Class 2 Exercise - Regression and Regularization

Overview

Re; DA2-Regression.pdf - model testing must be performed on data separated from model training. So far, we have used a *validation set* approach, separating data into training and testing samples *(we'll cover more realistic approaches in Resampling)*.

In the real world, Business environments and data are very dynamic and that can render models quickly obsolete. Derivative trading models are often only good for minutes *(models are often trained in parallel and parameters are passed between models)*. Even in relatively static data, categories are constantly added and the relationships *(parameter values)* between drivers and business outcome constantly changes.

One way to deal with this is to generalize the models so that they will perform reasonably well across scenarios. And one way to generalize a model is cross-sampling

> *Note: We will cover cross-sampling in the resampling section. Cross-sampling pulls n "folds" (datasets) and uses them to train and test - selecting the "best" model (reducing the effects of overtraining). In most cases, cross sampling (and other "grid" training environments) will reduce (or shrink) coefficient effects (see ISL ch. 6) by adding a regularization term to "penalize" large coefficient values. This exercise is intended to build intuition about this process.*

Data *(esp. accounting)* are backward looking *(so the idea of prediction is a bit chimerical)*. Technologies emerge, new competitors and substitute products are introduced, interest rates change, component and service prices rise and fall, new practices are discovered... the world is constantly changing and there's often no data until it's too late *(that's another area where Bayesian modeling adds capabilities)*.

Scenario

This scenario is really artificial, but excuse that for illustration purposes.

Let's say we have 2 samples: the first we'll split into a training and test *(or validation)* set. Then, let's say the second is an actual dataset that we get later - after the model has been deployed.

So, dividing the MPG data:

```
rmse <- function(error)
{
  sqrt(mean(error^2))
}

# get some data

dfMPG <- mpg

# lets say we sample data at 2 different times
# and different manufacturers are available at different times

dfSample1 = filter(dfMPG, manufacturer %in% c("chevrolet", "dodge", "jeep"))
dfSample1 = select(dfSample1, manufacturer, hwy, displ)
dfSample2 = filter(dfMPG, manufacturer %in% c("volkswagen", "toyota", "audi") & fl != 'd' & (displ < 5
dfSample2 = select(dfSample2, manufacturer, hwy, displ)
```

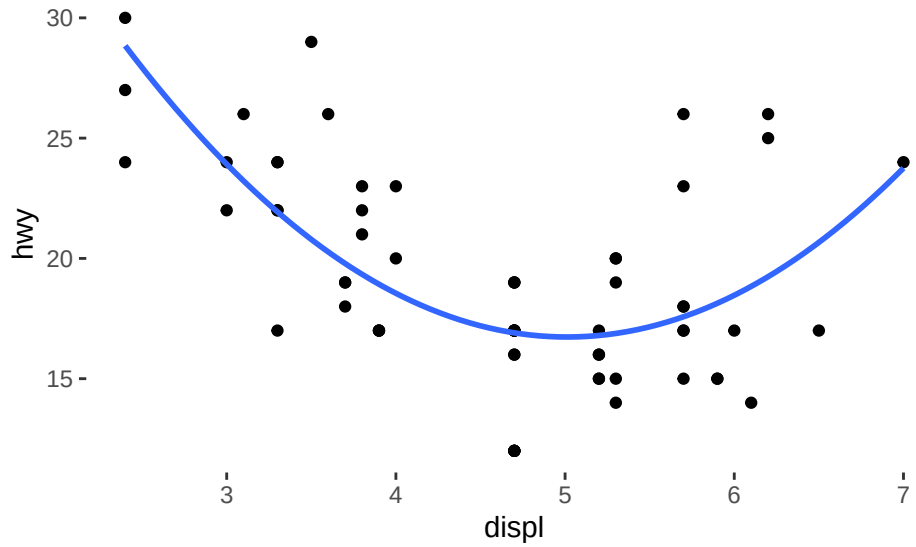
Build the model using normal equations and show rmse:

```
ModMatrix = model.matrix(hwy ~  displ + I(displ^2), data = dfSample1)
vY  = dfSample1$hwy

# solve for Betas
mBeta = as.numeric(solve(t(ModMatrix) %*% ModMatrix) %*% (t(ModMatrix) %*% vY))

# error estimated from training data
dfSample1$yhat1 = t(as.numeric(mBeta)%*%t(ModMatrix))
rmse(dfSample1$yhat1 - dfSample1$hwy)
```

```
## [1] 3.205879
```

Visualize the data with prediction:

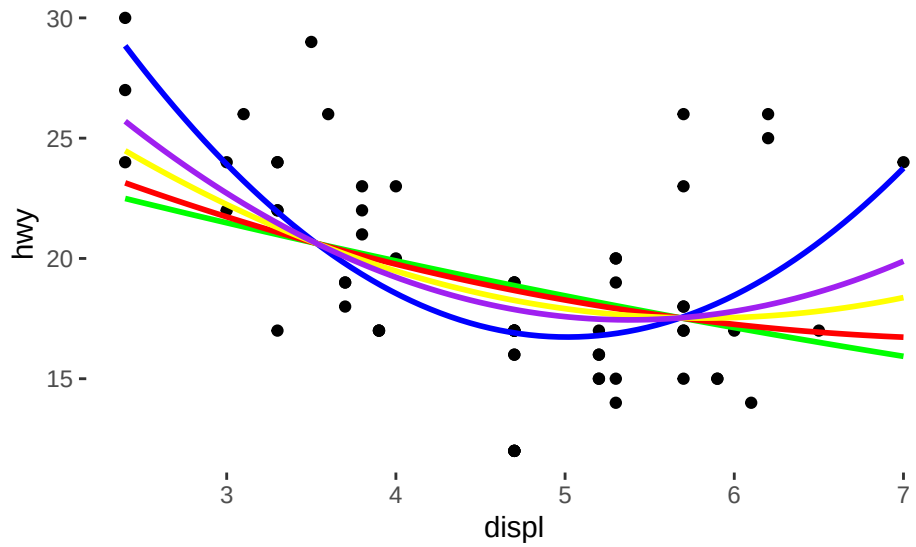
```
p = ggplot(data = dfSample1, aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth(aes(y = yhat1)) +
  theme(
    panel.background = element_rect(fill = "white")
  )
p
```



Now let's add regularization. You'll need the diagonal matrix (d) so that you can multiply the regularization factor by the model matrix (which is what you get from the model matrix. **We'll set the first element in d to 0 so we don't change the intercept term.** Also note that we create the diagonal dimensions to equal the columns in the model matrix *(refer back to linear algebra review)*.

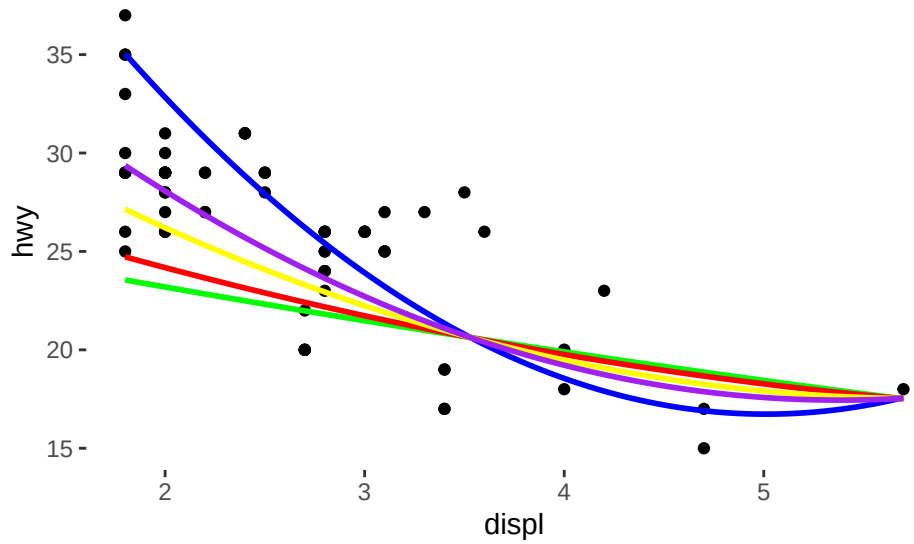
```
n = ncol(ModMatrix)
d = diag(1,n,n)
d[1,1] = 0 # set the intercept term

vBetaReg1 = as.numeric(solve(t(ModMatrix) %*% ModMatrix + (10 * d)) %*% (t(ModMatrix) %*% vY))
vBetaReg2 = as.numeric(solve(t(ModMatrix) %*% ModMatrix + (5 * d)) %*% (t(ModMatrix) %*% vY))
vBetaReg3 = as.numeric(solve(t(ModMatrix) %*% ModMatrix + (2 * d)) %*% (t(ModMatrix) %*% vY))
vBetaReg4 = as.numeric(solve(t(ModMatrix) %*% ModMatrix + (1 * d)) %*% (t(ModMatrix) %*% vY))
```

```
p = ggplot(data = dfSample1, aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth(aes(y = t(as.numeric(mBeta)%*%t(ModMatrix))), color = "blue") +
  geom_smooth(aes(y = t(as.numeric(vBetaReg1)%*%t(ModMatrix))), color = "green") +
  geom_smooth(aes(y = t(as.numeric(vBetaReg2)%*%t(ModMatrix))), color = "red") +
  geom_smooth(aes(y = t(as.numeric(vBetaReg3)%*%t(ModMatrix))), color = "yellow") +
  geom_smooth(aes(y = t(as.numeric(vBetaReg4)%*%t(ModMatrix))), color = "purple") +
  theme(
    panel.background = element_rect(fill = "white")

  )
p
```



Notice how the larger the reg term the straighter the line *(the more shrinkage effect on the coefficient)*. Now let's apply these models to the second dataset:

```
ModMatrix2 = model.matrix(hwy ~  displ + I(displ^2), data = dfSample2)

p = ggplot(data = dfSample2, aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth(aes(y = t(as.numeric(mBeta)%*%t(ModMatrix2))), color = "blue") +
  geom_smooth(aes(y = t(as.numeric(vBetaReg1)%*%t(ModMatrix2))), color = "green") +
  geom_smooth(aes(y = t(as.numeric(vBetaReg2)%*%t(ModMatrix2))), color = "red") +
  geom_smooth(aes(y = t(as.numeric(vBetaReg3)%*%t(ModMatrix2))), color = "yellow") +
  geom_smooth(aes(y = t(as.numeric(vBetaReg4)%*%t(ModMatrix2))), color = "purple") +
  theme(
    panel.background = element_rect(fill = "white") # I do this so it's easier to see in  class
  )
p
```



And estimate error:

```
rmse(t(as.numeric(mBeta)%*%t(ModMatrix2)) - dfSample2$hwy)
```

```
## [1] 3.956242
```

```
rmse(t(as.numeric(vBetaReg1)%*%t(ModMatrix2)) - dfSample2$hwy)
```

```
## [1] 5.373927
```

```
rmse(t(as.numeric(vBetaReg2)%*%t(ModMatrix2)) - dfSample2$hwy)
```

```
## [1] 4.795142
```

```
rmse(t(as.numeric(vBetaReg3)%*%t(ModMatrix2)) - dfSample2$hwy)
```

```
## [1] 3.758454
```

```
rmse(t(as.numeric(vBetaReg4)%*%t(ModMatrix2)) - dfSample2$hwy)
```

```
## [1] 3.136898
```

Notice that our "best" model *(in rmse terms)* is the vBetaReg4, in which we used a regularization penalty of 1 *(models with small, but significant, penalties performed better on new data vs. unpenalized normal equation - which is the same you would get with lm).*

So that's the idea of regularization. When we get into Bayesian analysis, we'll se how we use prior knowledge and pooling to generalize models in more sophisticated ways.