Lasso

Brooke Anderson

March 3, 2016

Reminder– best model to date

Add spline with knots at high temperature– on Kaggle: 0.41048.

```
mod_4 <- glm(count ~ year*hour*workingday*season +
               weather +
               ns(temp, knots = c(30, 35)),
             data = train, family = quasipoisson)
```

Lasso with caret

caret lets you fit a lasso model with glmnet, which lets you:

- Optimize on both complexity parameter (λ) and elastic-net mixing parameter (α)
- Use RMSLE when tuning
- Fit different flavors of GLM

See here for more on glmnet.

Lasso with `caret`

Elastic-net mixing parameter:

$$\alpha = \begin{cases} 0 & \text{ridge penalty} \\ 1 & \text{lasso penalty} \end{cases}$$

Non-zero variance predictors

I was getting some warnings about non-zero variance predictors, so I found and removed them using `nearZeroVar`:

```
my_train <- select(train, -datetime, -registered,
                   - casual)
my_train <- model.matrix(count ~ year * season *
                           workingday * hour +
                           holiday + temp + atemp +
                           humidity + windspeed +
                           month + yday + weather,
                         data = my_train)
nzv <- nearZeroVar(my_train)
my_train <- my_train[, -nzv]
```

Non-zero variance predictors

I need to do the same thing with the testing data (notice I'm using the `nzv` vector I measured from the training data, not doing a new one for the testing data):

```
my_test <- select(test, -datetime)
my_test <- model.matrix( ~ 1 + year * season *
                           workingday * hour +
                           holiday + temp + atemp +
                           humidity + windspeed +
                           month + yday + weather,
                         data = my_test)
my_test <- my_test[, -nzv]
```

Linear regression with glmnet

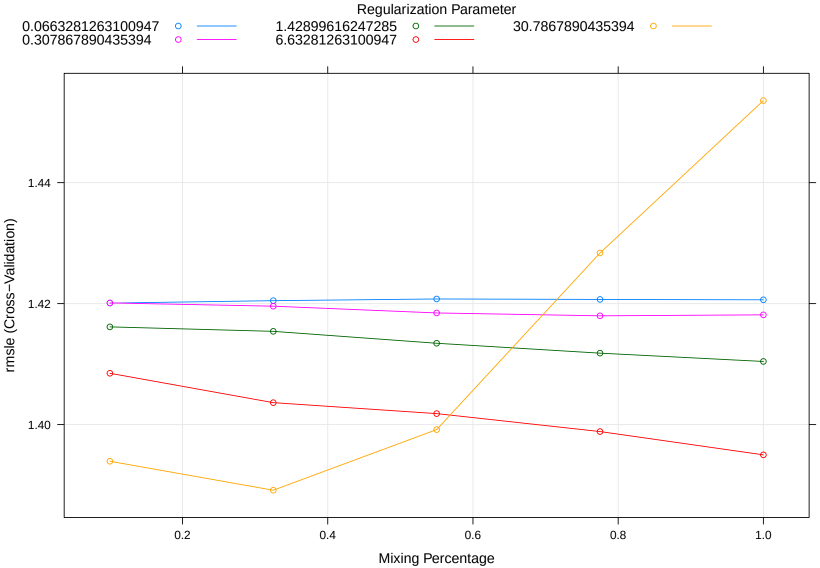
```
rmsle_fun <- function(data, lev = NULL,
                      model = NULL, ...){
  data$pred[data$pred < 0] <- 0
  log_p_1 <- log(data$pred + 1)
  log_a_1 <- log(data$obs + 1)
  sle <- (log_p_1 - log_a_1)^2
  rmsle <- sqrt(mean(sle))
  names(rmsle) <- "rmsle"
  return(rmsle)
}

fitControl <- trainControl(method = "cv",
                           number = 5,
                           summaryFunction = rmsle_fun)
```

Linear regression with glmnet

```
mod_1 <- train(y = train$count,
               x = my_train,
               preProcess = c("center", "scale"),
               method = "glmnet",
               trControl = fitControl,
               metric = "rmsle",
               maximize = FALSE,
               family = "gaussian",
               tuneLength = 5)
```

Linear regression with `glmnet`



Linear regression with glmnet

```
rmsle <- function(train_preds, actual_preds){
  train_preds[train_preds < 0] <- 0
  log_p_1 <- log(train_preds + 1)
  log_a_1 <- log(actual_preds + 1)
  sle <- (log_p_1 - log_a_1)^2
  rmsle <- sqrt(mean(sle))
  return(rmsle)
}

train_preds <- predict(mod_1, newdata = my_train)
train_preds[train_preds < 0] <- 0
rmsle(train_preds, train$count)
```

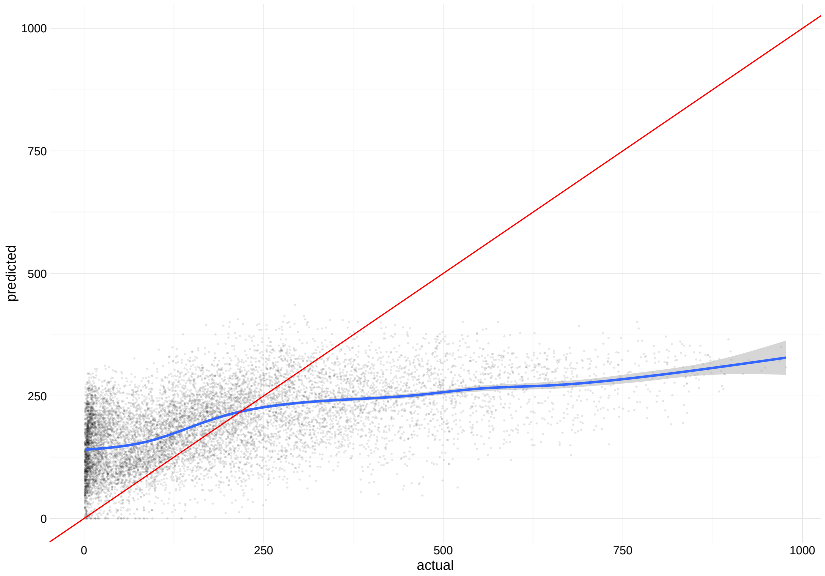
```
## [1] 1.388928
```

Linear regression with glmnet

```
test_preds <- predict(mod_1, newdata = my_test)
test_preds[test_preds < 0] <- 0
write_test_preds(test_preds,
                 mod = "elastic_net_gaussian")
```

On Kaggle, this got 1.39085.

Linear regression with glmnet



Poisson GLM with `glmnet`

From Hastie and Qian:

"The log-likelihood for observations $\{x_i, y_i\}_1^N$ is given by:

$$l(\beta|X, Y) = \sum_{i=1}^{N} (y_i(\beta_0 + \beta' x_i) - e^{\beta_0 + \beta^T x_i})$$

Poisson GLM with `glmnet`

"As before, we optimize the penalized log-likelihood:

$$\min_{\beta_0,\beta} -\frac{1}{N} l(\beta|X,Y) + \lambda\left((1-\alpha)\sum_{i=1}^{N}\beta_i^2/2) + \alpha\sum_{i=1}^{N}|\beta_i|\right)$$

Poisson GLM with `glmnet`

"Glmnet uses an outer Newton loop, and an inner weighted least-squares loop (as in logistic regression) to optimize this criterion."

Poisson GLM with glmnet

You need a new rmsle_fun function definition (notice you need to take the exponential of the predicted values):

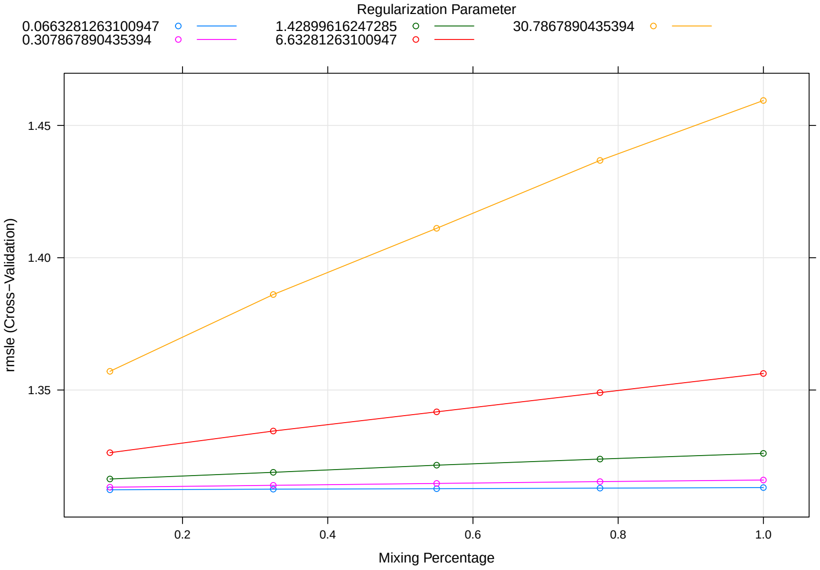
```
rmsle_fun <- function(data, lev = NULL,
                      model = NULL, ...){
  log_p_1 <- log(exp(data$pred) + 1)
  log_a_1 <- log(data$obs + 1)
  sle <- (log_p_1 - log_a_1)^2
  rmsle <- sqrt(mean(sle))
  names(rmsle) <- "rmsle"
  return(rmsle)
}
```

Poisson GLM with `glmnet`

Then train the model:

```
mod_2 <- train(y = train$count,
               x = my_train,
               preProcess = c("center", "scale"),
               method = "glmnet",
               trControl = fitControl,
               metric = "rmsle",
               maximize = FALSE,
               family = "poisson",
               tuneLength = 5)
```

Poisson GLM with `glmnet`



Poisson GLM with glmnet

```r
rmsle <- function(train_preds, actual_preds){
  log_p_1 <- log(train_preds + 1)
  log_a_1 <- log(actual_preds + 1)
  sle <- (log_p_1 - log_a_1)^2
  rmsle <- sqrt(mean(sle))
  return(rmsle)
}

train_preds <- predict(mod_2, newdata = my_train)
train_preds <- exp(train_preds)
rmsle(train_preds, train$count)
```

```
## [1] 1.311138
```

Poisson GLM with glmnet

```
test_preds <- predict(mod_2, newdata = my_test)
test_preds <- exp(test_preds)
write_test_preds(test_preds,
                 mod = "elastic_net_poisson")
```

The score on Kaggle was 1.37032.

Poisson GLM with `glmnet`