 2layhid.2.to.3layhid1    9.06620026552
## 2layhid.3.to.3layhid1    0.01574876977
## 2layhid.4.to.3layhid1   -0.48737434960
## Intercept.to.3layhid2    0.27687613736
## 2layhid.1.to.3layhid2    0.65576551928
## 2layhid.2.to.3layhid2   -0.27542120945
## 2layhid.3.to.3layhid2    0.70294459421
## 2layhid.4.to.3layhid2    1.56904552811
## Intercept.to.pm2.5       1.01435537956
## 3layhid.1.to.pm2.5       0.32516764721
## 3layhid.2.to.pm2.5      -1.18722919376
```

other implementations

This took about an hour to run and did not do as well as the neuralnet() algorithm.

Commenting it out.

```
{r} library(caret) grid1 <- expand.grid(.decay=c(0.5, 0.1),
.size=c(4,5,6)) nnetfit <- train(pm2.5 ~ ., data=train_scaled,
method="nnet", maxit=1000, tuneGrid=grid1) # nnetfit$bestTune
size 6 decay 0.1 pred <- predict(nnetfit, newdata=test_scaled)
pred_unscale <- pred * (max(test$pm2.5) - min(test$pm2.5))
+ min(test$pm2.5) cor_nn4 <- cor(pred_unscale, test$pm2.5)
mse_nn4 <- mean((pred_unscale - test$pm2.5)^2) # cor 0.669
mse 5075 #
```