Notebook1

Code

CS4375 Intro to Machine Learning

Cory Pekkala

Leo Nguyen

About the Data

Overview

This data (https://archive.ics.uci.edu/ml/datasets/Power+consumption+of+Tetouan+city), gathered from UCI Machine Learning Repository, is concerned with the energy consumption in Tetouan city, a city in North Morroco. There are 52,416 observations in this dataset and 9 features. All of the features are numerical except for the DateTime column - which contains character arrays.

Attributes

1. Date Time: Each ten minutes.
2. Temperature: Weather Temperature of Tetouan city.
3. Humidity: Weather Humidity of Tetouan city.
4. Wind Speed of Tetouan city.
5. general diffuse flows
6. diffuse flows
7. power consumption of zone 1 of Tetouan city.
8. power consumption of zone 2 of Tetouan city.
9. power consumption of zone 3 of Tetouan city.

Source

Abdulwahed Salam, Abdelaaziz El Hibaoui Faculty of Sciences Tetouan, Morocco Abdelmalek Essaadi University Sallam.ye ‘@’ yahoo.com, hibaoui ‘@’ uae.ma

Libraries

Hide

```
library(corrplot)
library(caret)
library(ggplot2)
library(egg)
library(e1071)
library(Metrics)
```

Preprocessing

Initial Read

Hide

```
df <- read.csv("Tetuan City power consumption.csv")
head(df)
```

Hide

```
str(df)
```

There are 52,416 observations in this dataset and 9 features. All of the features are numerical except for the DateTime column. One peculiarity of the DateTime column is that all of the strings are unique.

Cleaning

For checking the NA's per column - utility function.

Hide

```
showNaN <- function(frame) {
 sapply(frame, function(x) sum(is.na(x)))
}
```

Every string in the DateTime columns is unique.

Hide

```
udts <- unique(df[,"DateTime"])
length(udts)==length(df[,"DateTime"])
```

Instead of maintaining the original structure of this data, we're going to remove the DateTime column and replace it with 3 columns - "Day", "Month", "Year". We are not going to consider time down to the seconds in the data. We will use these new encoded features in exploratory data analysis.

Hide

```
df[,"Day"] <- substr(df[,"DateTime"],1,1)
df[,"Month"] <- substr(df[,"DateTime"],3,3)
df[,"Year"] <- substr(df[,"DateTime"],5,8)
df <- subset(df,select=-c(DateTime))
head(df)
```

converting Day, Month, and Year into numerical vectors.

Hide

```
df[,"Day"] <- as.numeric(factor(df[,"Day"]))
df[,"Month"] <- as.numeric(factor(df[,"Month"]))
df[,"Year"] <- as.numeric(factor(df[,"Year"]))
head(df)
```

making sure there were no NA's indirectly produced as a result of dropping the DateTime column and adjusting the Day, Month, and Year columns to be numeric.

Hide

```
showNaN(df)
```

checking dtypes of the dataframe once more to see changes reflected in the columns.

Hide

```
str(df)
```

Exploratory Data Analysis

We’re going to begin by checking the correlation of the features in our dataset.Hopefully that will give us a better idea of what predictors could be helpful in our regression model. We will try to predict the temperature.

Hide

```
C <- cor(df)
corrplot(C,method="color",tl.srt=45)
```

Dark blue squares point at strong positive correlations among features; whereas deep red squares point at strong negative correlations among features.

Let’s look more at ‘Wind.Speed’, ‘general.diffuse.flows’, ‘Humidity’, ‘Zone.1.Power.Consumption’, ‘Zone.2..Power.Consumption’, ‘Zone.3..Power.Consumption’, and ‘Day’.

Hide

```
C[1,]
```

Let’s do some plots between these features against Temperature.

Hide

```
# function for a scatter plot comparing a predictor to the Temperature target, with optional reg
ression
temp.plt.scatter <- function(frame,col,point.color="black",reg.color="lightred",with.regression=
TRUE) {
  res <- ggplot(frame, aes(x=frame[,col],y=Temperature)) +
    geom_point(shape=12, color=point.color) +
    labs(title=paste("Temperature vs",col)) +
    xlab(col)

  if(with.regression) {
    res <- ggplot(frame, aes(x=frame[,col],y=Temperature)) +
            geom_point(shape=12, color=point.color) +
            geom_smooth(method=lm, se=FALSE, linetype="solid",color=reg.color) +
            labs(title=paste("Temperature vs",col)) +
            xlab(col)
  }
  res
}

# function to get the outlier indices from the box and whiskers plot between temperature and an
 associated column
temp.plt.outlier.indices <- function(frame,col) {
  outs <- boxplot.stats(frame[,col])$out
  which(frame[,col] %in% c(outs))
}
```

Hide

```
# Day is essentially categorical ranging from 1-9 (inclusive) ~ using barplot
g1 <- ggplot(data=df,aes(x=Temperature)) +
  geom_density(color="black",fill="#00FFFF")
g2 <- ggplot(data=df,aes(x=Wind.Speed)) +
  geom_density(color="black",fill="#6cf597")
g3 <- ggplot(data=df,aes(x=general.diffuse.flows)) +
  geom_density(color="black",fill="#97baf0")
g4 <- ggplot(data=df,aes(x=Zone.1.Power.Consumption)) +
  geom_density(color="black",fill="#d2aeeb")


ggarrange(g1,g2,g3,g4,
          labels=c("Temperature","Wind.Speed","general.diffuse.flows","Zone.1.Power.Consumption"
),
          ncol=2,nrow=2)
```

Hide

```
g5 <- ggplot(data=df,aes(x=Zone.2..Power.Consumption)) +
  geom_density(color="black",fill="#964a76")
g6 <- ggplot(data=df,aes(x=Zone.3..Power.Consumption)) +
  geom_density(color="black",fill="#dec87a")
g7 <- ggplot(data=df,aes(x=Day)) +
  geom_density(color="black",fill="#d6625a")
g8 <- ggplot(data=df,aes(x=Humidity)) +
  geom_density(color="black",fill="#410a66")

ggarrange(g5,g6,g7,g8,
          labels=c("Zone.2..Power.Consumption","Zone.3..Power.Consumption","Day","Humidity"),
          ncol=2,nrow=2)
```

Hide

```
g1 <- temp.plt.scatter(frame=df,
                col="Wind.Speed",
                point.color="blue",
                with.regression=FALSE)
g2 <- temp.plt.scatter(frame=df,
                col="general.diffuse.flows",
                point.color="black",
                with.regression=FALSE)
g3 <- temp.plt.scatter(frame=df,
                col="Zone.1.Power.Consumption",
                point.color="orange",
                with.regression=FALSE)
g4 <- temp.plt.scatter(frame=df,
                col="Zone.2..Power.Consumption",
                point.color="DarkGreen",
                with.regression=FALSE)
g5 <- temp.plt.scatter(frame=df,
                col="Zone.3..Power.Consumption",
                point.color="HotPink",
                with.regression=FALSE)
g6 <- temp.plt.scatter(frame=df,
                col="Humidity",
                point.color="#410a66",
                with.regression=FALSE)
```

Hide

```
ggarrange(g1,g2,g3,ncol=2,nrow=2)
```

Hide

```
ggarrange(g4,g5,g6,ncol=2,nrow=2)
```

Linear Kernel SVM Regression

First Attempt

Hide

```
set.seed(10)
# only doing so for 10,000 samples to reduce time complexity of svm
trimmed <- df[sample(1:nrow(df),10000,replace=FALSE),]
i <- sample(1:nrow(trimmed),8000,replace=FALSE)
train <- trimmed[i,]
test <- trimmed[-i,]
nrow(train)
nrow(test)
```

Hide

```
svm_linear <- svm(Temperature~., data=train, kernel="linear",scale=TRUE)
summary(svm_linear)
```

Hide

```
pred <- predict(svm_linear, newdata=subset(test, select = -c(Temperature)))
print(paste("RMSE=",rmse(pred,test$Temperature)))
print(paste("MAE=",mae(pred,test$Temperature)))
print(paste("R^2=",cor(test$Temperature,pred)^2))
```

The root mean squared error is relatively low for the linear kernel, using default cost, gamma, and epsilon hyperparameters. The value being around 5.2 signals at the average distance between what the model predicted and the data contained within the test set. Ultimately, it tells us that the model fits the data pretty well.

Hide

```
res_df <- data.frame(predictions=pred, actual=test$Temperature, day=factor(test$Day))
i <- sample(1:nrow(res_df),1000,replace=FALSE)
res_df <- res_df[i,]
ggplot(res_df, aes(x=actual, y=predictions, group=day)) +
  geom_point(shape=12, aes(color=day)) +
  theme_minimal() +
  theme(legend.position="top") +
  geom_abline() +
  labs(title="Actual Temperature vs Predicted Temperature (1000 observations)")
```

Hide

```
# function to observe predictions vs actual day by day
plt.day <- function(dayNum, color="black") {
  day_idx <- which(res_df$day==dayNum)
  adj_df <- res_df[day_idx,]
  ggplot(adj_df, aes(x=actual,y=predictions)) +
    geom_point(shape=12,color=color) +
    theme_minimal() +
    theme(legend.position="top")
}
```

Hide

```
d1 <- plt.day(1,color="red")
d2 <- plt.day(2,color="orange")
d3 <- plt.day(3,color="#7cae00")
d4 <- plt.day(4,color="green")
d5 <- plt.day(5,color="#00bfc4")
d6 <- plt.day(6,color="blue")
d7 <- plt.day(7,color="darkblue")
d8 <- plt.day(8,color="purple")
d9 <- plt.day(9,color="hotpink")
```

Hide

```
# plotting the predicted temperature vs the actual temperature for each day
ggarrange(d1,d2,d3,d4,d5,d6,d7,d8,d9,
          ncol=3,nrow=3,
          labels=c("Day 1", "Day 2", "Day 3", "Day 4","Day 5","Day 6","Day 7","Day 8","Day 9"),
          heights=c(50,50,50),widths=c(50,50,50))
```

Computing R^2, RMSE, MAE for each individual day

Hide

```
reg.day.report <- function(dayNum=1) {
  day_idx <- which(res_df$day==dayNum)
  adj_df <- res_df[day_idx,]
  c(dayNum,rmse(adj_df$predictions,adj_df$actual),
    mae(adj_df$predictions,adj_df$actual),
    cor(adj_df$actual,adj_df$predictions)^2)
}
```

Hide

```
ddf <- data.frame(day=1:9,rmse=1:9,mae=1:9,r2=1:9)
for(dayNum in 1:9) {
  res <- reg.day.report(dayNum)
  ddf[dayNum,'day'] <- res[1]
  ddf[dayNum,'rmse'] <- res[2]
  ddf[dayNum,'mae'] <- res[3]
  ddf[dayNum,'r2'] <- res[4]
}
```

Hide

```
# order by rmse,mae (ascending) and r2 (descending)
ddf[order(ddf$rmse, ddf$mae, -ddf$r2),]
```

Final Attempt

It appears Day 3,4,6,7,9 had the best R^2 values. Let's see if an additional model of linear kernel svm can be made, but this time, we will only include samples from 2,3,4 to see if we get a better overall RMSE, MAE, and R^2 score for our model. With the adjustment of removing the higher RMSE, MAE, and lower R^2 days, the overall RMSE and MAE of our svm with a linear kernel has decreased, while the R^2 has increased! Let's make one more adjustment, this time we will try to tune a linear svm using various values of cost and gamma, on the same data, to see if we can make it even better.

Hide

```
set.seed(13)
# only doing so for 10,000 samples to reduce time complexity of svm
trimmed <- df[sample(1:nrow(df),10000,replace=FALSE),]
i <- sample(seq(1,3), size=nrow(trimmed),replace=TRUE, prob=c(.8,.2,.2))
train <- trimmed[i==1,]
test <- trimmed[i==2,]
val <- trimmed[i==3,]
# finding indices containing day 3,4,6,7,9
train <- train[train$Day %in% c(3,4,6,7,9),]
test <- test[test$Day %in% c(3,4,6,7,9),]
val <- val[val$Day %in% c(3,4,6,7,9),]
nrow(train)
nrow(test)
nrow(val)
```

Hide

```
set.seed(14)
tune_svm_linear <- tune(svm, Temperature~., data=val, kernel="linear",
                        ranges=list(cost=c(0.001,0.01,0.1,1,5,10,100),
                                    gamma=c(0.5,1,2,3,4)))
tune_svm_linear$best.model
```

Hide

```
pred <- predict(tune_svm_linear$best.model, newdata=subset(test, select = -c(Temperature)))
print(paste("RMSE=",rmse(pred,test$Temperature)))
print(paste("MAE=",mae(pred,test$Temperature)))
print(paste("R^2=",cor(test$Temperature,pred)^2))
```

Hide

```
res_df <- data.frame(predictions=pred, actual=test$Temperature, day=factor(test$Day))
i <- sample(1:nrow(res_df),1000,replace=TRUE)
res_df <- res_df[i,]
ggplot(res_df, aes(x=actual, y=predictions, group=day)) +
  geom_point(shape=12, aes(color=day)) +
  theme_minimal() +
  theme(legend.position="top") +
  geom_abline() +
  labs(title="Actual Temperature vs Predicted Temperature (1000 observations)")
```

Linear Kernel Conclusion

We began by training a base model of a support vector machine with a linear kernel, including all predictors against Temperature. Through visualization we discovered that certain ‘days’ over time, fit better to the data than other days. We found the indices for each day, created a subset of our dataframe for those indices, and computed the Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-Squared (R^2) for each of the days. After plotting this into another dataframe visualization, we found which days produced minimal error terms and maximal R^2 values.

We then created a test, train, and validation dataset on a sample of the overall data - limited to 10,000 rows. This was done to save on complexity when we tuned our hyperparameters. In addition, we siphoned the minimal RMSE, MAE and maximal R^2 days out of each portion of data. Reducing the test and validation sets to under 1,000 observations and train to under 5,000 observations. From there, we tuned the cost and gamma hyperparameters for the support vector machine with a linear kernel with our validation set. As a result, R computed a ‘best model’ for us.

We took the best model produced from the tuning, fit it to the training data, made predictions, and then compared with test. Finallly, we computed the RMSE, MAE, and R^2 once more, and plotted the predictions our model made about temperature vs the actual temperature for those days, with a regression line to show the improvement in fit.

Overall, the first model gave us these numbers:

```
1. RMSE= 3.17623332952596
2. MAE = 2.42853155185427
3. R^2 = 0.723499180584428
```

Then, our final model gave us these numbers:

```
1. RMSE= 2.07338794945982
2. MAE= 1.62697464700517
3. R^2= 0.858783871648357
```

As we can see, every metric improved with tuning!

Radial Kernel SVM Regression

Now that there has been some exploratory data analysis and showing the correlation of the ‘days’ in our data to the target ‘temperature’, we will continue with the same techniques above, but now just for the other kernels for svm.

We will get right into tuning the model by first splitting on a subset of the data into train, test, and validate once more.

Hide

```
set.seed(23)
# only doing so for 10,000 samples to reduce time complexity of svm
trimmed <- df[sample(1:nrow(df),10000,replace=FALSE),]
i <- sample(seq(1,3), size=nrow(trimmed),replace=TRUE, prob=c(.8,.2,.2))
train <- trimmed[i==1,]
test <- trimmed[i==2,]
val <- trimmed[i==3,]
# finding indices containing day 3,4,6,7,9
train <- train[train$Day %in% c(3,4,6,7,9),]
test <- test[test$Day %in% c(3,4,6,7,9),]
val <- val[val$Day %in% c(3,4,6,7,9),]
nrow(train)
nrow(test)
nrow(val)
```

Hide

```
set.seed(24)
tune_svm_radial <- tune(svm, Temperature~., data=val, kernel="radial",
                        ranges=list(cost=c(0.001,0.01,0.1,1,5,10,100),
                                    gamma=c(0.5,1,2,3,4)))
tune_svm_radial$best.model
```

Now that we got the best model, we will test it

Hide

```
pred <- predict(tune_svm_radial$best.model, newdata=subset(test, select = -c(Temperature)))
print(paste("RMSE=",rmse(pred,test$Temperature)))
print(paste("MAE=",mae(pred,test$Temperature)))
print(paste("R^2=",cor(test$Temperature,pred)^2))
```

Hide

```
res_df <- data.frame(predictions=pred, actual=test$Temperature, day=factor(test$Day))
i <- sample(1:nrow(res_df),1000,replace=TRUE)
res_df <- res_df[i,]
ggplot(res_df, aes(x=actual, y=predictions, group=day)) +
  geom_point(shape=12, aes(color=day)) +
  theme_minimal() +
  theme(legend.position="top") +
  geom_abline() +
  labs(title="Actual Temperature vs Predicted Temperature (1000 observations)")
```

Polynomial Kernel SVM Regression

Now that there has been some exploratory data analysis and showing the correlation of the ‘days’ in our data to the target ‘temperature’, we will continue with the same techniques above, but now just for the other kernels for svm.

We will get right into tuning the model by first splitting on a subset of the data into train, test, and validate once more.

Hide

```
set.seed(33)
# only doing so for 10,000 samples to reduce time complexity of svm
trimmed <- df[sample(1:nrow(df),10000,replace=FALSE),]
i <- sample(seq(1,3), size=nrow(trimmed),replace=TRUE, prob=c(.8,.2,.2))
train <- trimmed[i==1,]
test <- trimmed[i==2,]
val <- trimmed[i==3,]
# finding indices containing day 3,4,6,7,9
train <- train[train$Day %in% c(3,4,6,7,9),]
test <- test[test$Day %in% c(3,4,6,7,9),]
val <- val[val$Day %in% c(3,4,6,7,9),]
nrow(train)
nrow(test)
nrow(val)
```

Hide

```
set.seed(34)
tune_svm_polynomial <- tune(svm, Temperature~., data=val, kernel="polynomial",
                         ranges=list(cost=c(0.001,0.01,0.1,1,5,10,100),
                                     gamma=c(0.5,1,2,3,4)))
tune_svm_polynomial$best.model
```

Now that we got the best model, we will test it

Hide

```
pred <- predict(tune_svm_polynomial$best.model, newdata=subset(test, select = -c(Temperature)))
print(paste("RMSE=",rmse(pred,test$Temperature)))
print(paste("MAE=",mae(pred,test$Temperature)))
print(paste("R^2=",cor(test$Temperature,pred)^2))
```

Hide

```
res_df <- data.frame(predictions=pred, actual=test$Temperature, day=factor(test$Day))
i <- sample(1:nrow(res_df),1000,replace=TRUE)
res_df <- res_df[i,]
ggplot(res_df, aes(x=actual, y=predictions, group=day)) +
  geom_point(shape=12, aes(color=day)) +
  theme_minimal() +
  theme(legend.position="top") +
  geom_abline() +
  labs(title="Actual Temperature vs Predicted Temperature (1000 observations)")
```