# Package ‘pnn’

August 29, 2016

**Title** Probabilistic neural networks

**Description** The program pnn implements the algorithm proposed by Specht (1990). It is written in the R statistical language. It solves a common problem in automatic learning. Knowing a set of observations described by a vector of quantitative variables, we classify them in a given number of groups. Then, the algorithm is trained with this datasets and should guess afterwards the group of any new observation. This neural network has the main advantage to begin generalization instantaneously even with a small set of known observations. It is delivered with four functions (learn, smooth, perf and guess) and a dataset. The functions are documented with examples and provided with unit tests.

**URL** `http://flow.chasset.net/pnn/`

**Version** 1.0.1

**Author** Pierre-Olivier Chasset

**Maintainer** Pierre-Olivier Chasset `<pierre-olivier@chasset.net>`

**License** AGPL

**Suggests** testthat, roxygen2, rgenoud

**Collate** 'create.R' 'holdout.R' 'kernel.R' 'learn.R' 'pnn-package.r' 'data-norms.R' 'guess-category.R' 'guess-probabilities.R' 'perf.r' 'guess.r' 'smooth.R' 'what-else.R'

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2013-05-07 22:17:22

## R topics documented:

---

`pnn-package` *PNN*

---

## Description

Probabilistic neural network.

## Details

The package **PNN** implements the algorithm proposed by Specht (1990). It is written in the statistical langage R. It solves a common problem in automatic learning. Knowing a set of observations described by a vector of quantitative variables, we classify them in a given number of groups. Then, the algorithm is trained with this datasets and should guess afterwards the group of any new observation. This neural network has the main advantage to begin generalization instantaneously even with a small set of known observations.

The package **PNN** exports four functions. These funtions are documented with examples and provided with unit tests:

- `learn`: Create a new Probabilist neural network with a new training set or update an existing one with new known observations.
- `smooth`: Set the smoothing parameter. If the value is not known, a genetic algorithm search the best value.
- `perf`: Compute the performance of the Probabilist neural network.
- `guess`: Guess the category of a new observation.

To help the use of **PNN**, the package contains a dataset `norms`. You could find more documentation at the package website: `http://flow.chasset.net/pnn/`.

The Probabilist neural network ist the main object used by the four functions. It is a `list` with several description fields:

- `model`: A name of the model ("Probabilistic neural network" by default).
- `set`: The raw training set.
- `category.column`: See above.
- `categories`: The categories found in the `category.column` field.
- `k`: The number of variables.
- `n`: The number of observations.
- `sigma`: The smoothing parameter.
- `observed`: A `list` of observed categories.
- `guessed`: A `list` of guessed categories.
- `success`: The number of times that the neural network chooses the right category.

- `fails`: The number of times that the neural network fails to guess the right category.
- `success_rate`: The rate of sucess over all observations in training set.
- `bic`: It is an adapted version of the Bayesian Information Criterion helping to compare different version of Probabilist neural networks.

### Author(s)

Pierre-Olivier Chasset

### References

Specht D.F. (1990). Probabilistic neural networks. Neural networks, 3(1):109-118.

### See Also

`learn`, `smooth`, `perf`, `guess`, `norms`

### Examples

```
library(pnn)
data(norms)

# The long way
pnn <- learn(norms)
pnn <- smooth(pnn, sigma=0.9)
pnn$sigma
## Not run: pnn <- perf(pnn) # Optional
## Not run: pnn$success_rate # Optional
guess(pnn, c(1,1))
guess(pnn, c(2,1))
guess(pnn, c(1.5,1))

# The short way
guess(smooth(learn(norms), sigma=0.8), c(1,1))
guess(smooth(learn(norms), sigma=0.8), c(2,1))
guess(smooth(learn(norms), sigma=0.8), c(1.5,1))

# Demonstrations
## Not run: demo("norms-trainingset", "pnn")
## Not run: demo("small-trainingset", "pnn")
```

---

## `guess` *Guess*

---

### Description

Infers the category of a new observation.

## Usage

```
guess(nn, X)
```

## Arguments

| | |
|---|---|
| `nn` | A trained and smoothed Probabilistic neural network. |
| `X` | A vector describing a new observation. |

## Details

Given an already trained and smoothed Probabilistic neural network, the function `guess` gives the category with the highest probability, and the probabilities of each category.

## Value

A `list` of the guessed category and the probabilities of each category.

## See Also

`pnn-package`, `learn`, `smooth`, `perf`, `norms`

## Examples

```
library(pnn)
data(norms)
pnn <- learn(norms)
pnn <- smooth(pnn, sigma=0.8)
guess(pnn, c(1,1))
guess(pnn, c(1,1))$category
guess(pnn, c(1,1))$probabilities
guess(pnn, c(2,1))
guess(pnn, c(1.5,1))
```

---

`learn` *Learn*

---

## Description

Create or update a Probabilist neural network.

## Usage

```
learn(set, nn, category.column = 1)
```

## Arguments

| | |
|---|---|
| `set` | Data frame representing the training set. The first column is used to define the category of each observation (set `category.column` if it is not the case). |
| `nn` | A Probabilistic neural network with or without training. |
| `category.column` | The field number of the category (1 by default). |

## Details

The function `learn` aims to create a new Probabilist neural network with a training set, or update the training set of an already trained Probabilist neural network. It sets the parameters `model`, `set`, `category.column`, `categories`, `k` and `n` of the neural network.

## Value

A trained Probabilist neural network.

## See Also

`pnn-package`, `smooth`, `perf`, `guess`, `norms`

## Examples

```
library(pnn)
data(norms)
pnn <- learn(norms)
pnn$model
pnn$set[1:10,]
pnn$category.column
pnn$categories
pnn$k
pnn$n
```

---

## `norms` *Norms*

---

## Description

Gaussian random sample dataset.

## Format

A data-frame with 400 rows and 3 columns (category `c` and coordinates `x` and `y`).

## Details

This dataset generates gaussian random points with mean equals to 1 and standard deviation equals to 0.25. Each point has a category attribute A or B. They are centered at (1,1) and (2,2) for category A; centered at (1,2) and (2,1) for category B.

## See Also

`pnn-package`, `learn`, `smooth`, `perf`, `guess`

## Examples

```
library(pnn)
data(norms)
# Just see the first observations
norms[1:10,]
```

---

`perf` *Perf*

---

## Description

Performance of a Probabilist neural network.

## Usage

```
perf(nn)
```

## Arguments

| | |
|---|---|
| `nn` | A trained and smoothed Probabilist neural network. |

## Details

The function `perf` uses a hold-out method. This method takes the training set used by the function `learn` and iterate over each observation trying to guess the current observation with a reduced training set (without the current observation).It generates:

- Two lists of `observed` and `guessed` values.
- the following statistics: number of `success` and `fails`, a sucess rate (`success_rate`) and a `bic` indicator.

## Value

A probabilist neural network updated with its performance.

## See Also

`pnn-package`, `learn`, `smooth`, `guess`, `norms`

## Examples

```
library(pnn)
data(norms)
pnn <- learn(norms)
pnn <- smooth(pnn, sigma=0.8)
pnn <- perf(pnn)
pnn$observed
pnn$guessed
pnn$success
pnn$fails
pnn$success_rate
pnn$bic
```

---

`smooth` *Smooth*

---

## Description

Work around the smoothing parameter.

## Usage

```
smooth(nn, sigma, limits = c(0, 10))
```

## Arguments

| | |
|---|---|
| `nn` | A trained Probabilist neural network. |
| `limits` | Optional. A vector giving the interval (minimum, maximum) in which the function has to search the best value. |
| `sigma` | Optional. If the value is already known, it sets directly the parameter and do not search for the best value. |

## Details

The function `smooth` aims to help to set the smoothing parameter for a Probabilist neural network. If you have no idea of which value it can be, you can let the function finds the best value using a genetic algorithm. This can be done providing to the function only the parameter `nn`. This search takes some time, so if you have already an idea of the value, you can set it if you provide both parameters `nn` and `sigma`. If you want to check visually how fit is the sigma value, you can get a plot if you provide `nn` and set `plot` to TRUE. It sets the parameters `sigma` of the neural network.

## Value

A trained and smoothed Probabilistic neural network.

## References

Walter Mebane, Jr. and Jasjeet S. Sekhon. 2011. Genetic Optimization Using Derivatives: The rgenoud package for R. Journal of Statistical Software, 42(11): 1-26.

## See Also

pnn-package, learn, perf, guess, norms

## Examples

```
library(pnn)
data(norms)

# Search the best value
pnn <- learn(norms)
## Not run: pnn <- smooth(pnn)
## Not run: pnn$sigma

# Or set the value
pnn <- smooth(pnn, sigma=0.8)
pnn$sigma
```

# Index