uc3m | Universidad Carlos III de Madrid

OPENCOURSEWARE
APRENDIZAJE AUTOMÁTICO PARA EL ANÁLISIS DE DATOS
GRADO EN ESTADÍSTICA Y EMPRESA

Ricardo Aler

CC BY NC SA

AjusteHiperSVM

Ricardo Aler

1/8/2019

Ajuste hiper-parámetros SVMs

```
library(mlr)
```

```
## Warning: package 'mlr' was built under R version 3.5.3
## Loading required package: ParamHelpers
## Warning: package 'ParamHelpers' was built under R version 3.5.3
```

```
library(mlbench)
library(mlrHyperopt) # Esta librería ayuda en la obtención de hiper-parámetros
library(ggplot2)     # Librería para hacer plots

# Esto es para que no haya demasiados mensajes de información en el documento
configureMlr(show.info = FALSE)

data(BostonHousing)
# Primero, definimos una tarea de regresión, cuyos datos están contenidos en el data.frame BostonHousin
task_bh <- makeRegrTask(data= BostonHousing, target="medv")

# Segundo, definimos el nombre delgoritmo de aprendizaje ("learner").
learner_name <- "regr.svm"

# Tercero, vemos que hiper-parámetros ajustables tiene
filterParams(getParamSet(learner_name), tunable = TRUE)
```

```
##                    Type  len            Def
## type           discrete    - eps-regression
## kernel         discrete    -         radial
## degree          integer    -              3
## gamma           numeric    -              -
## coef0           numeric    -              0
## cost            numeric    -              1
## nu              numeric    -            0.5
## cachesize       numeric    -             40
## tolerance       numeric    -          0.001
## epsilon         numeric    -              -
## shrinking       logical    -           TRUE
## scale     logicalvector <NA>           TRUE
##                                      Constr Req Tunable Trafo
## type          eps-regression,nu-regression   -    TRUE     -
## kernel    linear,polynomial,radial,sigmoid   -    TRUE     -
## degree                            1 to Inf   Y    TRUE     -
## gamma                             0 to Inf   Y    TRUE     -
## coef0                          -Inf to Inf   Y    TRUE     -
## cost                              0 to Inf   Y    TRUE     -
## nu                             -Inf to Inf   Y    TRUE     -
```

```
## cachesize                          -Inf to Inf   -    TRUE     -
## tolerance                             0 to Inf   -    TRUE     -
## epsilon                               0 to Inf   Y    TRUE     -
## shrinking                                    -   -    TRUE     -
## scale                                        -   -    TRUE     -
```

```
# Nuestro learner inicial va a ser una SVM con kernel gausiano (radial). Lo definimos así:

learner_svm <- makeLearner(learner_name, kernel="radial")
```

Vemos que tiene muchos hiper-parámetros, pero sabemos que los más importantes son el **kernel** (linear o radial), el **cost** y **gamma** (que representa la desviación de la gausiana). Vamos a probar con una radial (gausiana), con lo que ajustaremos sólo **cost** y **gamma**

Es interesante saber que podemos conseguir ayuda sobre el método y de sus parámetros así:

```
helpLearner(learner_name)
```

```
## starting httpd help server ... done
```

```
helpLearnerParam(learner_name, "gamma")
```

```
##   *gamma*:
##      Type len Def   Constr Req Tunable Trafo
## 1 numeric   -   - 0 to Inf   Y    TRUE     -
## Used: train.
## Requires: kernel != "linear"
## Argument of: e1071::svm
##
## parameter needed for all kernels except linear (default: 1/(data
## dimension))
```

```
# Ahora habría que definir un espacio de búsqueda para los hiper-parámetros
# Una opción es ir a la página web # http://mlrhyperopt.jakob-r.de/parconfigs

# Una segunda opción es descargar directamente el espacio de búsqueda desde la web así:

pc = downloadParConfigs(learner.name="regr.svm") # http://mlrhyperopt.jakob-r.de/parconfigs
print(pc)
```

```
## list()
```

```
if(length(pc)>0) {
 ps = getParConfigParSet(pc[[1]], task=task_bh)
 print(ps)
}
# Desgraciadamente, en este caso no hay ninguno exactamente para ese learner
# Tendríamos que ir a la página web http://mlrhyperopt.jakob-r.de/parconfigs
# y buscarlo a mano, para svm.
```

En este caso, usaremos una tercera opción, más sencilla: la función **generateParConfig** nos da un espacio de búsqueda básico para un learner concreto.

```
pc_svm <- generateParConfig(learner=learner_name, task=task_bh)
ps_svm <- getParConfigParSet(pc_svm, task=task_bh)
print(ps_svm)
```

```
##          Type len  Def    Constr Req Tunable Trafo
## cost  numeric   -    0 -15 to 15   -    TRUE     Y
```

```
## gamma numeric   - -3.7 -15 to 15   -    TRUE     Y
```

Puede ser interesante ver que función se usa para el **trafo** (transformación), que en ambos casos es 2^x

```r
print(ps_svm$pars$cost$trafo)
```

```
## function (x)
## 2^x
## <environment: 0x0000000021b1dc48>
```

```r
print(ps_svm$pars$cost$trafo)
```

```
## function (x)
## 2^x
## <environment: 0x0000000021b1dc48>
```

También puede ser interesante saber cómo tendríamos que definir este espacio de búsqueda si hubiera que hacerlo a mano, o quisieramos cambiarlo. Sería así:

```r
ps_amano <- makeParamSet(
  makeNumericParam("cost", lower=-15, upper=+15, trafo = function(x) 2^x),
  makeNumericParam("gamma", lower=-15, upper=+15, trafo = function(x) 2^x)
)

print(ps_amano)
```

```
##          Type len Def    Constr Req Tunable Trafo
## cost  numeric   -   - -15 to 15   -    TRUE     Y
## gamma numeric   -   - -15 to 15   -    TRUE     Y
```

Ahora definimos cómo se búsca en el espacio de hiper-parámetros (con **randomsearch** en este caso y un **budget** de 100 evaluaciones), y cómo se evalúan los distintos hiper-parámetros (validación cruzada de tres folds).

```r
# Ojo, he puesto 1000 evaluaciones para hacer una buena visualización después, pero puede tardar mucho.
# Posiblemente con budget = 50 sea suficiente

control_grid <- makeTuneControlRandom(budget=1000)
inner_desc <- makeResampleDesc("CV", iter=3)

# Aquí construimos una secuencia de ajuste seguida de construcción de modelo
learner_tune_svm <- makeTuneWrapper(learner_svm, resampling = inner_desc, par.set = ps_svm, control = co

# También tenemos que definir cómo se va a evaluar el modelo final. En este caso con train/test (holdou
outer_desc <-  makeResampleDesc("Holdout")
set.seed(0)
outer_inst <- makeResampleInstance(outer_desc, task_bh)

# Por último, usamos resample para entrenar y evaluar learner_tune_svm
set.seed(0)
error_tune_svm <- resample(learner_tune_svm, task_bh, outer_inst, measures = list(rmse), extract = getTu
```

Vemos que los hiper-parámetros que se eligieron en la partición de entrenamiento son los siguientes (también se muestra el error que producen dichos hiper-parámetros en la validación cruzada de 3 folds que se usó para seleccionarlos):

```r
error_tune_svm$extract
```

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

```
## Tune result:
## Op. pars: cost=7.41; gamma=0.0626
## rmse.test.rmse=3.7578428
```

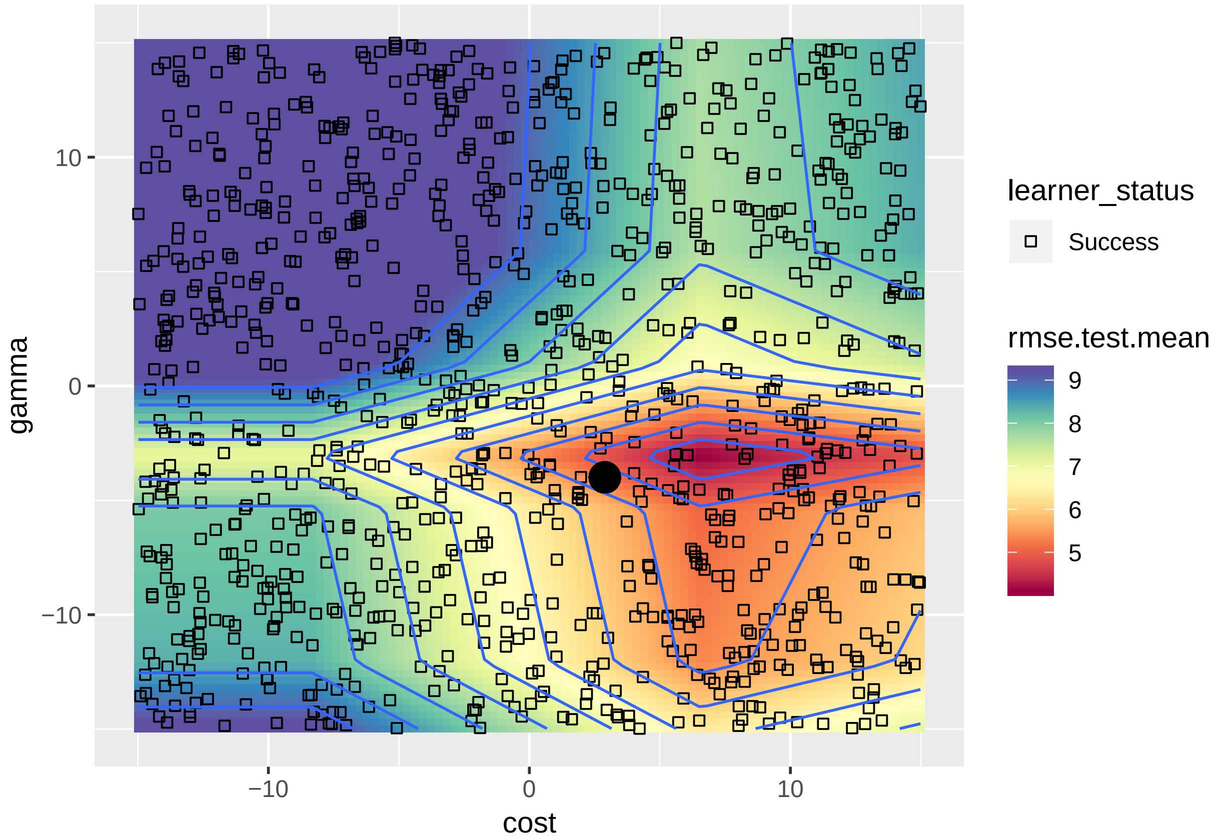
y el error del modelo final es:

```
error_tune_svm$aggr
```

```
## rmse.test.rmse
##       3.440738
```

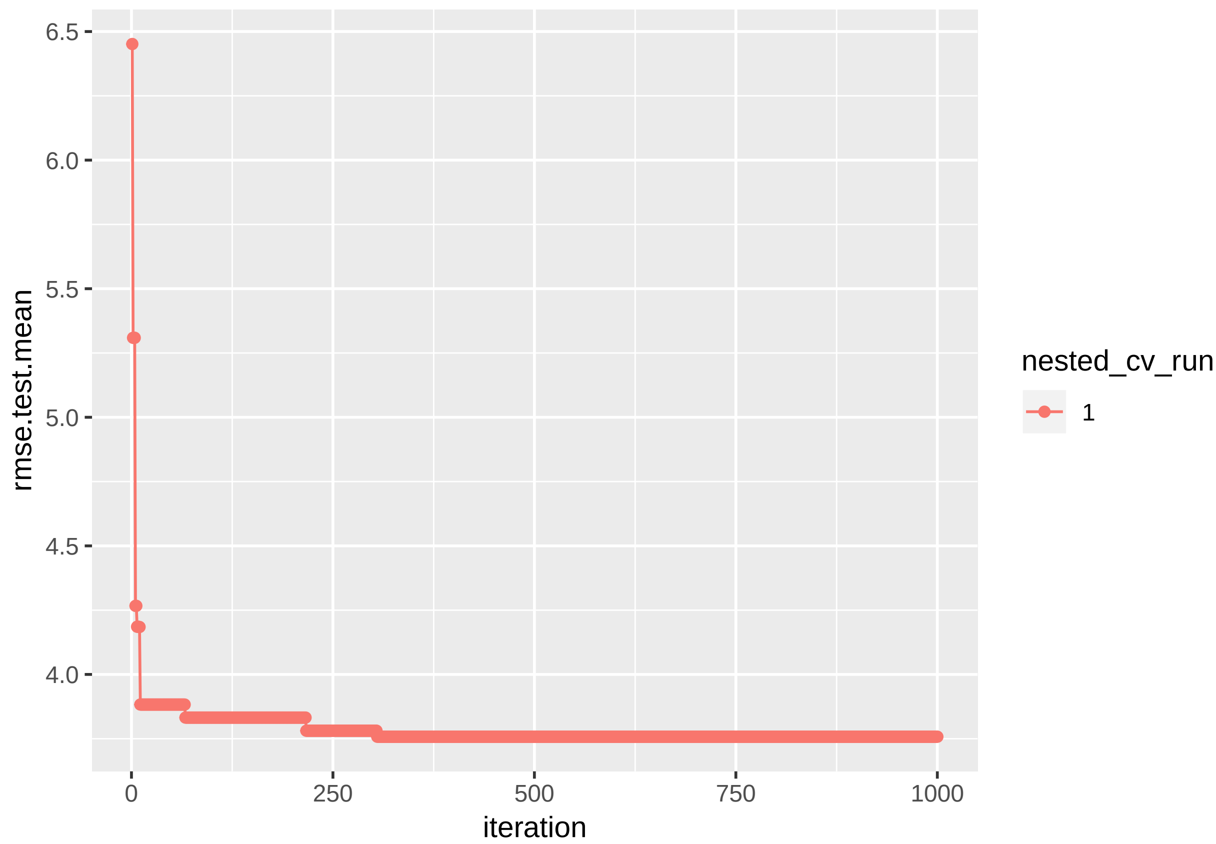
Por último, podemos ver un plot que nos muestra cómo cambia el error con los distintos valores de los hiper-parámetros:

```
data <- generateHyperParsEffectData(error_tune_svm, include.diagnostics = FALSE)

# Las siguientes dos líneas son para solventar un bug de MLR
names(data$data)[names(data$data)=="rmse.test.rmse"] <- "rmse.test.mean"
data$measures[data$measures=="rmse.test.rmse"] <- "rmse.test.mean"


plt = plotHyperParsEffect(data, x = "cost", y = "gamma", z = "rmse.test.mean",
  plot.type = "contour", interpolate = "regr.earth", show.experiments = TRUE)

plot(plt + geom_point(x=log2(error_tune_svm$extract[[1]]$x$cost), y=log2(error_tune_svm$extract[[1]]$x$g
```



A continuación podemos ver como se ha ido reduciendo el error con las iteraciones y parece que con bastantes menos que con Random Search se alcanzan mejores mínimos.

```r
plt = plotHyperParsEffect(data, x = "iteration", y = "rmse.test.mean",
  plot.type = "line")
plt
```



Ajuste con model-based optimization

Ahora definimos cómo se búsca en el espacio de hiper-parámetros con MBO

```r
library(mlrMBO)
```

```
## Loading required package: smoof

## Loading required package: BBmisc

##
## Attaching package: 'BBmisc'

## The following object is masked from 'package:base':
##
##     isFALSE

## Loading required package: checkmate
```

```r
# Dedica 80 iteraciones al ajuste
control = makeMBOControl()
control = setMBOControlTermination(control, iters = 80)
control = setMBOControlInfill(control, crit = makeMBOInfillCritEI())
```

```r
control_grid <-makeTuneControlMBO(mbo.control = control)

inner_desc <- makeResampleDesc("CV", iter=3)

# Aquí construimos una secuencia de ajuste seguida de construcción de modelo
learner_tune_svm <- makeTuneWrapper(learner_svm, resampling = inner_desc, par.set = ps_svm, control = c

# También tenemos que definir cómo se va a evaluar el modelo final. En este caso con train/test (holdou
outer_desc <-  makeResampleDesc("Holdout")
set.seed(0)
outer_inst <- makeResampleInstance(outer_desc, task_bh)

# Por último, usamos resample para entrenar y evaluar learner_tune_svm
set.seed(0)
error_tune_svm <- resample(learner_tune_svm, task_bh, outer_inst, measures = list(rmse), extract = getT
```

```
## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.
```

```
## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
```

```
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
```

```
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.

## Warning in (function (fn, nvars, max = FALSE, pop.size = 1000,
## max.generations = 100, : Stopped because hard maximum generation limit was
## hit.
```

Vemos que los hiper-parámetros que se eligieron en la partición de entrenamiento son los siguientes (también se muestra el error que producen dichos hiper-parámetros en la validación cruzada de 3 folds que se usó para seleccionarlos):

```
error_tune_svm$extract
```

```
## [[1]]
## Tune result:
## Op. pars: cost=14.4; gamma=0.0833
## rmse.test.rmse=3.7512245
```

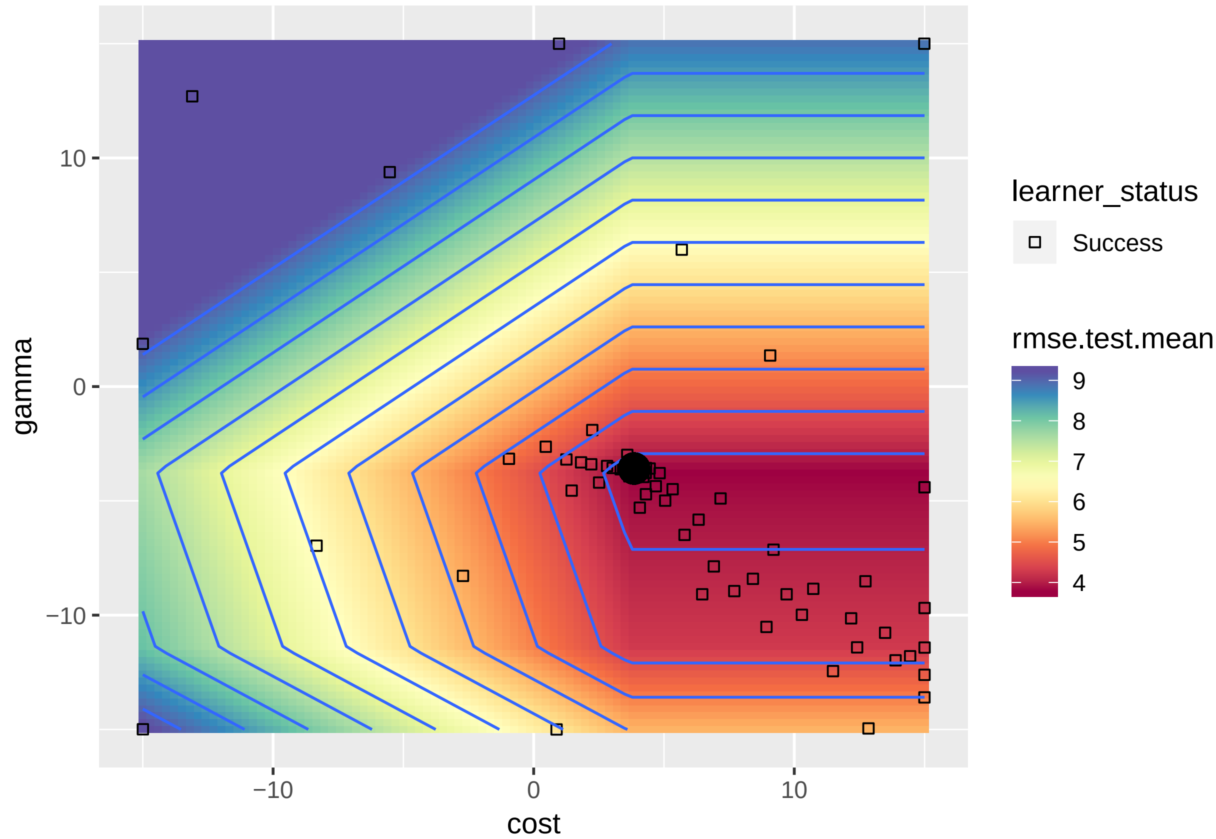
y el error del modelo final es:

```
error_tune_svm$aggr
```

```
## rmse.test.rmse
##       3.405956
```

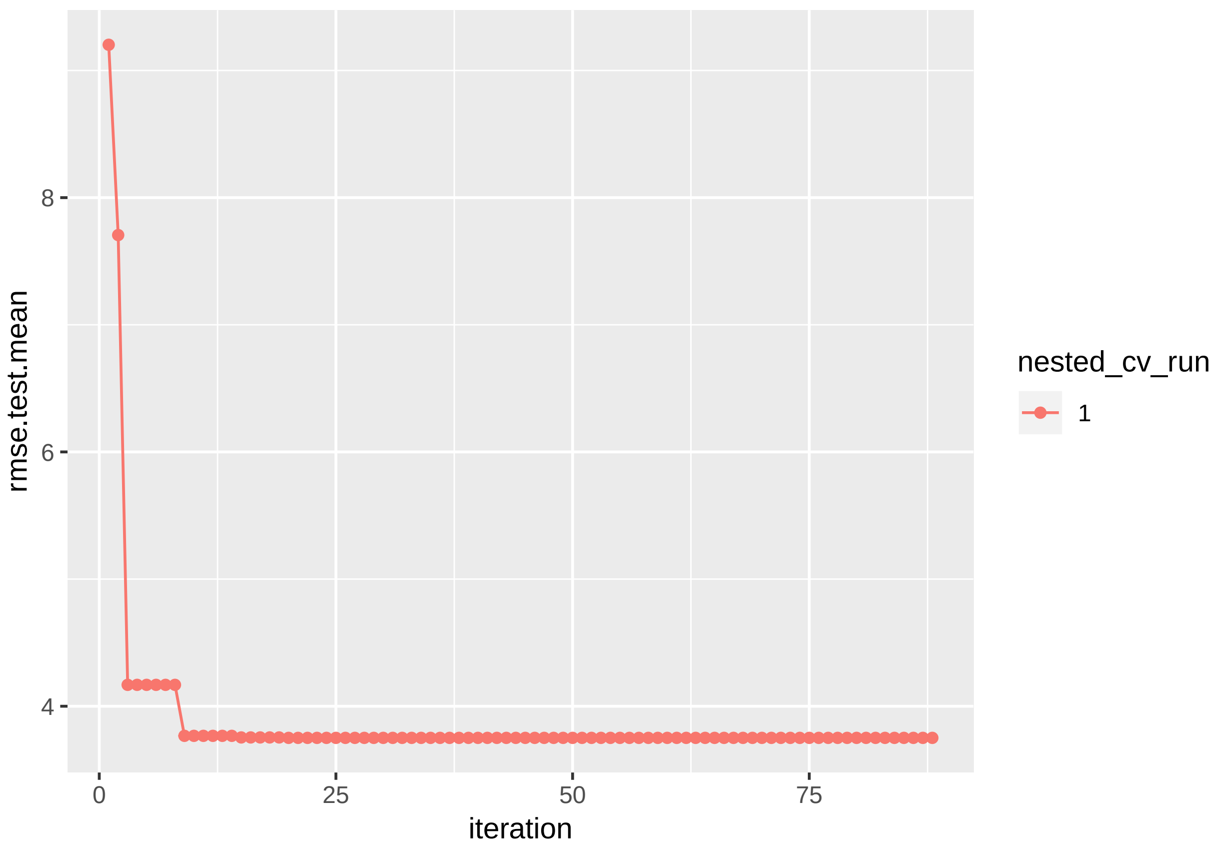
Por último, podemos ver un plot que nos muestra cómo cambia el error con los distintos valores de los hiper-parámetros. Aquí se ve como la búsqueda se concentra en una buena región (a diferencia de Random Search, que explora todas las zonas por igual).

```
data <- generateHyperParsEffectData(error_tune_svm, include.diagnostics = FALSE)

# Las siguientes dos líneas son para solventar un bug de MLR
names(data$data)[names(data$data)=="rmse.test.rmse"] <- "rmse.test.mean"
data$measures[data$measures=="rmse.test.rmse"] <- "rmse.test.mean"


plt = plotHyperParsEffect(data, x = "cost", y = "gamma", z = "rmse.test.mean",
  plot.type = "contour", interpolate = "regr.earth", show.experiments = TRUE)

plot(plt + geom_point(x=log2(error_tune_svm$extract[[1]]$x$cost), y=log2(error_tune_svm$extract[[1]]$x$g
```



A continuación podemos ver como se ha ido reduciendo el error con las iteraciones y parece que con bastantes menos que con Random Search se alcanzan mejores mínimos.

```
plt = plotHyperParsEffect(data, x = "iteration", y = "rmse.test.mean",
  plot.type = "line")
plt
```


rmse.test.mean
8
6
4
0
25
50
75
iteration
nested_cv_run
1