Multi-class Logistic, LDA and logistic lasso

ACM

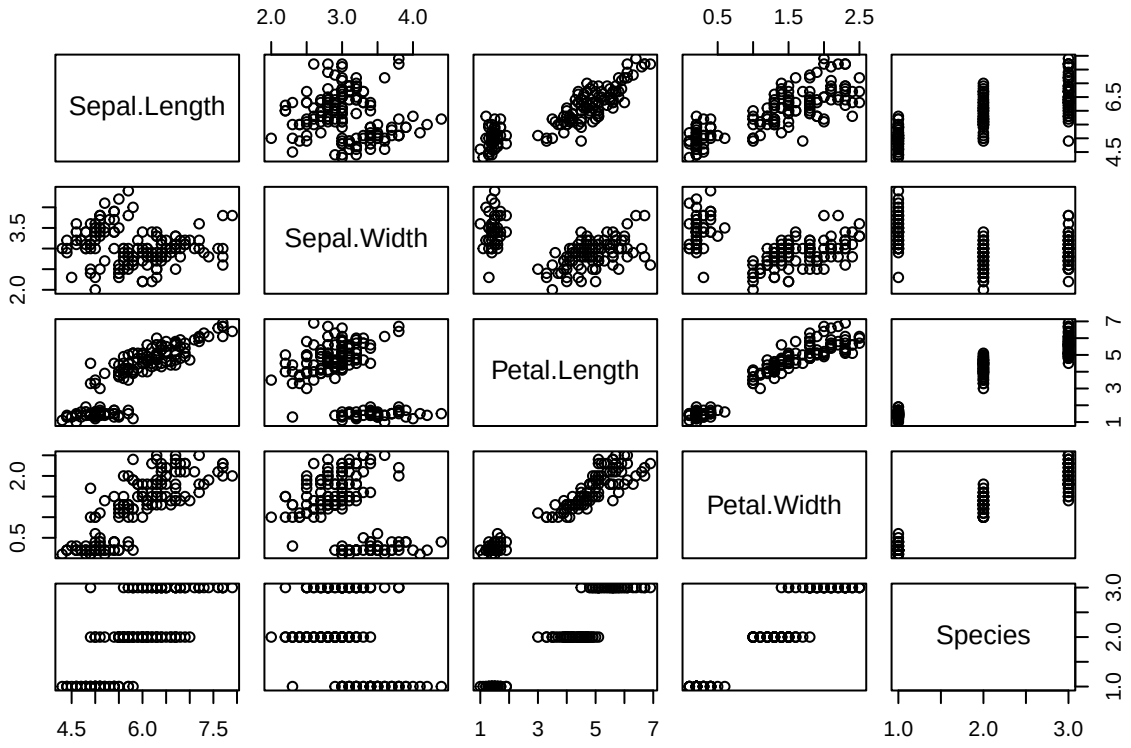
September 2023

Here we'll look at a quick example of multi-class LDA and logistic regression.

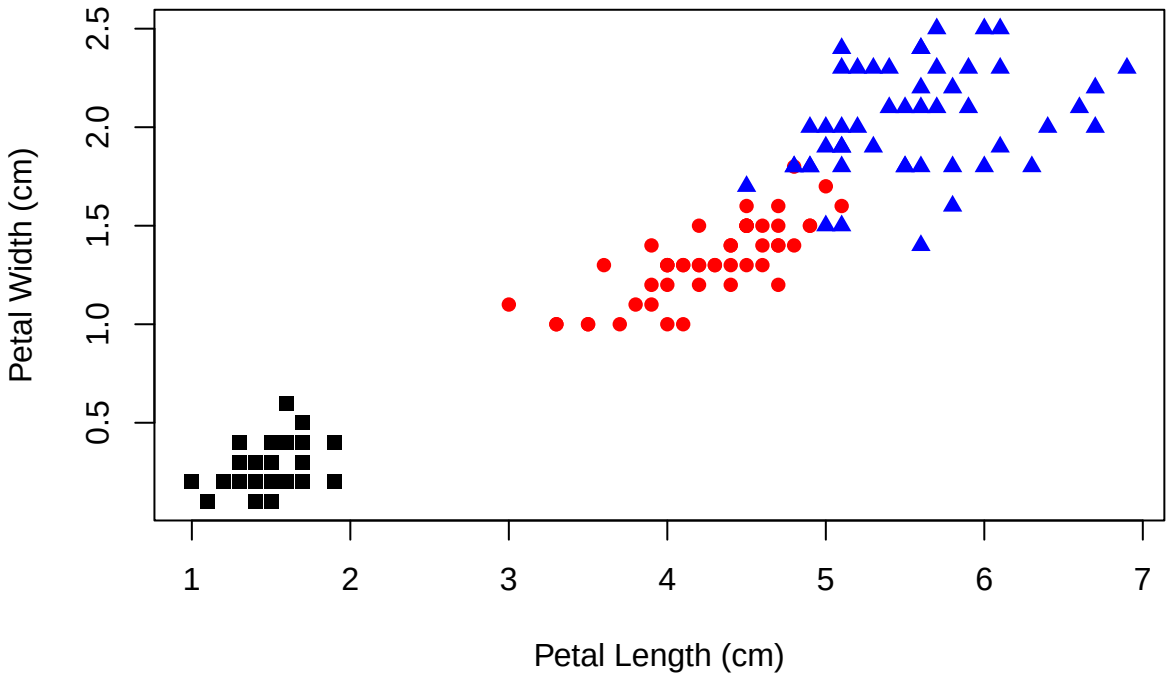
```
library(printr)
```

```
## Registered S3 method overwritten by 'printr':
##   method                from
##   knit_print.data.frame rmarkdown
```

```
library(ggplot2)
# Load the Iris data
data(iris)
plot(iris)
```




```
plot(Petal.Width ~ Petal.Length, col=c("black", "red", "blue")[Species],
     pch=(15:17)[Species], xlab="Petal Length (cm)",
     ylab="Petal Width (cm)", data=iris)
```




```
head(iris)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

```
library(MASS)
set.seed(9823479)
train <- sample(1:150, 100)
table(iris$Species[train])
```

setosa	versicolor	virginica
31	35	34

```
z <- lda(Species ~ ., data = iris, subset = train)
z
```

```
Call:
lda(Species ~ ., data = iris, subset = train)

Prior probabilities of groups:
    setosa versicolor  virginica
      0.31       0.35       0.34

Group means:
           Sepal.Length Sepal.Width Petal.Length Petal.Width
```

```
setosa         5.019355    3.412903     1.461290   0.2419355
versicolor     5.960000    2.825714     4.302857   1.3542857
virginica      6.567647    3.023529     5.555882   2.0411765

Coefficients of linear discriminants:
                    LD1        LD2
Sepal.Length  0.3263448  0.2583307
Sepal.Width   2.3175531 -2.6202166
Petal.Length -2.0326574  0.7725431
Petal.Width  -2.8992573 -2.7197269

Proportion of trace:
   LD1    LD2
0.9911 0.0089
```

Now let's predict for the test data:

```
pred <- predict(z, iris[-train, ])$class
tbl <- table(pred,iris$Species[-train])
prop.table(tbl)
```

pred/	setosa	versicolor	virginica
setosa	0.38	0.0	0.00
versicolor	0.00	0.3	0.02
virginica	0.00	0.0	0.30

```
sum(diag(prop.table(tbl)))
```

```
## [1] 0.98
```

```
library(nnet)
`?`(multinom)
```

```
Fit Multinomial Log-linear Models

Description:

     Fits multinomial log-linear models via neural networks.

Usage:

     multinom(formula, data, weights, subset, na.action,
              contrasts = NULL, Hess = FALSE, summ = 0, censored = FALSE,
              model = FALSE, ...)

Arguments:

 formula: a formula expression as for regression models, of the form
          'response ~ predictors'. The response should be a factor or a
          matrix with K columns, which will be interpreted as counts
          for each of K classes. A log-linear model is fitted, with
          coefficients zero for the first class. An offset can be
          included: it should be a numeric matrix with K columns if the
          response is either a matrix with K columns or a factor with K
          >= 2 classes, or a numeric vector for a response factor with
```

```
          2 levels. See the documentation of 'formula()' for other
          details.

    data: an optional data frame in which to interpret the variables
          occurring in 'formula'.

 weights: optional case weights in fitting.

  subset: expression saying which subset of the rows of the data should
          be used in the fit. All observations are included by default.

na.action: a function to filter missing data.

contrasts: a list of contrasts to be used for some or all of the
          factors appearing as variables in the model formula.

    Hess: logical for whether the Hessian (the observed/expected
          information matrix) should be returned.

    summ: integer; if non-zero summarize by deleting duplicate rows and
          adjust weights. Methods 1 and 2 differ in speed (2 uses 'C');
          method 3 also combines rows with the same X and different Y,
          which changes the baseline for the deviance.

censored: If Y is a matrix with 'K' columns, interpret the entries as
          one for possible classes, zero for impossible classes, rather
          than as counts.

   model: logical. If true, the model frame is saved as component
          'model' of the returned object.

     ...: additional arguments for 'nnet'
```

For this function the variables are supposed to be roughly 0 to 1. The fit is a little weird if they are scaled or if they are not scaled. Here, I don't scale them.

```
fit <- multinom(Species ~ ., data = iris)
```

```
## # weights:  18 (10 variable)
## initial  value 164.791843
## iter  10 value 16.177348
## iter  20 value 7.111438
## iter  30 value 6.182999
## iter  40 value 5.984028
## iter  50 value 5.961278
## iter  60 value 5.954900
## iter  70 value 5.951851
## iter  80 value 5.950343
## iter  90 value 5.949904
## iter 100 value 5.949867
## final  value 5.949867
## stopped after 100 iterations
```

```
# summarize the fit
summary(fit)
```

```
## Call:
## multinom(formula = Species ~ ., data = iris)
##
## Coefficients:
##            (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
## versicolor    18.69037    -5.458424   -8.707401     14.24477   -3.097684
## virginica    -23.83628    -7.923634  -15.370769     23.65978   15.135301
##
## Std. Errors:
##            (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
## versicolor    34.97116     89.89215    157.0415     60.19170    45.48852
## virginica     35.76649     89.91153    157.1196     60.46753    45.93406
##
## Residual Deviance: 11.89973
## AIC: 31.89973
```

Now we'll predict for the remaining subjects:

```
pred_log <- predict(fit, newdata = iris[-train,], "probs")
prec_class <- apply(pred_log,1,which.max)
tbl_2 <- table(prec_class,iris$Species[-train])
prop.table(tbl_2)
```

prec_class/	setosa	versicolor	virginica
1	0.38	0.0	0.00
2	0.00	0.3	0.02
3	0.00	0.0	0.30

```
sum(diag(prop.table(tbl_2)))
```

```
## [1] 0.98
```

```
library(glmnet)
```

```
Loading required package: Matrix
```

```
Loaded glmnet 4.1-8
```

```
## Here we fit the LASSO model to our data.  We'll look at this for the default
## nlambda = 100, and lambda.min.ratio = 0.0001.
`?`(glmnet)
```

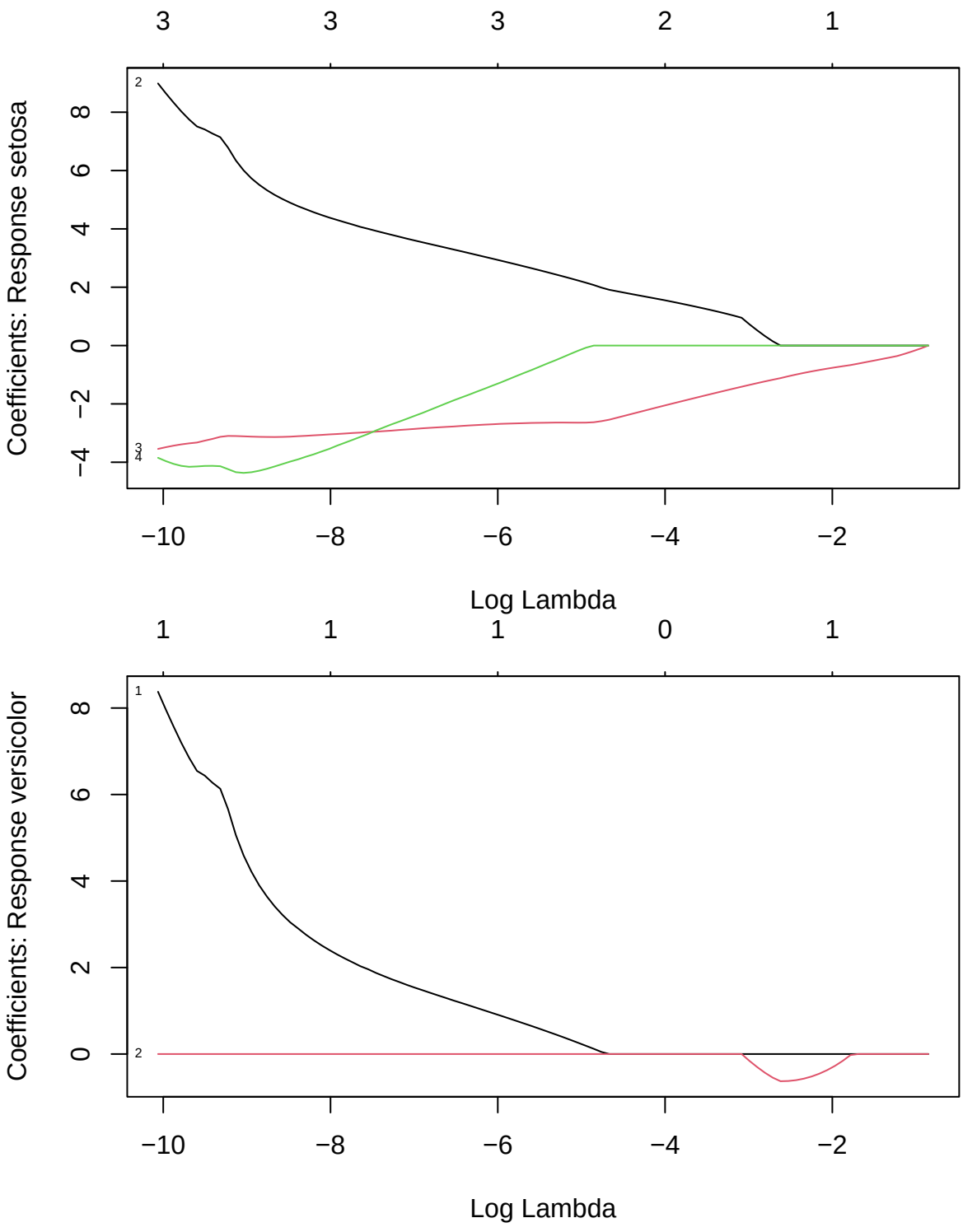
```
fit a GLM with lasso or elasticnet regularization

Description:

     Fit a generalized linear model via penalized maximum likelihood.
     The regularization path is computed for the lasso or elasticnet
     penalty at a grid of values for the regularization parameter
     lambda. Can deal with all shapes of data, including very large
     sparse data matrices. Fits linear, logistic and multinomial,
     poisson, and Cox regression models.

Usage:

     glmnet(
```

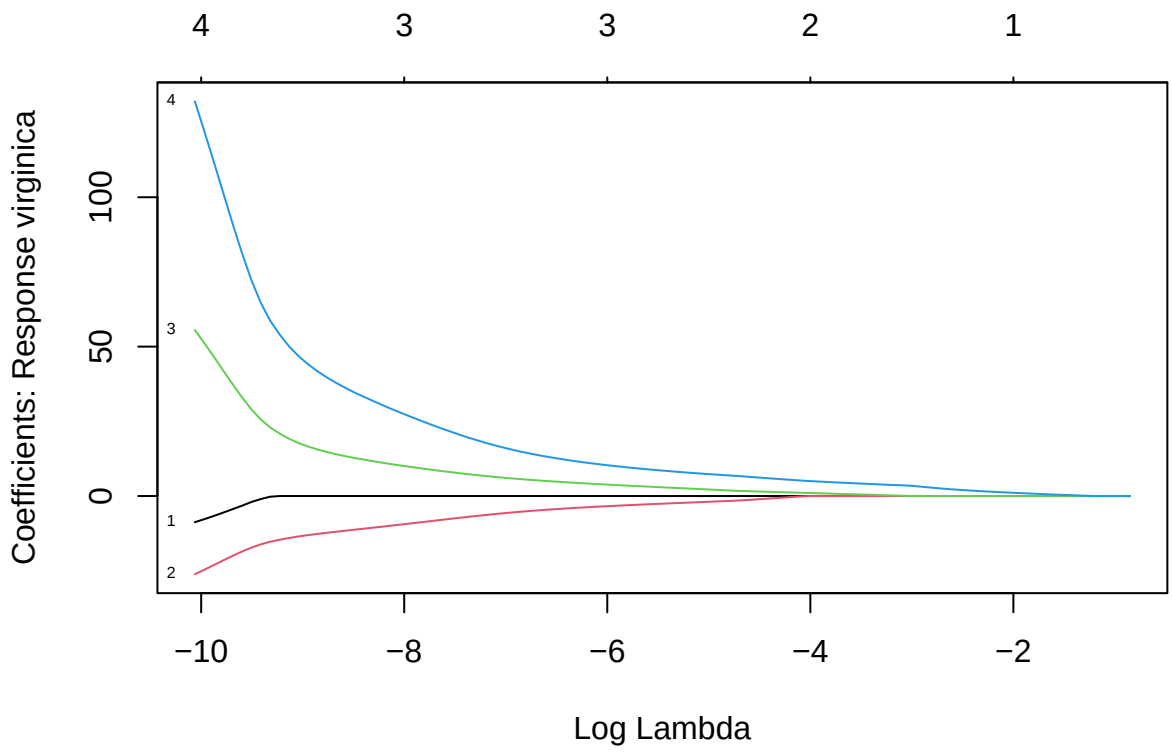
```
    x,
    y,
    family = c("gaussian", "binomial", "poisson", "multinomial", "cox", "mgaussian"),
    weights = NULL,
    offset = NULL,
    alpha = 1,
    nlambda = 100,
    lambda.min.ratio = ifelse(nobs < nvars, 0.01, 1e-04),
    lambda = NULL,
    standardize = TRUE,
    intercept = TRUE,
    thresh = 1e-07,
    dfmax = nvars + 1,
    pmax = min(dfmax * 2 + 20, nvars),
    exclude = NULL,
    penalty.factor = rep(1, nvars),
    lower.limits = -Inf,
    upper.limits = Inf,
    maxit = 1e+05,
    type.gaussian = ifelse(nvars < 500, "covariance", "naive"),
    type.logistic = c("Newton", "modified.Newton"),
    standardize.response = FALSE,
    type.multinomial = c("ungrouped", "grouped"),
    relax = FALSE,
    trace.it = 0,
    ...
  )

  relax.glmnet(fit, x, ..., maxp = n - 3, path = FALSE, check.args = TRUE)
```

```
head(iris)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

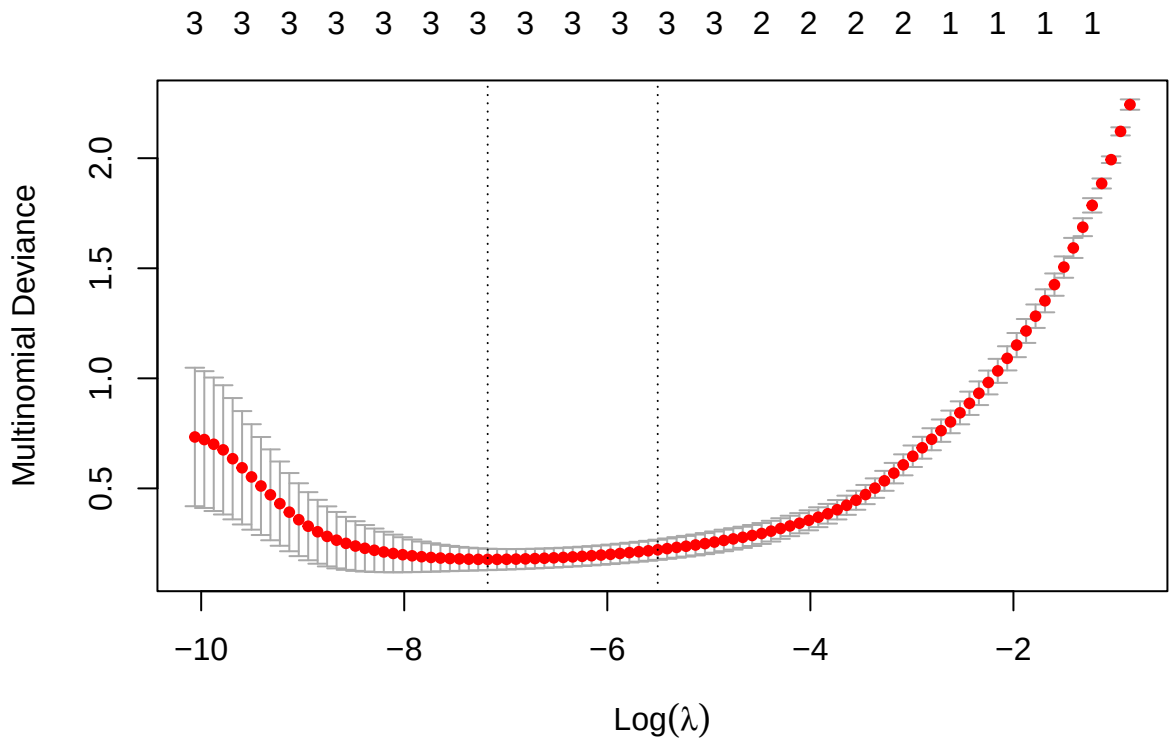
```
lasso_mod <- glmnet(as.matrix(iris[train, 1:4]), iris[train, 5], family = "multinomial",
    alpha = 1)
plot(lasso_mod, xvar = "lambda", label = TRUE)
```


3
3
3
2
1
Coefficients: Response setosa
2
3
4
8
6
4
2
0
−2
−4
−10
−8
−6
−4
−2
Log Lambda
1
1
1
0
1
Coefficients: Response versicolor
1
2
8
6
4
2
0
−10
−8
−6
−4
−2
Log Lambda



This gives a look at the path of the coefficients. Of course to fit the lasso model we need to select the lambda parameter. Here we use CV to implement this.

```
set.seed(12412)
cvfit = cv.glmnet(as.matrix(iris[train,1:4]), iris[train,5],
                  family="multinomial",alpha=1)
plot(cvfit)
```



```
cvfit$lambda.min
```

```
## [1] 0.0007630284
```

```
coef(cvfit, s = "lambda.min")
```

```
## $setosa
## 5 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept)  16.239223
## Sepal.Length  .
## Sepal.Width   3.731525
## Petal.Length -2.894364
## Petal.Width  -2.607045
##
## $versicolor
## 5 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept)  9.05159
## Sepal.Length 1.66568
## Sepal.Width  .
## Petal.Length .
## Petal.Width  .
##
## $virginica
## 5 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  -25.290813
## Sepal.Length   .
## Sepal.Width   -6.277133
## Petal.Length   6.633751
## Petal.Width   17.621958
```

```
logit_pred <- predict(cvfit, newx = as.matrix(iris[-train,1:4]), s = "lambda.min")
head(logit_pred)
```

```
## , , 1
##
##      setosa versicolor virginica
## 2  22.86028   17.21342 -31.31057
## 4  22.94400   16.71372 -31.27491
## 7  24.09219   16.71372 -32.05923
## 8  24.06345   17.37999 -33.15805
## 9  22.48713   16.38058 -30.68286
## 10 23.20470   17.21342 -33.03710
```

```
prec_class_lasso <- apply(logit_pred,1,which.max)
tbl_2 <- table(prec_class_lasso,iris$Species[-train])
prop.table(tbl_2)
```

prec_class_lasso/	setosa	versicolor	virginica
1	0.38	0.0	0.00
2	0.00	0.3	0.02
3	0.00	0.0	0.30

```r
sum(diag(prop.table(tbl_2)))
```

```
## [1] 0.98
```