# Package ‘MST.PMDN’

June 13, 2025

**Type** Package

**Title** Deep Multivariate Skew t-Parsimonious Mixture Density Network

**Version** 0.1.0

**Maintainer** Alex J. Cannon <alex.cannon@ec.gc.ca>

**Author** Alex J. Cannon [aut, cre] (<https://orcid.org/0000-0002-8025-3790>)

**Description** Implements the Multivariate Skew t-Parsimonious Mixture Density Network (MST-PMDN), a distributional deep learning regression framework based on a mixture of MST distributions. Provides user-facing functions to define, train, predict, and sample from deep MST-PMDN models built with torch for R.



**Encoding** UTF-8

**Depends** R (>= 3.5.0), torch (>= 0.13.0)

**Imports** coro, stats

**LazyData** true

**LazyDataCompression** xz

**NeedsCompilation** no

## Contents

MST.PMDN-package . . . . . . . . . . 2
define_mst_pmdn . . . . . . . . . . 4
loss_mst_pmdn . . . . . . . . . . 5
predict_mst_pmdn . . . . . . . . . . 6
sample_mst_pmdn . . . . . . . . . . 6
sample_mst_pmdn_df . . . . . . . . . . 7
train_mst_pmdn . . . . . . . . . . 8
t_cdf . . . . . . . . . . 11
t_cdf_fast . . . . . . . . . . 11
t_cdf_int . . . . . . . . . . 12
t_cdf_slow . . . . . . . . . . 12
wave_surge . . . . . . . . . . 13

**Index** **14**

`MST.PMDN-package` *Deep Multivariate Skew t-Parsimonious Mixture Density Network*

## Description

The **MST.PMDN** package implements the deep Multivariate Skew t-Parsimonious Mixture Density Network (MST-PMDN), a distributional deep learning regression framework built on **torch** for R. The **MST.PMDN** framework represents complicated joint output distributions as mixtures of MST components. A volume (L)-shape (A)-orientation (D) (LAD) eigenvalue decomposition parameterization provides a tractable, interpretable, and parsimonious representation of the MST scale matrices, while explicit modeling of skewness and heavy tails can represent asymmetric behavior and tail dependence observed in real-world data (e.g., compound events and extremes). Parameters of a mixture of MST distributions that describe a multivariate output are estimated by training a deep learning model with two multi-modal input branches, one for tabular inputs and the other for (optional) image inputs. The two branches are provided as user-defined **torch** modules. Outputs from each are concatenated and passed through a dense fusion network, which then leads to the MST-PMDN head. In the absence of both branches, the tabular inputs are fed directly into the dense network.

Following the approach used in model-based clustering (**mclust**), scale matrices in the MST-PMDN head are represented using an LAD eigen-decomposition parameterization. LAD attributes, the nu (or degrees of freedom) parameter (n), and the alpha (or skewness) parameter (s) can be forced to be Variable or Equal between mixture components (plus Identity for A and D). For n and s, the model can also be constrained to emulate a multivariate Gaussian (or normal) (N) distribution. In the case of n, users can specify fixed (F) values for nu by passing an optional `fixed_nu` vector. If an element of `fixed_nu` is set to `NA`, then the value of nu for this component is learned by the network. Different model types are specified by setting the argument `constraint = "EIINN"`, `"VEVFV"`, etc. where each letter position in the argument corresponds, respectively, to each of the LADns attributes. Furthermore, values of mu (or means) (m), pi (or mixing coefficients) (x), volume-shape-orientation attributes (LAD), nu (n), and skewness (s) for the mixtures can be made to be independent of inputs by specifying any combination of `constant_attr = "m"`, `"mx"`, ..., `"LADmxns"`.

By combining appropriate values of `constraint` and `constant_attr`, MST-PMDN can emulate the Gaussian finite mixture models implemented by **mclust**, i.e., for unconditional density estimation or model-based clustering. Similarly, if the constraint on the nu parameter (n) is loosened (e.g., `constraint = "VVVEN"` with `constant_attr = "LADmxn"`), MST-PMDN can emulate model-based multivariate t clustering models provided by **teigen**. Going one step further, removing the constraint on the skewness parameter (s) (e.g., `constraint = "VVVEE"` with `constant_attr = "LADmxns"`) implements model-based MST clustering.

While it can be used for model-based density estimation and clustering tasks, the primary purpose of the **MST.PMDN** package is to implement likelihood-based deep generative models. With unconstrained or partially constrained `constant_attr`, the MST-PMDN framework allows parameters of the mixture of multivariate Gaussian, t, or skew t distributions to depend on tabular and image covariates via user-specified **torch** modules.

## Details

Key user-facing functions include:

- `define_mst_pmdn`: construct a new MST-PMDN model object.
- `train_mst_pmdn`: fit an MST-PMDN model to training data.
- `predict_mst_pmdn`: generate predictive distributions or point predictions.
- `loss_mst_pmdn`: compute the negative log-likelihood loss for evaluation.
- `sample_mst_pmdn`: draw samples as `torch_tensor` objects.
- `sample_mst_pmdn_df`: draw samples returned as an R `data.frame`.

## References

Ambrogioni, L., Güçlü, U., van Gerven, M. A., & Maris, E. (2017). The kernel mixture network: A nonparametric method for conditional density estimation of continuous random variables. arXiv:1705.07111.

Andrews, J. L., & McNicholas, P. D. (2012). Model-based clustering, classification, and discriminant analysis via mixtures of multivariate t-distributions: the t EIGEN family. Statistics and Computing, 22, 1021-1029.

Azzalini, A., & Capitanio, A. (2003). Distributions generated by perturbation of symmetry with emphasis on a multivariate skew t-distribution. Journal of the Royal Statistical Society Series B: Statistical Methodology, 65(2), 367-389.

Andrews, J. L., Wickins, J. R., Boers, N. M., & McNicholas, P. D. (2018). teigen: An R package for model-based clustering and classification via the multivariate t distribution. Journal of Statistical Software, 83, 1-32.

Banfield, J. D., & Raftery, A. E. (1993). Model-based Gaussian and non-Gaussian clustering. Biometrics, 803-821.

Celeux, G., & Govaert, G. (1995). Gaussian parsimonious clustering models. Pattern Recognition, 28(5), 781-793.

Falbel D., & Luraschi, J. (2025). torch: Tensors and Neural Networks with 'GPU' Acceleration. R package version 0.14.2, https://github.com/mlverse/torch, https://torch.mlverse.org/docs.

Fraley, C., & Raftery, A. E. (2002). Model-based clustering, discriminant analysis, and density estimation. Journal of the American Statistical Association, 97(458), 611-631.

Fraley, C., & Raftery, A. E. (1998). How many clusters? Which clustering method? Answers via model-based cluster analysis. The Computer Journal, 41(8), 578-588.

Lee, S., & McLachlan, G. J. (2014). Finite mixtures of multivariate skew t-distributions: some recent and new results. Statistics and Computing, 24, 181-202.

Kingma, D. P., & Ba, J. (2015). Adam: a method for stochastic optimization. Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA. arXiv:1412.6980

Klein, N. (2024). Distributional regression for data analysis. Annual Review of Statistics and Its Application, 11:321-346.

Peel, D., & McLachlan, G.J. (2000). Robust mixture modelling using the t distribution. Statistics and Computing 10, 339–348.

Srucca, L., Fop, M., Murphy, T. B., & Raftery, A. E. (2016). mclust 5: Clustering, classification and density estimation using Gaussian finite mixture models. The R Journal, 8(1), 289-317.

Williams, P. M. (1996). Using neural networks to model conditional multivariate densities. Neural Computation, 8(4), 843-854.

---

`define_mst_pmdn` *Construct an Untrained MST-PMDN Model*

---

## Description

Initializes a deep Multivariate Skew t-Parsimonious Mixture Density Network (MST-PMDN) as a **torch** module, ready for fitting via `train_mst_pmdn`. The model predicts parameters of a mixture of multivariate skew t distributions using optional tabular and image feature extractors.

## Usage

```
define_mst_pmdn(
  input_dim,
  output_dim,
  hidden_dim,
  n_mixtures,
  constraint      = "VVVNN",
  constant_attr   = "",
  activation      = nn_relu,
  drop_hidden     = 0,
  image_module    = NULL,
  tabular_module  = NULL,
  fixed_nu        = NULL,
  range_nu        = c(3, 50),
  max_alpha       = 5,
  min_vol_shape   = 1e-2,
  min_mix_weight  = 1e-4,
  jitter          = 1e-6
)
```

## Arguments

| | |
|---|---|
| `input_dim` | Integer. Number of features in the raw tabular input (used if `tabular_module` is `NULL`). |
| `output_dim` | Integer. Dimension `d` of the multivariate response. |
| `hidden_dim` | Integer vector. Sizes of hidden layers in the dense fusion network. |
| `n_mixtures` | Integer. Number of skew t mixture components `M`. |
| `constraint` | Character. Code for volume–shape–orientation, degrees of freedom `nu`, and skew constraints (e.g., “`VVVFN`”). |
| `constant_attr` | Character. Flags for parameters held constant (e.g., “`m`” for means, “`s`” for skew, etc.). |

| | |
|---|---|
| `activation` | Function. **torch** activation for hidden layers (default `nn_relu`). |
| `drop_hidden` | Numeric between 0 and 1. Dropout probability after each hidden layer. |
| `image_module` | A **torch** `nn_module` for image feature extraction, or `NULL` to omit. |
| `tabular_module` | A **torch** `nn_module` for tabular feature extraction, or `NULL` to use raw inputs. |
| `fixed_nu` | Numeric vector of length `M`, or `NULL`. If non-`NULL`, fixes degrees of freedom for each component. |
| `range_nu` | Numeric vector of length 2: `c(min_nu, max_nu)` to clamp learned `nu`. |
| `max_alpha` | Numeric. Absolute bound for skewness parameter. |
| `min_vol_shape` | Numeric. Minimum allowed value for volume (L) and shape (A) diagonal parameters. |
| `min_mix_weight` | Numeric. Floor for each mixture weight to avoid zero probabilities. |
| `jitter` | Numeric. Small ridge added to covariance diagonals for numerical stability. |

## Value

An untrained `mst_pmdn_model`, which is a **torch** `nn_module` encapsulating the MST-PMDN head. Use `train_mst_pmdn` to fit it to data.

---

# `loss_mst_pmdn` *Negative Log-Likelihood Loss for MST-PMDN*

---

## Description

Computes the average negative log-likelihood under a mixture of multivariate skew t distributions for a batch of examples. The function extracts mixture weights, component means, Cholesky scale factors, degrees of freedom, and skewness parameters from `output`, then evaluates the MST-PMDN log-density at `target`, finally averaging (and negating) to produce the loss. The `nu_switch` argument determines at what `nu` value the code uses the faster (but approximate) versus slower CDF routines.

## Usage

```
loss_mst_pmdn(output, target, nu_switch = 20)
```

## Arguments

| | |
|---|---|
| `output` | A list as returned by a forward pass of an `mst_pmdn_model`, containing at least the following named `torch_tensor` elements: `pi`, `mu`, `scale_chol`, `nu`, and `alpha`. |
| `target` | A `torch_tensor` or numeric matrix of true responses, with shape `[batch_size, d]`. |
| `nu_switch` | Numeric scalar. Degrees-of-freedom threshold for switching between the `t_cdf_fast` and `t_cdf_slow` implementations of the Student t CDF. |

## Value

A single-element `torch_tensor` containing the batch-averaged negative log-likelihood.

predict_mst_pmdn *Generate predictions from a trained MST-PMDN model*

## Description

Wraps a trained mst_pmdn_model in evaluation mode, converts inputs to torch_tensor if necessary, and runs a forward pass without tracking gradients to produce mixture-distribution parameters.

## Usage

```
predict_mst_pmdn(model, new_inputs, image_inputs = NULL, device = "cpu")
```

## Arguments

| | |
|---|---|
| model | An mst_pmdn_model object (a **torch** module), typically returned by train_mst_pmdn. |
| new_inputs | A numeric matrix or torch_tensor of tabular predictors. If not already a tensor, it is converted via torch_tensor on the specified device. |
| image_inputs | Optional 4D array or torch_tensor of image data for models with an image_module. If provided, must match the module's expected shape; otherwise set to NULL. |
| device | Character; the torch device to use (e.g. "cpu" or "cuda"). Defaults to "cpu". |

## Value

A named list containing the output of the model's forward pass:

pi Mixture weights tensor of shape [batch_size, M].

mu Component means tensor of shape [batch_size, M, d].

scale_chol Cholesky scale factors tensor of shape [batch_size, M, d, d].

nu Degrees-of-freedom tensor of shape [batch_size, M].

alpha Skewness parameters tensor of shape [batch_size, M, d].

L, A, D Volume, shape, and orientation decomposition tensors for each component.

sample_mst_pmdn *Draw raw tensor samples from an MST-PMDN predictive mixture*

## Description

Performs mixture sampling from the multivariate skew t predictive distribution encapsulated in mdn_output. For each of the num_samples draws, a mixture component is sampled according to the mixture weights pi, and then a skew-*t* variate is generated via a Gaussian plus Gamma construction. The output tensors are permuted so that the first dimension indexes the draw number.

### Usage

```
sample_mst_pmdn(mdn_output, num_samples = 1, device = "cpu")
```

### Arguments

| | |
|---|---|
| mdn_output | A `list` as returned by `predict_mst_pmdn` or a forward pass of an `mst_pmdn_model`, containing at minimum the following named `torch_tensor` elements: `pi`, `mu`, `scale_chol`, `nu`, and `alpha`. |
| num_samples | Integer; number of random draws to generate per input case. Defaults to `1`. |
| device | Character; a **torch** device on which to perform sampling (e.g. `"cpu"` or `"cuda"`). Defaults to `"cpu"`. |

### Value

A `list` with components:

- `samples` A `torch_tensor` of shape `c(num_samples, batch_size, d)`, giving the sampled responses.
- `components` A `torch_tensor` of shape `c(num_samples, batch_size)`, containing the index (1–`M`) of the mixture component used for each draw.

---

## sample_mst_pmdn_df    *Draw MST-PMDN predictive samples as a data frame*

---

### Description

For each of `num_samples` draws per case, a mixture component is sampled according to the weights `pi`, and then a multivariate skew-*t* variate is generated via the usual Gaussian + Gamma construction. The results are returned in long format as an R `data.frame` with one row per draw, including the sampled values for each response dimension, the input case index, the draw index, and the component chosen.

### Usage

```
sample_mst_pmdn_df(mdn_output, num_samples = 1, device = "cpu")
```

### Arguments

| | |
|---|---|
| mdn_output | A `list` as returned by `predict_mst_pmdn` or a forward pass of an `mst_pmdn_model`, containing at minimum the following named `torch_tensor` elements: `pi`, `mu`, `scale_chol`, `nu`, and `alpha`. |
| num_samples | Integer; number of random draws to generate per input case. Defaults to `1`. |
| device | Character; the torch device on which sampling is performed (e.g.\ `"cpu"` or `"cuda"`). Defaults to `"cpu"`. |

## Value

A data.frame with num_samples * B rows (where B = number of input cases) and d + 3 columns:

V1,...,Vd Numeric; sampled values for each of the *d* response dimensions.

row Integer; index of the input case (1 through B).

draw Integer; draw number (1 through num_samples).

comp Factor; mixture component index (1 through M) used for that draw.

---

train_mst_pmdn *Train a Deep Multivariate Skew t Parsimonious Mixture Density Network*

---

## Description

Fits an MST-PMDN model by minimizing the negative log-likelihood on the supplied data, with support for optional image inputs, checkpointing, early stopping, gradient clipping, weight decay, dropout, and learning-rate scheduling.

## Usage

```
train_mst_pmdn(
  inputs,
  outputs,
  hidden_dim,
  n_mixtures,
  constraint             = "VVVNN",
  constant_attr          = "",
  fixed_nu               = NULL,
  range_nu               = c(3, 50),
  nu_switch              = 20,
  max_alpha              = 5,
  min_vol_shape          = 1e-2,
  min_mix_weight         = 1e-4,
  jitter                 = 1e-6,
  activation             = nn_tanh,
  epochs                 = 500,
  lr                     = 0.001,
  batch_size             = 16,
  max_norm               = 1,
  drop_hidden            = 0,
  wd_image               = 0,
  wd_tabular             = 0,
  checkpoint_interval    = 10,
  checkpoint_path        = "checkpoint.pt",
  resume_from_checkpoint = FALSE,
  model                  = NULL,
```

```
  early_stopping_patience= 50,
  validation_split       = 0.2,
  custom_split           = NULL,
  scheduler_step         = 50,
  scheduler_gamma        = 0.5,
  image_inputs           = NULL,
  image_module           = NULL,
  tabular_module         = NULL,
  device                 = "cpu"
)
```

## Arguments

| | |
|---|---|
| `inputs` | Numeric matrix or `torch_tensor` of predictors (rows = cases). |
| `outputs` | Numeric matrix or `torch_tensor` of responses (rows = cases). |
| `hidden_dim` | Integer vector giving the sizes of the hidden layers in the fusion network. |
| `n_mixtures` | Integer; number of mixture components `M`. |
| `constraint` | Character; MST-PMDN constraint code (volume–shape–orientation, degrees of freedon `nu`, and skew), e.g. `"VVVFN"`. |
| `constant_attr` | Character; flags for parameters held constant ("m" for means, "x" for mixing coefficients, "L" for volume, "A" for shape, "D" for orientation, "n" for degrees of freedom `nu`, and "s" for skew). |
| `fixed_nu` | Numeric vector of length `M`, or `NULL`; if non-`NULL`, fixes degrees of freedom `nu` per component. |
| `range_nu` | Numeric vector of length 2: clamp range for `nu`. |
| `nu_switch` | Numeric; `nu` threshold for switching between fast vs slow CDF routines. |
| `max_alpha` | Numeric; maximum absolute skewness parameter. |
| `min_vol_shape` | Numeric; minimum allowed volume (L) and shape (A) diagonal values. |
| `min_mix_weight` | Numeric; minimum mixture weight per component. |
| `jitter` | Numeric; small ridge added to covariance diagonals for stability. |
| `activation` | Activation function for hidden layers (e.g., `nn_tanh`, `nn_relu`). |
| `epochs` | Integer; maximum number of training epochs. |
| `lr` | Numeric; learning rate for the optimizer. |
| `batch_size` | Integer; mini-batch size. |
| `max_norm` | Numeric; maximum gradient norm for clipping. |
| `drop_hidden` | Numeric in [0,1]; dropout probability after each hidden layer. |
| `wd_image` | Numeric; weight decay (L2 penalty) for `image_module` parameters. |
| `wd_tabular` | Numeric; weight decay for `tabular_module` parameters. |
| `checkpoint_interval` | Integer; number of epochs between saved checkpoints. |
| `checkpoint_path` | Character; file path for saving model checkpoints. |

resume_from_checkpoint
: Logical; if `TRUE`, loads and resumes from an existing checkpoint.

model
: An existing `mst_pmdn_model` to continue training, or `NULL` to initialize a new one.

early_stopping_patience
: Integer; epochs with no validation improvement before stopping.

validation_split
: Numeric in [0,1]; fraction of cases held out for validation if `custom_split` is `NULL`.

custom_split
: Index or logical vector, or list with `train` and `validation` elements; overrides `validation_split`.

scheduler_step
: Integer; epochs between learning-rate decay steps.

scheduler_gamma
: Numeric; multiplicative factor for learning-rate decay.

image_inputs
: Optional array or `torch_tensor` of image data (4D) for multimodal models.

image_module
: Optional `nn_module` for processing `image_inputs`.

tabular_module
: Optional `nn_module` for processing `inputs`.

device
: Character; torch device to use (e.g., `"cpu"` or `"cuda"`).

## Details

Splits the data into training/validation sets (unless `custom_split` is provided), initializes or loads the model, and then runs an Adam optimization loop with optional gradient clipping, weight decay, dropout, early stopping, checkpointing, and learning-rate scheduling. After training, the best-performing model (by validation or training loss) is reloaded before returning.

## Value

A `list` with components:

**model** The trained `mst_pmdn_model` (torch module).

**train_loss_history** Numeric vector of training losses per epoch.

**val_loss_history** Numeric vector of validation losses per epoch, or `NULL` if none.

**best_train_epoch** Epoch number with lowest training loss (if no validation split).

**best_train_loss** Lowest training loss achieved (if no validation split).

**best_val_epoch** Epoch number with lowest validation loss (if used).

**best_val_loss** Lowest validation loss achieved (if used).

**final_epoch** Last completed epoch.

**train_indices** Indices of cases used for training.

**val_indices** Indices of cases used for validation.

## t_cdf *Unified interface to Student t CDF*

### Description

Conditionally dispatches to t_cdf_fast or t_cdf_slow, providing a single API for computing the Student t cumulative distribution in a manner compatible with torch computation graphs. Note: will be deprecated once torch.distributions.studentT has been ported to torch for R.

### Usage

```
t_cdf(z, nu, nu_switch = 20)
```

### Arguments

| | |
|---|---|
| z | Numeric vector or torch_tensor of quantiles. |
| nu | Degrees of freedom (scalar or tensor). |
| nu_switch | Value of nu below/above which t_cdf_slow/t_cdf_fast are used. |

### Value

A torch_tensor of CDF values.

## t_cdf_fast *Fast approximation of the Student t CDF*

### Description

Computes a differentiable "fast" approximation to the cumulative distribution function of the Student t distribution, suitable for use in **torch** computation graphs.

### Usage

```
t_cdf_fast(z, nu)
```

### Arguments

| | |
|---|---|
| z | Numeric vector or torch_tensor of quantiles. |
| nu | Degrees of freedom (scalar or tensor). |

### Value

A torch_tensor of the same shape as z, containing CDF values.

## t_cdf_int *Numerical integration–based Student t CDF*

### Description

Computes the Student t cumulative distribution function by numerically integrating the probability density function.

### Usage

```
t_cdf_int(t_val, nu, num_integration_points = 1000L)
```

### Arguments

| | |
|---|---|
| `t_val` | Numeric vector or `torch_tensor` of quantiles at which to evaluate the CDF. |
| `nu` | Degrees of freedom (scalar or tensor). |
| `num_integration_points` | Integer; number of points to use in the numerical integration. Defaults to `1000L`. |

### Value

A `torch_tensor` of CDF values corresponding to `t_val`.

## t_cdf_slow *Accurate but slower Student t CDF approximation*

### Description

Computes a more accurate but computationally slower approximation to the Student t CDF using `pt` and finite differences for the gradient.

### Usage

```
t_cdf_slow(z, nu)
```

### Arguments

| | |
|---|---|
| `z` | Numeric vector or `torch_tensor` of quantiles. |
| `nu` | Degrees of freedom (scalar or tensor). |

### Value

A `torch_tensor` of CDF values corresponding to `z`.

`wave_surge` *Wave–Surge Daily-Maximum Subset*

## Description

A subset of the CCCRIS node 181947 daily maximum wave and surge data and covariates.

## Usage

```
data(wave_surge)
```

## Format

A named `list` with three components:

**x** Numeric array of tabular predictors (harmonics of the seasonal cyle and lag-1 wave/surge data).

**x_image** Numeric array of image predictors (sea level pressure and sea level pressure gradient).

**y** Numeric matrix of scaled responses (daily maximum significant wave height and surge).

## Source

CCCRIS node 181947 (cccris.ca) and ERA5 reanalysis

# Index


∗ **datasets**
  wave_surge, 13
∗ **package**
  MST.PMDN-package, 2

data.frame, *3*, *7, 8*
define_mst_pmdn, *3*, 4

loss_mst_pmdn, *3*, 5

MST.PMDN-package, 2

predict_mst_pmdn, *3*, 6, *7*

sample_mst_pmdn, *3*, 6
sample_mst_pmdn_df, *3*, 7

t_cdf, 11
t_cdf_fast, *5*, *11*, 11
t_cdf_int, 12
t_cdf_slow, *5*, *11*, 12
torch_tensor, *3*, *5–7*, *9*, *11, 12*
train_mst_pmdn, *3–6*, 8

wave_surge, 13