# Package 'engression'

May 8, 2026

**Title** Engression Modelling

**Version** 0.1.6

**Description** Fits engression models for nonlinear distributional regression. Predictors and targets can be univariate or multivariate. Functionality includes estimation of conditional mean, estimation of conditional quantiles, or sampling from the fitted distribution. Training is done full-batch on CPU (the python version offers GPU-accelerated stochastic gradient descent). Based on ``Engression: Extrapolation through the lens of distributional regression'' by Xinwei Shen and Nicolai Meinshausen (2024) in JRSSB. Also supports classification (experimental).
<doi:10.1093/jrsssb/qkae108>.

**URL** https://github.com/xwshen51/engression/

**BugReports** https://github.com/xwshen51/engression/issues

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Imports** torch

**NeedsCompilation** no

**Author** Xinwei Shen [aut],
Nicolai Meinshausen [aut, cre]

**Maintainer** Nicolai Meinshausen <meinshausen@stat.math.ethz.ch>

**Repository** CRAN

**Date/Publication** 2026-04-16 05:10:08 UTC

## Contents

`engression` *Engression Function*

## Description

This function fits an engression model to the data. It allows for the tuning of several parameters related to model complexity. Variables are per default internally standardized (predictions are on original scale).

## Usage

```
engression(
  X,
  Y,
  noise_dim = 5,
  hidden_dim = 100,
  num_layer = 3,
  dropout = 0.0,
  batch_norm = FALSE,
  num_epochs = 1000,
  lr = 10^(-4),
  beta = 1,
  silent = FALSE,
  standardize = TRUE,
  output_activation = "identity"
)
```

## Arguments

| | |
|---|---|
| `X` | A matrix or data frame representing the predictors. |
| `Y` | A matrix or vector representing the target variable(s). If Y is a factor a classification model is fitted (experimental). |
| `noise_dim` | The dimension of the noise introduced in the model (default: 5). |
| `hidden_dim` | The size of the hidden layer in the model (default: 100). |
| `num_layer` | The number of layers in the model (default: 3). |
| `dropout` | The dropout rate to be used in the model in case no batch normalization is used. Only active if batch normalization is off. (default: 0.0) |
| `batch_norm` | A boolean indicating whether to use batch-normalization (default: FALSE). |
| `num_epochs` | The number of epochs to be used in training (default: 1000). |
| `lr` | The learning rate to be used in training (default: 10^-4). |
| `beta` | The beta scaling factor for energy loss (default: 1). |
| `silent` | A boolean indicating whether to suppress output during model training (default: FALSE). |
| `standardize` | A boolean indicating whether to standardize the input data (default: TRUE). |

output_activation
: Specification of the activation function used at the output layer. This value is passed unchanged to `engressionfit()` where it is validated and interpreted. The default `"identity"` reproduces linear output. Other options are `"tanh"`, `"sigmoid"`, `"relu"`, `"softplus"`

## Value

An engression model object with class "engression".

## Examples

```
if (torch::torch_is_installed()) {
  n = 1000
  p = 5

  X = matrix(rnorm(n*p),ncol=p)
  Y = (X[,1]+rnorm(n)*0.1)^2 + (X[,2]+rnorm(n)*0.1) + rnorm(n)*0.1
  Xtest = matrix(rnorm(n*p),ncol=p)
  Ytest = (Xtest[,1]+rnorm(n)*0.1)^2 + (Xtest[,2]+rnorm(n)*0.1) + rnorm(n)*0.1

  ## fit engression object
  engr = engression(X,Y)
  print(engr)

  ## prediction on test data
  Yhat = predict(engr,Xtest,type="mean")
 cat("\n correlation between predicted and realized values: ", signif(cor(Yhat, Ytest),3))
  plot(Yhat, Ytest,xlab="prediction", ylab="observation")

  ## quantile prediction
  Yhatquant = predict(engr,Xtest,type="quantiles")
  ord = order(Yhat)
 matplot(Yhat[ord], Yhatquant[ord,], type="l", col=2,lty=1,xlab="prediction", ylab="observation")
  points(Yhat[ord],Ytest[ord],pch=20,cex=0.5)

  ## sampling from estimated model
  Ysample = predict(engr,Xtest,type="sample",nsample=1)

  ## plot of realized values against first variable
  oldpar <- par()
  par(mfrow=c(1,2))
  plot(Xtest[,1], Ytest, xlab="Variable 1", ylab="Observation")
  ## plot of sampled values against first variable
  plot(Xtest[,1], Ysample, xlab="Variable 1", ylab="Sample from engression model")
  par(oldpar)
}
```

`predict.engression` *Prediction Function for Engression Models*

## Description

This function computes predictions from a trained engression model. It allows for the generation of point estimates, quantiles, or samples from the estimated distribution.

## Usage

```
## S3 method for class 'engression'
predict(
  object,
  Xtest,
  type = c("mean", "sample", "quantile")[1],
  trim = 0.05,
  quantiles = 0.1 * (1:9),
  nsample = 200,
  drop = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| `object` | A trained engression model returned from the engression function. |
| `Xtest` | A matrix or data frame representing the predictors in the test set. |
| `type` | The type of prediction to make. "mean" for point estimates, "sample" for samples from the estimated distribution, or "quantile" for quantiles of the estimated distribution (default: "mean"). |
| `trim` | The proportion of extreme values to trim when calculating the mean (default: 0.05). |
| `quantiles` | The quantiles to estimate if type is "quantile" (default: 0.1*(1:9)). |
| `nsample` | The number of samples to draw if type is "sample" (default: 200). |
| `drop` | A boolean indicating whether to drop dimensions of length 1 from the output (default: TRUE). |
| `...` | additional arguments (currently ignored) |

## Value

A matrix or array of predictions.

## Examples

```
if (torch::torch_is_installed()) {
  n = 1000
  p = 5

  X = matrix(rnorm(n*p),ncol=p)
  Y = (X[,1]+rnorm(n)*0.1)^2 + (X[,2]+rnorm(n)*0.1) + rnorm(n)*0.1
  Xtest = matrix(rnorm(n*p),ncol=p)
  Ytest = (Xtest[,1]+rnorm(n)*0.1)^2 + (Xtest[,2]+rnorm(n)*0.1) + rnorm(n)*0.1

  ## fit engression object
  engr = engression(X,Y)
  print(engr)

  ## prediction on test data
  Yhat = predict(engr,Xtest,type="mean")
 cat("\n correlation between predicted and realized values: ", signif(cor(Yhat, Ytest),3))
  plot(Yhat, Ytest,xlab="prediction", ylab="observation")

  ## quantile prediction
  Yhatquant = predict(engr,Xtest,type="quantiles")
  ord = order(Yhat)
 matplot(Yhat[ord], Yhatquant[ord,], type="l", col=2,lty=1,xlab="prediction", ylab="observation")
  points(Yhat[ord],Ytest[ord],pch=20,cex=0.5)

  ## sampling from estimated model
  Ysample = predict(engr,Xtest,type="sample",nsample=1)
}
```

---

`print.engression` *Print an Engression Model Object*

---

## Description

This function is a utility that displays a summary of a fitted Engression model object.

## Usage

```
## S3 method for class 'engression'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A trained engression model returned from the engressionfit function. |
| `...` | additional arguments (currently ignored) |

## Value

This function does not return anything. It prints a summary of the model, including information about its architecture and training process, and the loss values achieved at several epochs during training.

## Examples

```
if (torch::torch_is_installed()) {
  n = 1000
  p = 5

  X = matrix(rnorm(n*p),ncol=p)
  Y = (X[,1]+rnorm(n)*0.1)^2 + (X[,2]+rnorm(n)*0.1) + rnorm(n)*0.1

  ## fit engression object
  engr = engression(X,Y)
  print(engr)
}
```

# Index