

# Package ‘pros’

December 6, 2018

**Title** Penalized Regression on Steroids

**Version** 0.1

**Author** Austin David Brown <brow5079@umn.edu>

**Maintainer** Austin David Brown <brow5079@umn.edu>

**Description** This is a project for STAT8053 at the University of Minnesota.

**Depends** R (>= 3.5.1)

**License** Licensed under the Apache-2 (<https://www.apache.org/licenses/LICENSE-2.0>) license.

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.1

## R topics documented:

cv.pros . . . . .	1
predict.cv_pros . . . . .	2
predict.pros . . . . .	3
pros . . . . .	3

<b>Index</b>	<b>5</b>
--------------	----------

---

cv.pros	<i>Cross-validation</i>
---------	-------------------------

---

## Description

The K-fold cross-validation function.

## Usage

```
cv.pros(X, y, K_fold = 5, alpha = c(1, 0, 0, 0, 0, 0),  
        lambdas = seq(10^(-7), 1, 0.1), algorithm = "proximal_gradient_cd",  
        max_iter = 1e+05, tolerance = 10^(-3))
```

**Arguments**

<code>X</code>	the matrix of the data
<code>y</code>	the vector of response values
<code>alpha</code>	the convex combination of length 7 corresponding to the penalties: <ul style="list-style-type: none"> <li>• l1 penalty</li> <li>• l2 penalty</li> <li>• l4 penalty</li> <li>• l6 penalty</li> <li>• l8 penalty</li> <li>• l10 penalty</li> </ul>
<code>lambdas</code>	A vector of dual penalization values to be evaluated
<code>algorithm</code>	the optimization algorithm <ul style="list-style-type: none"> <li>• proximal_gradient_cd</li> <li>• subgradient_cd</li> </ul>
<code>max_iter</code>	maximum iterations. This also tunes the step size.
<code>tolerance</code>	tolerance

**Value**

A class `cv_pros`

**Examples**

```
cv = cv.pros(X_train, y_train)
pred = predict(cv, X_test)
```

---

predict.cv_pros	<i>Cross-validation Prediction</i>
-----------------	------------------------------------

---

**Description**

The cross-validation prediction function.

**Usage**

```
## S3 method for class 'cv_pros'
predict(cv_prosObj, X_new)
```

**Arguments**

<code>cv_prosObj</code>	an object of class <code>cv_pros</code>
<code>X_new</code>	the matrix of the data to predict

**Value**

A vector of prediction values.

**Examples**

```
cv = cv.pros(X_train, y_train)
pred = predict(cv, X_test)
```

---

predict.pros

*Pros Prediction*


---

**Description**

The prediction function.

**Usage**

```
## S3 method for class 'pros'
predict(prosObj, X)
```

**Arguments**

prosObj	an object of class pros
X	the matrix of the data to predict

**Value**

A vector of prediction values.

**Examples**

```
fit = pros(X_train, y_train, lambda = .1)
pred = predict(fit, X_test)
```

---

pros

*Pros*


---

**Description**

The fit function for a specific lambda value.

**Usage**

```
pros(X, y, alpha = c(1, 0, 0, 0, 0, 0), lambda,
     algorithm = "proximal_gradient_cd", max_iter = 1e+05,
     tolerance = 10^(-3))
```

**Arguments**

<code>X</code>	the matrix of the data
<code>y</code>	the vector of response values
<code>alpha</code>	the convex combination of length 7 corresponding to the penalties: <ul style="list-style-type: none"><li>• l1 penalty</li><li>• l2 penalty</li><li>• l4 penalty</li><li>• l6 penalty</li><li>• l8 penalty</li><li>• l10 penalty</li></ul>
<code>lambda</code>	the dual penalization value
<code>algorithm</code>	the optimization algorithm <ul style="list-style-type: none"><li>• proximal_gradient_cd</li><li>• subgradient_cd</li></ul>
<code>max_iter</code>	maximum iterations. This also tunes the step size.
<code>tolerance</code>	tolerance

**Value**

A class `pros`

**Examples**

```
fit = pros(X_train, y_train, lambda = .1)
pred = predict(fit, X_test)
```

# Index

`cv.pros`, [1](#)

`predict.cv_pros`, [2](#)

`predict.pros`, [3](#)

`pros`, [3](#)