Package ‘discrim’

July 21, 2021

Title Model Wrappers for Discriminant Analysis

Version 0.1.3


License MIT + file LICENSE

URL https://discrim.tidymodels.org

BugReports https://github.com/tidymodels/discrim/issues

Depends parsnip (\textasciitilde 0.1.6.9000),
R (\textasciitilde 2.10)

Imports dials,
purrr,
rlang,
tibble,
utils,
withr

Suggests covr,
dplyr,
earth,
ggplot2,
klar,
MASS,
mda,
mbench,
modeldata,
naivebayes,
sda,
sparsediscrim (\textasciitilde 0.3.0),
spelling,
testthat,
xml2

Encoding UTF-8
Description

discrim_flexible() defines a model that fits a discriminant analysis model that can use nonlinear features created using multivariate adaptive regression splines (MARS).

There are different ways to fit this model. See the engine-specific pages for more details:

- **earth** (default)

More information on how **parsnip** is used for modeling is at [https://www.tidymodels.org/](https://www.tidymodels.org/).

Usage

discrim_flexible(
  mode = "classification",
  engine = "earth",
  num_terms = NULL,
  prod_degree = NULL,
  prune_method = NULL
)

Arguments

- **mode** A single character string for the type of model. The only possible value for this model is "classification".
- **engine** A single character string specifying what computational engine to use for fitting.
- **num_terms** The number of features that will be retained in the final model, including the intercept.
- **prod_degree** The highest possible interaction degree.
- **prune_method** The pruning method.
discrim_linear

Details

This function only defines what type of model is being fit. Once an engine is specified, the method to fit the model is also defined.

The model is not trained or fit until the `fit.model_spec()` function is used with the data.

References

https://www.tidymodels.org, Tidy Models with R

See Also

earth engine details

Examples

```r
parabolic_grid <-
  expand.grid(X1 = seq(-5, 5, length = 100),
              X2 = seq(-5, 5, length = 100))

fda_mod <-
  discrim_flexible(num_terms = 3) %>%
  # increase `num_terms` to find smoother boundaries
  set_engine("earth") %>%
  fit(class ~ ., data = parabolic)

parabolic_grid$fda <-
  predict(fda_mod, parabolic_grid, type = "prob$$.pred_Class1

library(ggplot2)

ggplot(parabolic, aes(x = X1, y = X2)) +
  geom_point(aes(col = class), alpha = .5) +
  geom_contour(data = parabolic_grid, aes(z = fda), col = "black", breaks = .5) +
  theme_bw() +
  theme(legend.position = "top") +
  coord_equal()
```

discrim_linear

Linear discriminant analysis

Description

discrim_linear() defines a model that estimates a multivariate distribution for the predictors separately for the data in each class (usually Gaussian with a common covariance matrix). Bayes’ theorem is used to compute the probability of each class, given the predictor values.

There are different ways to fit this model. See the engine-specific pages for more details:

- MASS (default)
- mda
- sparsediscrim

More information on how `parsnip` is used for modeling is at https://www.tidymodels.org/.
Usage

discrim_linear(
  mode = "classification",
  engine = "MASS",
  penalty = NULL,
  regularization_method = NULL
)

Arguments

mode  A single character string for the type of model. The only possible value for this model is "classification".

engine  A single character string specifying what computational engine to use for fitting.

penalty  An non-negative number representing the amount of regularization used by some of the engines.

regularization_method  A character string for the type of regularized estimation. Possible values are: "diagonal", "min_distance", "shrink_cov", and "shrink_mean" (sparsediscrim engine only).

Details

This function only defines what type of model is being fit. Once an engine is specified, the method to fit the model is also defined.

The model is not trained or fit until the fit.model_spec() function is used with the data.

References

https://www.tidymodels.org, Tidy Models with R

See Also

MASS engine details, mda engine details, sparsediscrim engine details

Examples

parabolic_grid <-
  expand.grid(X1 = seq(-5, 5, length = 100),
              X2 = seq(-5, 5, length = 100))

lda_mod <-
  discrim_linear(penalty = .1) %>%
  set_engine("mda") %>%
  fit(class ˜ ., data = parabolic)

parabolic_grid$lda <-
  predict(lda_mod, parabolic_grid, type = "prob")$.pred_Class1

library(ggplot2)

# ggplot(parabolic, aes(x = X1, y = X2)) +
ggplot(parabolic, aes(x = X1, y = X2)) +
  geom_point(aes(col = class), alpha = .5) +
  geom_contour(data = parabolic_grid, aes(z = lda), col = "black", breaks = .5) +
  theme_bw() +
**discrim_quad**  

```r
theme(legend.position = "top") +
coord_equal()
```

---

<table>
<thead>
<tr>
<th>discrim_quad</th>
<th>Quadratic discriminant analysis</th>
</tr>
</thead>
</table>

**Description**

discrim_quad() defines a model that estimates a multivariate distribution for the predictors separately for the data in each class (usually Gaussian with separate covariance matrices). Bayes' theorem is used to compute the probability of each class, given the predictor values.

There are different ways to fit this model. See the engine-specific pages for more details:

- MASS (default)
- sparsediscrim

More information on how parsnip is used for modeling is at [https://www.tidymodels.org/](https://www.tidymodels.org/).

**Usage**

discrim_quad(
  mode = "classification",
  engine = "MASS",
  regularization_method = NULL
)

**Arguments**

- **mode**
  A single character string for the type of model. The only possible value for this model is "classification".

- **engine**
  A single character string specifying what computational engine to use for fitting.

- **regularization_method**
  A character string for the type of regularized estimation. Possible values are: "diagonal", "shrink_cov", and "shrink_mean" (sparsediscrim engine only).

**Details**

This function only defines what type of model is being fit. Once an engine is specified, the method to fit the model is also defined.

The model is not trained or fit until the fit.model_spec() function is used with the data.

**References**

[https://www.tidymodels.org](https://www.tidymodels.org), Tidy Models with R

**See Also**

MASS engine details, sparsediscrim engine details
Examples

```r
parabolic_grid <-
  expand.grid(X1 = seq(-5, 5, length = 100),
              X2 = seq(-5, 5, length = 100))

qda_mod <-
  discrim_quad() %>%
  set_engine("MASS") %>%
  fit(class ~ ., data = parabolic)

parabolic_grid$qda <-
  predict(qda_mod, parabolic_grid, type = "prob")$.pred.Class1

library(ggplot2)

ggplot(parabolic, aes(x = X1, y = X2)) +
  geom_point(aes(col = class), alpha = .5) +
  geom_contour(data = parabolic_grid, aes(z = qda), col = "black", breaks = .5) +
  theme_bw() +
  theme(legend.position = "top") +
  coord_equal()
```

discrim_regularized  

Regularized discriminant analysis

discrim_regularized() defines a model that estimates a multivariate distribution for the predictors separately for the data in each class. The structure of the model can be LDA, QDA, or some amalgam of the two. Bayes’ theorem is used to compute the probability of each class, given the predictor values.

There are different ways to fit this model. See the engine-specific pages for more details:

- **klaR** (default)

More information on how `parsnip` is used for modeling is at [https://www.tidymodels.org/](https://www.tidymodels.org/).

Usage

```r
discrim_regularized(
  mode = "classification",
  engine = "klaR",
  frac_common_cov = NULL,
  frac_identity = NULL
)
```

Arguments

- **mode**: A single character string for the type of model. The only possible value for this model is "classification".
- **engine**: A single character string specifying what computational engine to use for fitting.
- **frac_common_cov, frac_identity**: Numeric values between zero and one.
Details

There are many ways of regularizing models. For example, one form of regularization is to penalize model parameters. Similarly, the classic James–Stein regularization approach shrinks the model structure to a less complex form.

The model fits a very specific type of regularized model by Friedman (1989) that uses two types of regularization. One modulates how class-specific the covariance matrix should be. This allows the model to balance between LDA and QDA. The second regularization component shrinks the covariance matrix towards the identity matrix.

For the penalization approach, `discrim_linear()` with a `mda` engine can be used. Other regularization methods can be used with `discrim_linear()` and `discrim_quad()` can used via the `sparsediscrim` engine for those functions.

This function only defines what type of model is being fit. Once an engine is specified, the `method` to fit the model is also defined.

The model is not trained or fit until the `fit.model_spec()` function is used with the data.

References

[https://www.tidymodels.org](https://www.tidymodels.org), *Tidy Models with R*


See Also

`klaR` engine details

Examples

```r
parabolic_grid <-
  expand.grid(X1 = seq(-5, 5, length = 100),
               X2 = seq(-5, 5, length = 100))

rda_mod <-
  discrim_regularized(frac_common_cov = .5, frac_identity = .5) %>%
  set_engine("klaR") %>%
  fit(class ~ ., data = parabolic)

parabolic_grid$rda <-
  predict(rda_mod, parabolic_grid, type = "prob")$.pred_Class1

library(ggplot2)

ggplot(parabolic, aes(x = X1, y = X2)) +
  geom_point(aes(col = class), alpha = .5) +
  geom_contour(data = parabolic_grid, aes(z = rda), col = "black", breaks = .5) +
  theme_bw() +
  theme(legend.position = "top") +
  coord_equal()
```
Parameter objects for Regularized Discriminant Models

Description

discrim_regularized() describes the effect of frac_common_cov() and frac_identity(). smoothness() is an alias for the adjust parameter in stats::density().

Usage

frac_common_cov(range = c(0, 1), trans = NULL)
frac_identity(range = c(0, 1), trans = NULL)
smoothness(range = c(0.5, 1.5), trans = NULL)

Arguments

range A two-element vector holding the defaults for the smallest and largest possible values, respectively.
trans A trans object from the scales package, such as scales::log10 Trans() or scales::reciprocal Trans(). If not provided, the default is used which matches the units used in range. If no transformation, NULL.

Details

These parameters can modulate a RDA model to go between linear and quadratic class boundaries.

Value

A function with classes "quant_param" and "param"

Examples

frac_common_cov()

Naive Bayes models

Description

naive_Bayes() defines a model uses Bayes’ theorem to compute the probability of each class, given the predictor values.

There are different ways to fit this model. See the engine-specific pages for more details:

- klaR (default)
- naivebayes

More information on how parsnip is used for modeling is at https://www.tidymodels.org/.
naive_Bayes

Usage

naive_Bayes(
  mode = "classification",
  engine = "klaR",
  smoothness = NULL,
  Laplace = NULL
)

Arguments

mode       A single character string for the type of model. The only possible value
            for this model is "classification".
engine     A single character string specifying what computational engine to use for
            fitting.
smoothness An non-negative number representing the relative smoothness of the
            class boundary. Smaller examples result in model flexible boundaries and
            larger values generate class boundaries that are less adaptable
Laplace    A non-negative value for the Laplace correction to smoothing low-frequency
            counts.

Details

This function only defines what type of model is being fit. Once an engine is specified, the
method to fit the model is also defined.

The model is not trained or fit until the fit.model_spec() function is used with the data.

References

https://www.tidymodels.org, Tidy Models with R

See Also

klaR engine details, naivebayes engine details

Examples

parabolic_grid <-
  expand.grid(X1 = seq(-5, 5, length = 80),
              X2 = seq(-5, 5, length = 80))

nb_mod <-
  naive_Bayes(smoothness = .8) $>
  set_engine("klaR") $>
  fit(class ~ ., data = parabolic)$

parabolic_grid$nb <-
  predict(nb_mod, parabolic_grid, type = "prob")$pred_Class1

library(ggplot2)

ggplot(parabolic, aes(x = X1, y = X2)) +
  geom_point(aes(col = class), alpha = .5) +
  geom_contour(data = parabolic_grid, aes(z = nb), col = "black", breaks = .5) +
  theme_bw() +
  theme(legend.position = "top") +


**parabolic**  
*Parabolic class boundary data*

**Description**  
Parabolic class boundary data

**Details**  
These data were simulated. There are two correlated predictors and two classes in the factor outcome.

**Value**  
parabolic  
a data frame

**Examples**  
```r  
data(parabolic)  
library(ggplot2)  
ggplot(parabolic, aes(x = X1, y = X2, col = class)) +  
  geom_point(alpha = .5) +  
  theme_bw()  
```

---

**update.discrim_flexible**  
*Update a model specification*

**Description**  
Update a model specification

**Usage**  
```r  
## S3 method for class 'discrim_flexible'  
update(  
  object,  
  num_terms = NULL,  
  prod_degree = NULL,  
  prune_method = NULL,  
  fresh = FALSE,  
  ...  
)  
```

```r  
## S3 method for class 'discrim_linear'  
update(  
  object,  
```
Arguments

**object**
A model specification.

**num_terms**
The number of features that will be retained in the final model, including the intercept.

**prod_degree**
The highest possible interaction degree.

**prune_method**
The pruning method.

**fresh**
a logical for whether the arguments should be modified in-place of or replaced wholesale.

**penalty**
An non-negative number representing the amount of regularization used by some of the engines.

**regularization_method**
A character string for the type of regularized estimation. Possible values are: "diagonal", "min_distance", "shrink_cov", and "shrink_mean" (sparsediscrim engine only).

**frac_common_cov**
Numeric values between zero and one.

**frac_identity**
Numeric values between zero and one.

**smoothness**
An non-negative number representing the the relative smoothness of the class boundary. Smaller examples result in model flexible boundaries and larger values generate class boundaries that are less adaptable.

**Laplace**
A non-negative value for the Laplace correction to smoothing low-frequency counts.