ATUS Microdata Forecasting (Example for Portfolio)

Charles Ye

```r
library(tidyverse)
library(xtable)
library(glmnet) # Elastic Net
library(e1071) # SVM

# Settings for PDF compilation
knitr::opts_chunk$set(
  echo = T, results = 'asis',
  fig.width = 10, fig.height = 4, out.width = '5in', out.height = '2in',
  fig.pos = '!h', fig.align = 'center', warnings = F, message = F
  )
options(xtable.include.rownames = F)
options(xtable.size = 'footnotesize')
options(xtable.table.placement = '!h')
```

1 Elastic Net Estimation

We use a microdata dataset compiled from the American Time Use Survey (ATUS), see `https://www.bls.gov/tus/`.

```r
## Load Data from ATUS
tmp = tempfile()
download.file('https://www.bls.gov/tus/special.requests/atusresp-2019.zip', tmp)
df1 = unz(tmp, 'atusresp_2019.dat') %>% readr::read_delim(., delim = ',')
download.file('https://www.bls.gov/tus/special.requests/atussum-2019.zip', tmp)
df2 = unz(tmp, 'atussum_2019.dat') %>% readr::read_delim(., delim = ',')
download.file('https://www.bls.gov/tus/special.requests/atuscps-2019.zip', tmp)
df3 = unz(tmp, 'atuscps_2019.dat') %>% readr::read_delim(., delim = ',')
# Join datasets together by hh respondent ID
# (join ATUS-CPS by both hh & individual respondent ID)
df =
  df1 %>%
  dplyr::inner_join(., df2, by = 'TUCASEID') %>%
  dplyr::inner_join(., df3, by = c('TUCASEID', 'TULINENO'))
unlink(tmp)
rm(df1, df2, df3, tmp)

# Data dictionary https://www.bls.gov/tus/atusintcodebk19.pdf
sleepDf =
  df %>%
  dplyr::transmute(
    .,
```

```
    time_sleeping = t010101,
    time_insomnia = t010102,
    age = TEAGE,
    is_male = ifelse(TESEX == 1, 1, 0),
    is_student = ifelse(TESCHFT == 1, 1, 0),
    is_employed = ifelse(TELFS.x %in% c(1, 2), 1, 0),
    has_children = ifelse(TRNHHCHILD == 1, 1, 0),
    number_children = TRCHILDNUM.x,
    age_youngest_child = TRYHHCHILD.x,
    weekly_earnings = PEERN,
    hh_size = HRNUMHOU,
    spouse_hours = TESPUHRS,
    hours_working = TEHRUSLT.x,
    time_alone = TRTALONE_WK,
    time_childcare = TRTCHILD,
    time_family = TRTFAMILY,
    time_friends = TRTFRIEND,
    time_eldercare = TRTEC.x
    ) %>%
  dplyr::mutate_all(., function(x) as.numeric(x))

rm(df)
```

Our dataset comprises of 9435 observations and 18 variables: *time_sleeping, time_insomnia, age, is_male, is_student, is_employed, has_children, number_children, age_youngest_child, weekly_earnings, hh_size, spouse_hours, hours_working, time_alone, time_childcare, time_family, time_friends, time_eldercare.* The first variable, number of minutes slept in the previous day, will be the dependent variable. The other 17 variables have been selected from the original ATUS dataset as they seem likely to have some relationship with time sleeping; these variables will be the independent variables in our models.

As an example, the below table shows the first 10 observations and first 8 variables in our dataset.

```
sleepDf %>%
  .[1:10, 1:8] %>%
  xtable(., caption = 'First 5 Observations') %>%
  print(.)
```

time_sleeping	time_insomnia	age	is_male	is_student	is_employed	has_children	number_children
660.00	0.00	85.00	0.00	0.00	0.00	0.00	0.00
540.00	0.00	25.00	0.00	0.00	1.00	0.00	1.00
645.00	0.00	20.00	0.00	0.00	1.00	0.00	0.00
375.00	0.00	61.00	0.00	0.00	0.00	0.00	0.00
480.00	0.00	34.00	1.00	0.00	1.00	0.00	1.00
365.00	0.00	53.00	0.00	0.00	1.00	0.00	0.00
900.00	0.00	26.00	0.00	0.00	1.00	1.00	3.00
470.00	0.00	45.00	1.00	0.00	1.00	0.00	4.00
515.00	0.00	85.00	1.00	0.00	0.00	0.00	0.00
230.00	0.00	74.00	1.00	0.00	0.00	0.00	0.00

Table 1: First 5 Observations

The dataset is then split into a test set and a training set.

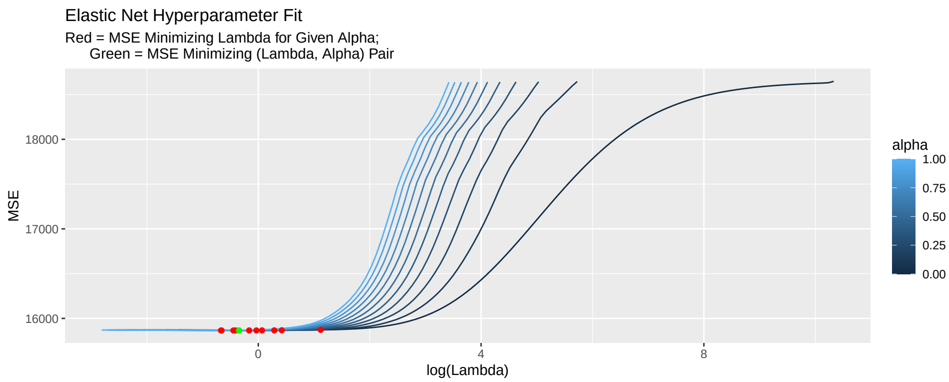
```
trainDf = sleepDf[1:(floor(nrow(sleepDf)/2)),]
testDf = sleepDf[floor(nrow(sleepDf)/2 + 1):(nrow(sleepDf)),]
```

Next, we estimate hyperparameters to conduct elastic net regularization on the model. We iterate through $\alpha = 0, .1, .2, \ldots, 1$ and use the `cv.glmnet` function to iterate over λ values for each α. At each (α, λ) pair, the MSE resulting from 10-fold cross-validation is conducted. The graph below shows the minimum λ value at each α, as well as the (α, λ) pair resulting in the lowest overall MSE. The corresponding table shows the exact values for the hyperparmeters which give the lowest overall MSE.

```r
set.seed(123123)
folds = sample(1:10, nrow(trainDf), replace = TRUE)

# Iterate through alpha = 0, .1, ..., 1 and select the optimal lambda at each
glmResult = lapply(seq(0, 1, .1), function(.alpha) {
  cv =
    glmnet::cv.glmnet(
      x = trainDf %>% dplyr::select(., -time_sleeping) %>% as.matrix(.),
      y = trainDf %>% dplyr::select(., time_sleeping) %>% as.matrix(.),
      foldid = folds,
      alpha = .alpha
    )

  tibble(alpha = .alpha, lambda = cv$lambda, mse = cv$cvm) %>%
    dplyr::mutate(., min_lambda_for_given_alpha = (mse == min(mse))) %>%
    return(.)
  }) %>%
  dplyr::bind_rows(.) %>%
  dplyr::mutate(., min_overall = (mse == min(mse)))

glmOptim = glmResult %>% dplyr::filter(., min_overall == TRUE)

cvPlot =
  glmResult %>%
  ggplot(.) +
  geom_line(aes(x = log(lambda), y = mse, group = alpha, color = alpha)) +
  geom_point(
    data = glmResult %>% dplyr::filter(., min_lambda_for_given_alpha == TRUE),
    aes(x = log(lambda), y = mse), color = 'red'
    ) +
  geom_point(
    data = glmResult %>% dplyr::filter(., min_overall == TRUE),
    aes(x = log(lambda), y = mse), color = 'green'
    ) +
  labs(
    x = 'log(Lambda)', y = 'MSE', color = 'alpha',
    title = 'Elastic Net Hyperparameter Fit',
    subtitle = 'Red = MSE Minimizing Lambda for Given Alpha;
    Green = MSE Minimizing (Lambda, Alpha) Pair'
    )
```

Next, we estimate hyperparameters to conduct elastic net regularization on the model. We iterate through $\alpha = 0, .1, .2, \ldots, 1$ and use the `cv.glmnet` function to iterate over λ values for each α. At each (α, λ) pair, the MSE resulting from 10-fold cross-validation is conducted. The graph below shows the minimum λ value at each α, as well as the (α, λ) pair resulting in the lowest overall MSE. The corresponding table shows the exact values for the hyperparmeters which give the lowest overall MSE.

```r
print(cvPlot)
```




```r
glmOptim %>%
  xtable(., caption = 'Optimal Hyperparameters') %>%
  print(.)
```

alpha	lambda	mse	min_lambda_for_given_alpha	min_overall
0.60	0.71	15865.17	TRUE	TRUE

Table 2: Optimal Hyperparameters

These hyperparameters are then used to estimate the coefficients, shown in the below table. Note that two coefficients have been shrunken to zero and effectively removed from the regression, indicating these coefficients had little predictive power.

```r
glmObj =
  glmnet::glmnet(
    x = trainDf %>% dplyr::select(., -time_sleeping) %>% as.matrix(.),
    y = trainDf %>% dplyr::select(., time_sleeping) %>% as.matrix(.),
    alpha = glmOptim$alpha,
    lambda = glmOptim$lambda
  )

coefMat = glmObj %>% coef(.) %>% as.matrix(.)

coefMat %>%
  as.data.frame(.) %>%
  rownames_to_column(., var = 'Covariate') %>%
  setNames(., c('Covariate', 'Estimate')) %>%
  xtable(., caption = 'Elastic Net Estimates', digits = 5) %>%
  print(.)
```

We then multiply the coefficient matrix by the test data matrix to get the predicted values of `time_sleep`. Then we subtract these from the actual test data to get the residuals. Goodness-of-fit statistics are shown below.

```r
# OOS fitting
oosFit =
  testDf %>%
  dplyr::select(., -time_sleeping) %>%
  dplyr::bind_cols(constant = 1, .) %>%
```

Covariate	Estimate
(Intercept)	729.06338
time_insomnia	-0.49894
age	-0.60369
is_male	-3.47788
is_student	-4.48876
is_employed	-20.12913
has_children	17.40865
number_children	-4.81052
age_youngest_child	0.00000
weekly_earnings	-0.00055
hh_size	-1.38152
spouse_hours	-0.20265
hours_working	-0.92708
time_alone	-0.20383
time_childcare	0.00000
time_family	-0.12707
time_friends	-0.14868
time_eldercare	-0.01909

Table 3: Elastic Net Estimates

```
    as.matrix(.) %>%
    {. %*% coefMat} %>%
    as.data.frame(.) %>%
    as_tibble(.) %>%
    setNames(., 'yhat')

  # Get residuals and goodness-of-fit statistics
  gofDf =
    oosFit %>%
    dplyr::bind_cols(., y = testDf$time_sleeping) %>%
    dplyr::mutate(., resids = y - yhat) %>%
    dplyr::summarize(., MAE = mean(abs(resids)), SSE = sum(resids^2), MSE = mean(resids^2))

  gofDf %>%
    xtable(., caption = 'Elastic Net OOS Goodness-of-Fit') %>%
    print(.)
```

MAE	SSE	MSE
93.20	70222973.69	14884.06

Table 4: Elastic Net OOS Goodness-of-Fit

Finally, we run a typical OLS regression on the same training dataset. Coefficients are shown below.

```
  lm(time_sleeping ~ ., sleepDf) %>%
    xtable(., caption = 'OLS Regression Results') %>%
    print(., include.rownames = TRUE)
```

We find the predicted values by using the OLS estimated coefficients on the test data matrix. Residuals are calculated and goodness-of-fit statistics are shown below.

```
  # Now compare to regular OLS
  olsOosFit =
    testDf %>%
    dplyr::select(., -time_sleeping) %>%
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	731.1363	8.1734	89.45	0.0000
time_insomnia	-0.5490	0.0407	-13.47	0.0000
age	-0.6415	0.0999	-6.42	0.0000
is_male	-1.4873	2.6397	-0.56	0.5732
is_student	-5.8462	6.7815	-0.86	0.3887
is_employed	-27.5152	4.7443	-5.80	0.0000
has_children	4.4790	12.1576	0.37	0.7126
number_children	-8.0169	2.3913	-3.35	0.0008
age_youngest_child	0.7858	0.3110	2.53	0.0115
weekly_earnings	0.0001	0.0002	0.39	0.6959
hh_size	-1.0306	1.6631	-0.62	0.5355
spouse_hours	-0.2352	0.0684	-3.44	0.0006
hours_working	-0.8608	0.1014	-8.49	0.0000
time_alone	-0.1995	0.0061	-32.49	0.0000
time_childcare	0.0059	0.0086	0.68	0.4946
time_family	-0.1295	0.0071	-18.19	0.0000
time_friends	-0.1499	0.0112	-13.34	0.0000
time_eldercare	-0.0456	0.0198	-2.31	0.0210

Table 5: OLS Regression Results

```
    dplyr::bind_cols(constant = 1, .) %>%
    as.matrix(.) %>%
    {. %*% coef(lm(time_sleeping ~ ., sleepDf))} %>%
    as.data.frame(.) %>%
    as_tibble(.) %>%
    setNames(., 'yhat')

  olsGofDf =
    olsOosFit %>%
    dplyr::bind_cols(., y = testDf$time_sleeping) %>%
    dplyr::mutate(., resids = y - yhat) %>%
    dplyr::summarize(., MAE = mean(abs(resids)), SSE = sum(resids^2), MSE = mean(resids^2))

  olsGofDf %>%
    xtable(., caption = 'OLS OOS Goodness-of-Fit') %>%
    print(.)
```

MAE	SSE	MSE
92.78	69608439.69	14753.80

Table 6: OLS OOS Goodness-of-Fit

OLS ends up providing a better out-of-sample fit than the elastic net process. This is likely because in the OLS results, almost all the regression coefficients are significant, suggesting that they all have some predictive power on `time_sleeping`. This implies that any shrinkage of covariates from the elastic net regularization process will have little positive effect on out-of-sample forecasting. This tells us that standard OLS may be a better choice than machine learning techniques when the covariates are intuitively and clearly relevant to the dependent variable.

2 SVM Estimation

```r
# Create dataset
sleepDf2 =
  sleepDf %>%
  dplyr::mutate(., has_insomnia = ifelse(time_insomnia > 0, 1, 0)) %>%
  dplyr::mutate(., has_insomnia = as.factor(has_insomnia)) %>%
  dplyr::select(., -time_insomnia)

trainDf = sleepDf2[1:(floor(nrow(sleepDf2)/2)),]
testDf = sleepDf2[floor(nrow(sleepDf2)/2 + 1):(nrow(sleepDf2)),]

set.seed(12345)
# Tune hyperparameters of SVM
tuneRes =
  tune.svm(
    has_insomnia ~ ., data = sleepDf2, kernel = 'radial',
    type = 'C-classification', cost = 2^(0:3)
    )

tuneRes2 =
  tune.svm(
    has_insomnia ~ ., data = sleepDf2,
    kernel = 'linear', type = 'C-classification', cost = 2^(0:3)
    )

# Do OOS testing of SVM
svmFit =
  tuneRes$best.model %>%
  predict(., newdata = testDf %>% dplyr::select(., -has_insomnia))

svmGofDf =
  tibble(yhat = svmFit, y = testDf$has_insomnia) %>%
  dplyr::mutate(., resids = as.numeric(y) - as.numeric(yhat)) %>%
  dplyr::summarize(., MAE = mean(abs(resids)), SSE = sum(resids^2), MSE = mean(resids^2))

# Run regular logit model and do OOS testing
glmGofDf =
  glm(has_insomnia ~ ., data = trainDf, family = 'binomial') %>%
  predict(
    .,
    newdata = testDf %>% dplyr::select(., -has_insomnia),
    type = 'response'
    ) %>%
  tibble(yhat = ., y = testDf$has_insomnia) %>%
  dplyr::mutate(., resids = as.numeric(y) - as.numeric(yhat)) %>%
  dplyr::summarize(., MAE = mean(abs(resids)), SSE = sum(resids^2), MSE = mean(resids^2))
```

We now alter the dataset used previously to create a new dependent variable, `has_insomnia`, a binary variable indicating whether the individual experienced any minutes of insomnia over the past week. Our independent variables are now *time_sleeping, age, is_male, is_student, is_employed, has_children, number_children, age_youngest_child, weekly_earnings, hh_size, spouse_hours, hours_working, time_alone, time_childcare, time_family, time_friends, time_eldercare.*

As before, we break up the dataset into a testing and training dataset. We run two alternative kernels, radial and linear, and use the `tune` function to perform a grid search over the cost functions $2^0, 2^1, \ldots, 2^5$. We find that the lowest error is provided by the linear kernel with cost function 1.

We fit this SVM model to the test dataset and derive the following out-of-sample goodness-of-fit statistics for the residuals.

```
svmGofDf %>%
  xtable(., caption = 'SVM Goodness-of-Fit') %>%
  print(.)
```

MAE	SSE	MSE
0.08	369.00	0.08

Table 7: SVM Goodness-of-Fit

To benchmark the SVM model, we fit a standard logit model as well and derive the coefficient estimates below.

```
glm(has_insomnia ~ ., data = trainDf, family = 'binomial') %>%
  xtable(., caption = 'Logit Model Fit') %>%
  print(., include.rownames = TRUE)
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-0.4326	0.4696	-0.92	0.3570
time_sleeping	-0.0021	0.0004	-4.85	0.0000
age	0.0022	0.0043	0.50	0.6149
is_male	-0.3983	0.1173	-3.39	0.0007
is_student	-0.0694	0.2987	-0.23	0.8162
is_employed	-0.5222	0.2116	-2.47	0.0136
has_children	-1.1782	1.0181	-1.16	0.2472
number_children	-0.0777	0.1056	-0.74	0.4618
age_youngest_child	-0.0035	0.0143	-0.25	0.8055
weekly_earnings	-0.0000	0.0000	-0.93	0.3546
hh_size	0.0840	0.0694	1.21	0.2261
spouse_hours	0.0024	0.0030	0.81	0.4186
hours_working	-0.0034	0.0047	-0.71	0.4750
time_alone	-0.0007	0.0003	-2.44	0.0148
time_childcare	-0.0002	0.0004	-0.54	0.5903
time_family	-0.0012	0.0003	-3.83	0.0001
time_friends	-0.0021	0.0007	-3.23	0.0012
time_eldercare	-0.0016	0.0013	-1.23	0.2171

Table 8: Logit Model Fit

```
glmGofDf %>%
  xtable(., caption = 'Logit Model Goodness-of-Fit') %>%
  print(.)
```

MAE	SSE	MSE
1.00	5058.33	1.07

Table 9: Logit Model Goodness-of-Fit

Comparison of the out-of-sample testing results between the models suggests that SVM performs the better fit. This is likely because the data is high-dimensional but we have relatively few observations; moreover the independent variables likely have a nonlinear relationship with the dependent variable (e.g. demographic covariates like age tend to have nonlinear, nonmonotonic effects on a person's likelihood of having insomnia).