Selected Topics in Applied Machine Learning: An integrating view on data analysis and learning algorithms

ESSLLI '2015
Barcelona, Spain

http://ufal.mff.cuni.cz/esslli2015

Barbora Hladká
hladka@ufal.mff.cuni.cz

Martin Holub
holub@ufal.mff.cuni.cz

Charles University in Prague,
Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics

Block 4.1
Regularization practically for both MOV and VPR

	MOV	**VPR**
type of task	regression	classification
getting examples by	collecting	annotation
# of examples	100,000	250
# of features	32	363
categorical/binary	29/18	0/363
numerical	3	0
output values	1–5	5 discrete categories

Experiments in R – Outline

MOV

- Reporting results for cross-validation
- Fitting a linear model (see `lin-reg-mov-cv.R`)
- Fitting a ridge regression model (see `lin-reg-ridge-mov-cv.R`)
- Fitting a lasso model (see `lin-reg-lasso-mov-cv.R`)

VPR

- Reporting results for cross-validation
- Fitting a logistic regression model (see `log-reg-[cry|submit]-cv-manually.R`)
- Fitting a ridge regression model (see `log-reg-ridge-vpr-cv.R`)
- Fitting a lasso model (see `log-reg-lasso-vpr-cv.R`)

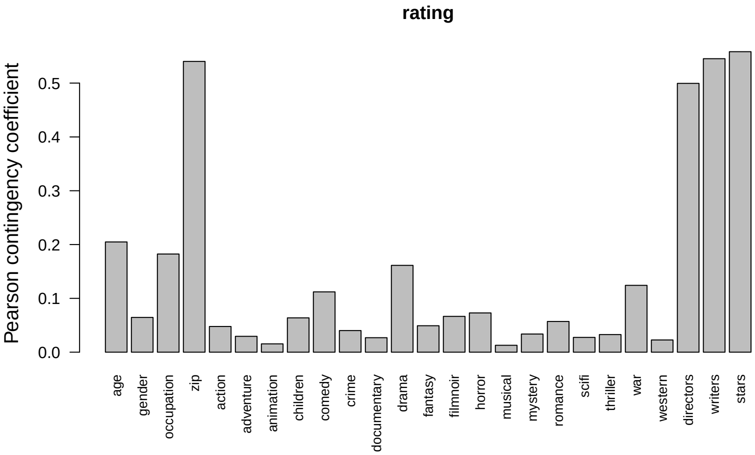
MOV – fitting a linear model

- Let's build a simple linear model for predicting user's rating of a given movie. We use the user's features and the movie's features only, i.e. we do not incorporate any similarity between users and between movies.

```
# 'features' are used for prediction
> fit <- lm(rating ~ features)
```

MOV – fitting a linear model

Recall association between categorical feature and target value



MOV – fitting a linear model

features	cross-validation MSE
IMDB_RATING	1.083
GENDER+IMDB_RATING	1.083
AGE+OCCUPATION+GENRE_DRAMA+IMDB_RATING	1.064

MOV – fitting a Ridge regression model

```
> library(glmnet)
# https://cran.r-project.org/web/packages/glmnet/glmnet.pdf
>
# features
> x <- model.matrix(rating ~ age+occupation+genre_drama
       + imdb_rating, examples)
>
# target values
> y <- data.matrix(examples$rating)
>
# run 5-cross-validation ridge regression (i.e. alpha = 0)
> fit <- cv.glmnet(x, y, foldid=foldid, alpha=0)
```

MOV – fitting a Ridge regression model

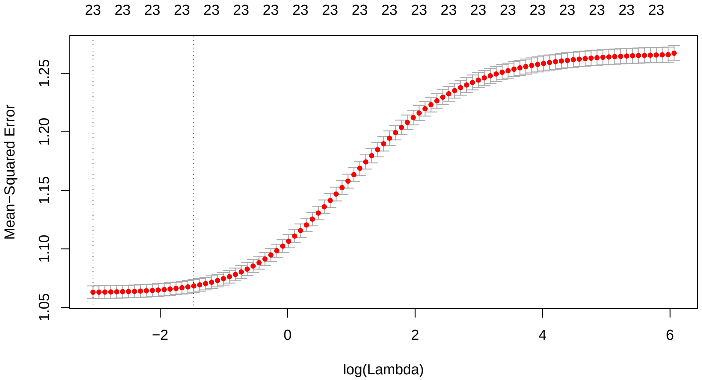
Explore `fit`

```
# we let glmnet() to choose its own sequence of lambda values
> fit$lambda
[1] 429.66248764 391.49245644 356.71334560 325.02391512 ...
[6] 269.84056440 245.86867237 224.02637716 204.12449125 ...
...
[21]  66.84156820  60.90354750  55.49304420  50.56319510  ...
...
[86]   0.15804618   0.14400579   0.13121272   0.11955614   ...
[91]   0.09925761   0.09043984   0.08240541   0.07508474   ...
[96]   0.06233667   0.05679885   0.05175300   0.04715540
```

MOV – fitting a Ridge regression model

Explore `fit` – cross-validation curve

```
> plot(fit)
```



MOV – fitting a Ridge regression model

Explore `fit` – cross-validation curve

```
> fit$lambda.min # lambda that gives minimum cv mse
[1] 0.0471554
> log(fit$lambda.min)
[1] -3.054307
> min(fit$cvm) # minimum cv mse
[1] 1.06399
# larger value of lambda whose cv mse is 1 standard error larger
> fit$lambda.1se
[1] 0.2089277
> log(fit$lambda.1se)
[1] -1.472733
```

MOV – fitting a Ridge regression model

Explore `fit`

- `df` is the number of non-zero parameters for a given lambda

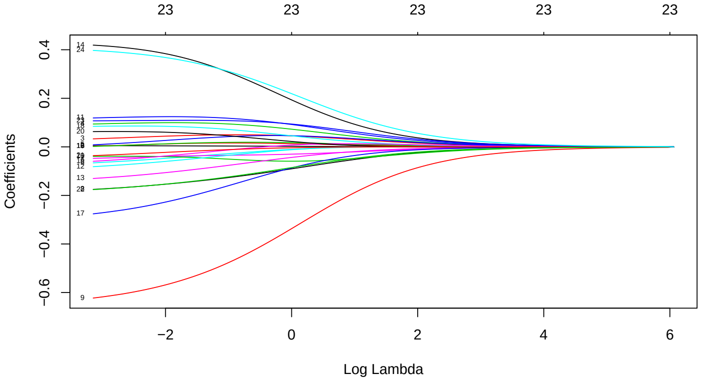
```
> fit$glmnet.fit

Call:  glmnet(x = x, y = y, alpha = 0)

       Df      %Dev    Lambda
  [1,] 23 3.564e-37 429.70000
  [2,] 23 1.010e-03 391.50000
  [3,] 23 1.108e-03 356.70000
  ...
[37,] 23 2.331e-02  15.09000
...
[100,] 23 1.616e-01   0.04297
```

MOV – fitting a Ridge regression model

Explore `fit` – regularization path

```
> plot(fit$glmnet.fit, "lambda",   label=TRUE)
```

- each curve corresponds to one feature



MOV – fitting a Ridge regression model

Unregularized estimates vs. Ridge regression estimates

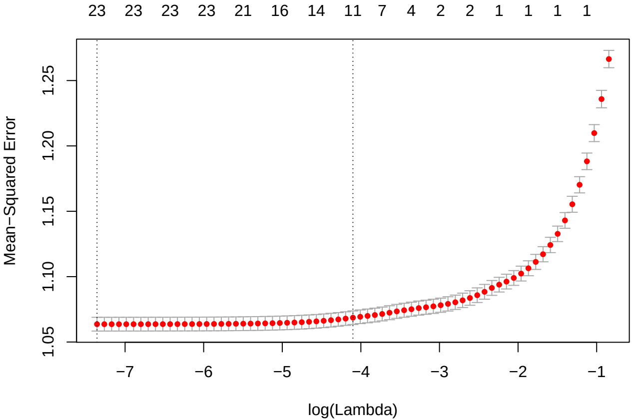
```
> ridge <- coef(fit, s=fit$lambda.min) # parameter values for lambda.min
> zero <- coef(fit, s = 0, exact=TRUE) # parameter values for lambda = 0
> cbind2(zero, ridge)
                                  1            1
(Intercept)             0.376634335  0.497590184
age                     0.005201767  0.004832107
occupationartist        0.022191412  0.033819976
occupationdoctor        0.086138329  0.094690343
occupationeducator     -0.006627488  0.009627136
occupationengineer     -0.080638616 -0.064661316
occupationentertainment -0.056201213 -0.046720724
occupationexecutive    -0.191292180 -0.173860994
occupationhealthcare   -0.656019848 -0.619830020
occupationhomemaker    -0.042836783 -0.039045713
occupationlawyer        0.111040316  0.119646464
occupationlibrarian    -0.098395097 -0.080264258
occupationmarketing    -0.148221157 -0.128767035
occupationnone          0.433419727  0.417505629
occupationother        -0.007124955  0.003446476
occupationprogrammer   -0.009180186  0.002451768
occupationretired      -0.308190962 -0.273113626
occupationsalesman      0.079477215  0.084773844
occupationscientist    -0.076566156 -0.058078408
occupationstudent       0.059671605  0.062926340
occupationtechnician   -0.048487291 -0.035228035
occupationwriter       -0.193246258 -0.174712643
genre_drama             0.105591480  0.107312053
imdb_rating             0.412526305  0.395788369
```

MOV – fitting a lasso model

```
# features
> x <- model.matrix(rating ~ age+occupation
                          + imdb_rating, examples)
>
# target values
> y <- data.matrix(examples$rating)
>
# run 5-cross-validation lasso (i.e. alpha = 1)
> fit <- cv.glmnet(x, y, foldid=foldid, alpha=1)
```

MOV – fitting a lasso model

Explore `fit` – cross-validation curve



MOV – fitting a lasso model

Unregularized vs. lambda.min vs. lambda.1se estimates

```
(Intercept)               0.377078574  0.3782515309  0.6078346257
age                       0.005207882  0.0050746589  0.0009571005
occupationartist          0.021374317  0.0237744629  .
occupationdoctor          0.085295012  0.0838152441  .
occupationeducator       -0.007487083  .             0.0044888344
occupationengineer       -0.081408873 -0.0722145447  .
occupationentertainment  -0.056924539 -0.0463603409  .
occupationexecutive      -0.192059048 -0.1813629805 -0.0578740291
occupationhealthcare     -0.656798789 -0.6453823905 -0.5014057727
occupationhomemaker      -0.043570398 -0.0254413127  .
occupationlawyer          0.110293177  0.1119697836  .
occupationlibrarian      -0.099137462 -0.0888058578  .
occupationmarketing      -0.148950014 -0.1369952599  .
occupationnone            0.432765259  0.4315957579  0.2224678511
occupationother          -0.007770340  .              .
occupationprogrammer     -0.009789887 -0.0007535344  .
occupationretired        -0.308978366 -0.2931022762 -0.0179402725
occupationsalesman        0.078868294  0.0788989435  .
occupationscientist      -0.077183517 -0.0653459574  .
occupationstudent         0.059128735  0.0634587473  .
occupationtechnician     -0.049085193 -0.0388110293  .
occupationwriter         -0.193878629 -0.1839234292 -0.0898684481
genre_drama               0.105586861  0.1045008045  0.0741112519
imdb_rating               0.412523572  0.4120395961  0.3984697536
```

MOV – Ridge regression and lasso

```
> lambda.min.lasso
[1] 0.0006380352
> lambda.min.ridge
[1] 0.0471554
> min.cv.mse.lasso
[1] 1.062645
> min.cv.mse.ridge
[1] 1.06399
# cv for lambda.1se is 1.067
```

VPR

We address a classification task for *cry*, namely

- binary classification – "1" vs. all
- multiclass classification – "1", "2", "4", "u", "x"

- Reporting results for cross-validation
- Fitting a logistic regression model (see `log-reg-vpr-cv.R`)
- Fitting a ridge regression model (see `log-reg-ridge-vpr-cv.R`)
- Fitting a lasso model (see `log-reg-lasso-vpr-cv.R`)

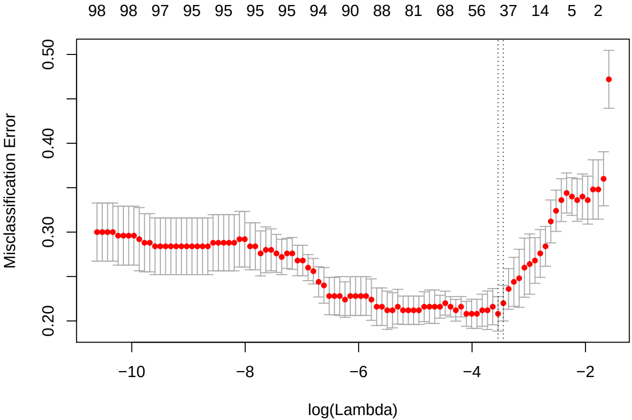
VPR – fitting a lasso model

Binary classification "1" vs. all

```
# filter out uneffective features
> ...
# get the number of features after filtering
168
# run 9-cross-validation lasso (i.e. alpha = 1)
> fit <- cv.glmnet(x, y, family = "binomial", foldid=foldid,
                  type.measure = "class", alpha=1)
> ...
> fit$lambda.min
[1] 0.028965
> min(fit$cvm)
[1]  0.208
```

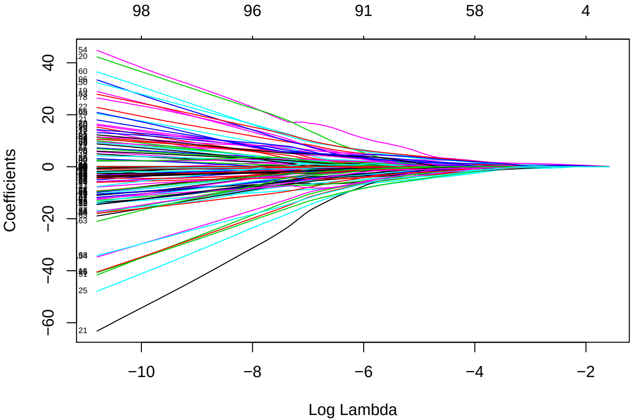
VPR – fitting a lasso model

Explore `fit` – cross-validation curve



VPR – fitting a lasso model

Explore `fit` – regularization path



VPR – fitting a lasso model

Lasso estimates vs. unregularized estimates

Number of non-zero parameters at lambda.min is 45 (out of 168)

```
> lasso <- coef(fit, s=fit$lambda.min)
> zero <- coef(fit, s = 0, exact=TRUE)
> cbind2(zero, lasso)
...
MF.3p_verbs           1.098401e+01  0.831887984
MF.3p_modal          -1.265944e+01  .
MF.3p_adverbial       1.195433e+00  .
MF.3p_to             -1.666759e+01  .
MF.3p_wh.pronoun     -2.713782e+01 -1.143650091
MF.3p_wh.adverb       3.544223e+01  .
MF.3p_be             -1.372386e+01  .
MF.2p_nominal        -1.431017e+01  .
MF.2p_adjective      -1.024928e+01  .
MF.2p_verbs           1.132535e+01  0.899228776
MF.2p_modal           8.648575e+00  .
MF.2p_adverbial       5.887964e+01  0.387374637
MF.2p_to             -2.418016e+01  .
MF.2p_wh.pronoun     -1.060266e+02 -0.293936826
MF.2p_wh.adverb       4.066672e+01  1.320097705
MF.2p_be             -3.675107e-01  .
...
MF.1p_modal          -1.635712e+01  .
MF.1p_adverbial      -1.900712e+01  .
...
```

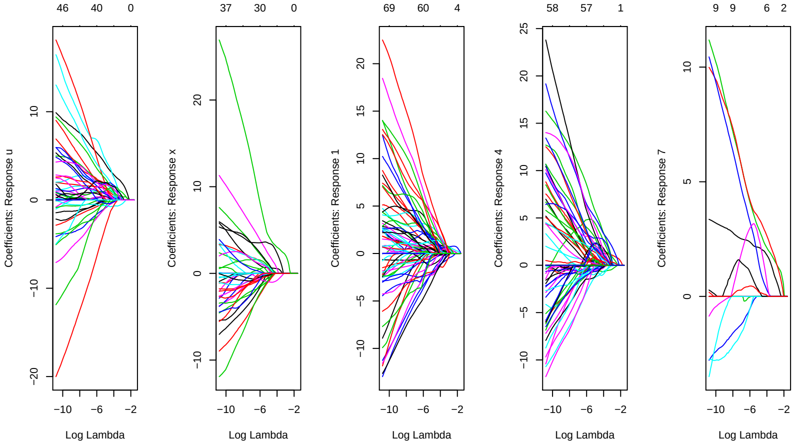
VPR – fitting a lasso model

Multiclass classification

```
# filter out uneffective features
> ...
> fit <- cv.glmnet(x, y, family = "multinomial", foldid=foldid,
                   type.measure = "class", alpha=1)
> ...
> min(fit$cvm)
[1] 0.308
> fit$lambda.min
[1] 0.04202326
```

VPR – fitting a lasso model

Multiclass classification



Block 4.2
Introduction to practical feature selection

Goal of the feature selection process = find a minimum set of variables that contain all the substantial information about predicting the target value

- reduced feature space dimension in the dataset
- enhanced generalization and improved prediction performance by reducing overfitting
- better chance to analyse the impact/importance of the features
- removing highly dependent features (some learning methods do not work well with them)
- lower model complexity and improved model interpretability
- feasible/shorter training times

Feature selection methods

Feature selection methods can be basically divided into

- **filters** – select feature subsets as a pre-processing step, independently of the learning method
- **wrappers** – use a machine learning algorithm in conjunction with internal cross validation procedure to score feature subsets by measuring their predictive power
- **embedded methods** – perform feature selection during the process of training