Package 'tamd'

June 4, 2026

Type Package

Title Transcendental Algorithm for Mixtures of Distributions

Version 1.0.2

Date 2026-05-31

Maintainer Ernest Fokoue <epfeqa@rit.edu>

Description Implements the Transcendental Algorithm for Mixtures of Distributions (TAMD), a penalized likelihood framework for fitting finite Gaussian mixture models. TAMD augments the Expectation-Maximization (EM) algorithm with analytic barrier terms built from the Hellinger affinity that diverge on the singular locus, actively preventing component coalescence and weight degeneracy. Provides the core TAMD fitting function, closed-form Hellinger affinity and gradient computations, the Transcendental Affinity Criterion (TAC) for geometry-aware model selection, the regularity index rho (a scalar diagnostic for mixture fit quality), and reproduction scripts for all simulation studies. Methods are described in Fokoue (2024) <doi:10.48550/arXiv.2602.03889>.
See also Titterington, Smith and Makov (1985, ISBN:0-471-90510-4) and Watanabe (2009, ISBN:978-0-521-86408-7).

License GPL-3

Encoding UTF-8

Depends R (>= 4.0.0)

Imports stats, utils, MASS, mvtnorm

Suggests testthat (>= 3.0.0)

NeedsCompilation no

Author Ernest Fokoue [aut, cre]

Repository CRAN

Date/Publication 2026-06-04 11:50:02 UTC

Contents

`compute_aic` *AIC for Gaussian Mixture Models*

Description

Standard Akaike Information Criterion. Provided for comparison with `compute_tac`.

Usage

```
compute_aic(ll, K, d)
```

Arguments

`ll`	Numeric. Normalized log-likelihood (mean over n).
`K`	Integer. Number of components.
`d`	Integer. Dimension.

Value

Numeric scalar. AIC value (lower is better).

See Also

`compute_tac`, `compute_bic`, `tamd_select`

Examples

```
set.seed(1)
X <- simulate_gmm(200, K = 2,
  pi    = c(0.5, 0.5),
  mu    = matrix(c(-3, 0, 3, 0), nrow = 2),
  Sigma = array(rep(diag(2), 2), dim = c(2, 2, 2)))
fit <- tamd(X, K = 2, seed = 1)
cat("AIC =", round(fit$aic, 2), "\n")
cat("TAC =", round(fit$tac, 2), "\n")
```

`compute_bic` *BIC for Gaussian Mixture Models*

Description

Standard Bayesian Information Criterion using the nominal parameter count d_theta. Known to overcount parameters for singular mixture models (Keribin 2000, Watanabe 2009). Provided for comparison with `compute_tac`.

Usage

```
compute_bic(ll, K, d, n)
```

Arguments

`ll`	Numeric. Normalized log-likelihood (mean over n).
`K`	Integer. Number of components.
`d`	Integer. Dimension.
`n`	Integer. Sample size.

Value

Numeric scalar. BIC value (lower is better).

See Also

`compute_tac`, `tamd_select`

Examples

```
set.seed(1)
X <- simulate_gmm(200, K = 2,
  pi    = c(0.5, 0.5),
  mu    = matrix(c(-3, 0, 3, 0), nrow = 2),
  Sigma = array(rep(diag(2), 2), dim = c(2, 2, 2)))
fit <- tamd(X, K = 2, seed = 1)
cat("BIC =", round(fit$bic, 2), "\n")
cat("TAC =", round(fit$tac, 2), "\n")
cat("TAC < BIC:", fit$tac < fit$bic, "\n")
```

`compute_tac` *Transcendental Affinity Criterion (TAC)*

Description

Computes the TAC for a fitted Gaussian mixture. TAC uses a geometry-corrected effective parameter count that depends on the pairwise Hellinger affinities, giving more accurate model selection than BIC for singular mixture models (Theorem E).

Usage

```
compute_tac(X, theta, K, d, A, lambda_n, lambda_wt, lambda_sc)
```

Arguments

`X`	Numeric n x d data matrix.
`theta`	List with pi, mu, Sigma (from a tamd fit).
`K`	Integer. Number of components.
`d`	Integer. Data dimension.
`A`	Numeric K x K Hellinger affinity matrix.
`lambda_n`	Numeric. Penalty strength.
`lambda_wt`	Numeric. Weight barrier coefficient.
`lambda_sc`	Numeric. Scale barrier coefficient.

Details

The TAC formula is:

$$TAC(K) = -2 \sum \log p(X_i) + 2\lambda_n R_T(\hat{\theta}) + p_{eff}(\hat{\theta}) \log n$$

where the effective parameter count is:

$$p_{eff} = d_\theta - 2 \sum_{k<j} \frac{A_{kj}}{1 - A_{kj}} \cdot \frac{d_{pair}}{2}$$

For well-separated components (A_kj near 0), p_eff = d_theta and TAC reduces to BIC. For near-coalesced components, p_eff < d_theta, giving a smaller, more accurate penalty.

Value

Numeric scalar. TAC value (lower is better). Automatically available as `fit$tac` from `tamd`.

References

Fokoue, E. (2024). Theorem E and Definition TAC.

See Also

tamd, tamd_select, compute_bic, regularity_index

Examples

```
set.seed(42)
X <- simulate_gmm(300, K = 2,
  pi    = c(0.5, 0.5),
  mu    = matrix(c(-3, 0, 3, 0), nrow = 2),
  Sigma = array(rep(diag(2), 2), dim = c(2, 2, 2)))
fit <- tamd(X, K = 2, seed = 42)
cat("TAC (from fit):", round(fit$tac, 2), "\n")
cat("BIC (from fit):", round(fit$bic, 2), "\n")
cat("TAC < BIC:", fit$tac < fit$bic, "\n")
```

---

compute_tic *Transcendental Information Criterion (TIC)*

Description

Computes TIC, the general-family counterpart to TAC. Uses the barrier-corrected RLCT as the complexity penalty. Automatically available as `fit$tic` from tamd.

Usage

```
compute_tic(X, theta, K, d, lambda_n, lambda_wt, lambda_sc)
```

Arguments

X	Numeric n x d data matrix.
theta	List with pi, mu, Sigma.
K	Integer. Number of components.
d	Integer. Dimension.
lambda_n	Numeric. Penalty strength.
lambda_wt	Numeric. Weight barrier coefficient.
lambda_sc	Numeric. Scale barrier coefficient.

Value

Numeric scalar. TIC value (lower is better).

See Also

compute_tac, tamd

Examples

```
set.seed(1)
X <- simulate_gmm(200, K = 2,
  pi    = c(0.5, 0.5),
  mu    = matrix(c(-2, 0, 2, 0), nrow = 2),
  Sigma = array(rep(diag(2), 2), dim = c(2, 2, 2)))
fit <- tamd(X, K = 2, seed = 1)
cat("TIC =", round(fit$tic, 2), "\n")
cat("TAC =", round(fit$tac, 2), "\n")
```

`hellinger_affinity` *Hellinger Affinity Between Two Gaussian Distributions*

Description

Computes the Hellinger affinity between two multivariate Gaussian distributions in closed form, using Cholesky decomposition for numerical stability. A(eta_i, eta_j) = 1 when distributions are identical (coalescence); A(eta_i, eta_j) near 0 when distributions are well-separated.

Usage

```
hellinger_affinity(mu_i, Sigma_i, mu_j, Sigma_j)
```

Arguments

`mu_i`	Numeric vector of length d. Mean of component i.
`Sigma_i`	Numeric d x d positive-definite matrix. Covariance of component i.
`mu_j`	Numeric vector of length d. Mean of component j.
`Sigma_j`	Numeric d x d positive-definite matrix. Covariance of component j.

Details

The closed-form expression for Gaussian distributions is:

$$\mathsf{A}(\eta_i, \eta_j) = \frac{|\Sigma_i|^{1/4}|\Sigma_j|^{1/4}}{|M_{ij}|^{1/2}} \exp\left\{-\frac{1}{8}\delta_{ij}^{\top} M_{ij}^{-1}\delta_{ij}\right\}$$

where M_ij = (Sigma_i + Sigma_j)/2 and delta_ij = mu_i - mu_j. Computation is in log-space for stability.

Value

Numeric scalar in [0, 1].

See Also

`hellinger_affinity_matrix`, `separation_gradient_mean`

Examples

```
## Two well-separated 2D Gaussians: affinity near 0
A1 <- hellinger_affinity(
  mu_i = c(0, 0), Sigma_i = diag(2),
  mu_j = c(4, 4), Sigma_j = diag(2))
cat("Well-separated affinity:", round(A1, 6), "\n")

## Two nearly identical Gaussians: affinity near 1
A2 <- hellinger_affinity(
  mu_i = c(0, 0), Sigma_i = diag(2),
  mu_j = c(0.01, 0), Sigma_j = diag(2))
cat("Near-coalesced affinity:", round(A2, 6), "\n")

## Identical: affinity = 1
A3 <- hellinger_affinity(
  mu_i = c(1, 2), Sigma_i = diag(2),
  mu_j = c(1, 2), Sigma_j = diag(2))
cat("Identical affinity:", round(A3, 6), "\n")
```

---

hellinger_affinity_matrix

Pairwise Hellinger Affinity Matrix

Description

Computes the symmetric K x K matrix of pairwise Hellinger affinities for all component pairs in a Gaussian mixture.

Usage

```
hellinger_affinity_matrix(mu, Sigma)
```

Arguments

`mu`	Numeric d x K matrix. Column k is mean of component k.
`Sigma`	Numeric d x d x K array. Sigma[,,k] is covariance of component k.

Value

Symmetric K x K matrix with ones on diagonal and A_kj in [0,1) off diagonal.

See Also

`hellinger_affinity`

Examples

```
K <- 3; d <- 2
mu    <- matrix(c(-3, 0,  0, 0,  3, 0), nrow = d)
Sigma <- array(rep(diag(d), K), dim = c(d, d, K))
A     <- hellinger_affinity_matrix(mu, Sigma)
print(round(A, 4))
cat("Min separation Delta =",
    round(min(1 - A[upper.tri(A)]), 4), "\n")
```

`label_match` *Label-Matched Parameter Comparison*

Description

Finds the permutation of fitted component labels that minimizes MSE between fitted and true means. Handles the label-switching invariance of mixture models. Used in all paper simulation studies (Surgery S3, Fokoue 2024).

Usage

```
label_match(mu_hat, mu_true)
```

Arguments

`mu_hat`	Numeric d x K matrix of fitted component means.
`mu_true`	Numeric d x K matrix of true component means.

Value

A list with elements:

mse Label-matched MSE between fitted and true means.

perm Best permutation as integer vector of length K.

mu_matched Permuted fitted means aligned to truth.

See Also

`tamd`, `simulate_gmm`

Examples

```
## True means
mu_true <- matrix(c(-3, 0,  0, 0,  3, 0), nrow = 2)

## Fitted means in wrong order (components 3, 1, 2)
mu_hat  <- mu_true[, c(3, 1, 2)]

res <- label_match(mu_hat, mu_true)
cat("MSE after label matching:", round(res$mse, 10), "\n")
cat("Best permutation:", res$perm, "\n")
```

plot.tamd *Plot a tamd Object*

Description

Produces diagnostic plots for a fitted TAMD model. Shows convergence history and, for d <= 2, a scatter plot with fitted component ellipses labeled by regularity index rho.

Usage

```
## S3 method for class 'tamd'
plot(x, X = NULL,
     which = c("both", "history", "fit"), ...)
```

Arguments

x	A tamd object returned by tamd.
X	Optional numeric n x d data matrix. Required for the "fit" panel. For d=1 shows a density histogram with fitted mixture curve. For d=2 shows a scatter plot with 95% probability ellipses per component.
which	Character. "both" (default): convergence history and fitted mixture side by side. "history": convergence plot only. "fit": fitted mixture only (requires X).
...	Ignored.

Value

Invisibly returns x.

See Also

tamd, print.tamd

Examples

```
## Bivariate mixture: scatter + ellipses
set.seed(42)
X <- simulate_gmm(300, K = 3,
  pi    = rep(1/3, 3),
  mu    = matrix(c(-4, 0,  0, 0,  4, 0), nrow = 2),
  Sigma = array(rep(diag(2), 3), dim = c(2, 2, 3)))
fit <- tamd(X, K = 3, seed = 42)
plot(fit, X = X, which = "both")

## Univariate mixture: density histogram

set.seed(7)
X1 <- simulate_gmm(500, K = 2,
  pi    = c(0.5, 0.5),
```

```
  mu    = matrix(c(-3, 3), nrow = 1),
  Sigma = array(rep(matrix(1), 2), dim = c(1, 1, 2)))
fit1 <- tamd(X1, K = 2, lambda_n = 0.05, seed = 7)
plot(fit1, X = X1, which = "fit")
```

`print.tamd` *Print a tamd Object*

Description

Concise summary of a fitted TAMD model: dimensions, criteria, regularity index, and component weights.

Usage

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

Arguments

`x`	A tamd object returned by `tamd`.
`...`	Ignored.

Value

Invisibly returns `x`.

See Also

`tamd`, `summary.tamd`, `plot.tamd`

Examples

```
set.seed(1)
X <- simulate_gmm(200, K = 2,
  pi    = c(0.5, 0.5),
  mu    = matrix(c(-3, 0, 3, 0), nrow = 2),
  Sigma = array(rep(diag(2), 2), dim = c(2, 2, 2)))
fit <- tamd(X, K = 2, seed = 1)
print(fit)
```

`regularity_index` *Regularity Index (Titterington Theorem)*

Description

Computes the regularity index rho in (0,1) from the Titterington Theorem (Fokoue 2024, Part vi). This scalar summarizes how close the fitted mixture is to a fully regular statistical model.

rho -> 1 as n -> infinity (with K = K_0, lambda_n -> 0). Computed free of charge from TAMD fit output.

Usage

```
regularity_index(A, d, K)
```

Arguments

`A`	Numeric K x K Hellinger affinity matrix.
`d`	Integer. Dimension of the data.
`K`	Integer. Number of components.

Details

The effective complexity (Titterington Theorem Part iii) is:

$$\mathfrak{C}(\hat{\theta}_n) = d_\theta(K) - 2\sum_{k<j} \frac{\hat{\mathsf{A}}_{kj}}{1-\hat{\mathsf{A}}_{kj}} \cdot \frac{d + d(d+1)/2}{2}$$

The regularity index is:

$$\rho(\hat{\theta}_n) = \frac{\mathfrak{C}(\hat{\theta}_n) - 2\lambda_{unpen}(K)}{d_\theta(K) - 2\lambda_{unpen}(K)}$$

Practical guidelines:

- rho > 0.90: Reliable. Trust parameter estimates and clustering.
- 0.70 < rho <= 0.90: Moderate. Verify K is appropriate.
- rho <= 0.70: Caution. Near-singular. Density estimation only. Reduce K or increase n.

Value

Numeric scalar in (0, 1).

References

Fokoue, E. (2024). The Transcendental Algorithm for Mixtures of Distributions. The Titterington Theorem, Part (vi).

See Also

tamd, rlct_gaussian

Examples

```
## Well-separated: rho near 1
A_good <- matrix(c(1, 0.02, 0.02, 1), 2, 2)
rho_good <- regularity_index(A_good, d = 2, K = 2)
cat("Well-separated rho =", round(rho_good, 4), "\n")

## Near-coalesced: rho near 0
A_bad <- matrix(c(1, 0.85, 0.85, 1), 2, 2)
rho_bad <- regularity_index(A_bad, d = 2, K = 2)
cat("Near-coalesced rho =", round(rho_bad, 4), "\n")

## From a tamd fit
set.seed(1)
X <- simulate_gmm(200, K = 2,
  pi    = c(.5, .5),
  mu    = matrix(c(-3, 0, 3, 0), 2),
  Sigma = array(rep(diag(2), 2), c(2, 2, 2)))
fit <- tamd(X, K = 2, seed = 1)
cat("fit$rho =", round(fit$rho, 4), "\n")
```

---

rlct_gaussian *Real Log-Canonical Threshold for Gaussian Mixtures*

Description

Returns the real log-canonical threshold (RLCT) for an unpenalized K-component Gaussian mixture. The RLCT governs the asymptotic generalization error of singular models. TAMD raises this value strictly (Theorem D, Fokoue 2024).

Usage

```
rlct_gaussian(K, d)
```

Arguments

K	Integer. Number of fitted components.
d	Integer. Dimension.

Value

Numeric. RLCT value (always less than d_theta(K)/2). For K=2, d=1: exact value 3/4.

References

Watanabe, S. (2009). Algebraic Geometry and Statistical Learning Theory. Cambridge University Press.

Yamazaki, K. and Watanabe, S. (2003). Singularities in mixture models. Neural Networks, 16(7), 1029–1038.

Examples

```
cat("K=2, d=1 RLCT =", rlct_gaussian(2, 1), "\n")  # 3/4
cat("d_theta/2 =", ((2-1) + 2*1 + 1*2/2)/2, "\n")  # 3/2
cat("RLCT < d_theta/2:", rlct_gaussian(2,1) < 1.5, "\n")
```

`separation_gradient_mean` *Separation Barrier Gradients*

Description

Computes the gradient of the separation barrier with respect to component means and covariances. These are the transcendental correction terms added to EM in the TAMD M-step.

From Theorem B(iv) (Fokoue 2024), the mean gradient is a Mahalanobis-weighted repulsive force:

$$\nabla_{\mu_k} \mathcal{B}_{sep} = \sum_{j \neq k} \frac{\mathsf{A}_{kj}}{1 - \mathsf{A}_{kj}} \cdot \frac{1}{4} M_{kj}^{-1} (\mu_k - \mu_j)$$

This diverges as A_kj -> 1 (components coalesce), providing infinite repulsive force exactly when needed.

Usage

```
separation_gradient_mean(mu, Sigma, A)
separation_gradient_cov(mu, Sigma, A, lambda_sc = 0)
```

Arguments

`mu`	Numeric d x K matrix of component means.
`Sigma`	Numeric d x d x K array of covariances.
`A`	Numeric K x K Hellinger affinity matrix (from `hellinger_affinity_matrix`).
`lambda_sc`	Numeric. Scale barrier coefficient. Default 0.

Value

`separation_gradient_mean`: Numeric d x K matrix. Column k is the gradient of the separation barrier w.r.t. mu_k.

`separation_gradient_cov`: Numeric d x d x K array. Slice [,,k] is the gradient w.r.t. Sigma_k.

References

Fokoue, E. (2024). The Transcendental Algorithm for Mixtures of Distributions. Theorem B(iv) and Lemma B.1.

See Also

hellinger_affinity_matrix, tamd

Examples

```
## Near-coalesced components: large gradient (Theorem B)
d <- 2; K <- 2
mu    <- matrix(c(-0.1, 0, 0.1, 0), nrow = d)
Sigma <- array(rep(diag(d), K), dim = c(d, d, K))
A     <- hellinger_affinity_matrix(mu, Sigma)
G     <- separation_gradient_mean(mu, Sigma, A)
cat("Separation gradient norm:", round(norm(G, "F"), 4), "\n")
cat("(Large = strong repulsion, as predicted by Theorem B)\n")

## Well-separated: small gradient
mu2   <- matrix(c(-4, 0, 4, 0), nrow = d)
A2    <- hellinger_affinity_matrix(mu2, Sigma)
G2    <- separation_gradient_mean(mu2, Sigma, A2)
cat("Well-separated gradient norm:", round(norm(G2, "F"), 6), "\n")
```

---

simulate_gmm *Simulate Data from a Gaussian Mixture Model*

Description

Generates n i.i.d. observations from a K-component Gaussian mixture with specified parameters. Used in all paper simulation studies (Fokoue 2024).

Usage

```
simulate_gmm(n, K, pi, mu, Sigma, seed = 42L)
```

Arguments

n	Integer. Number of observations.
K	Integer. Number of components.
pi	Numeric vector of length K. Mixing weights (automatically normalized to sum to 1).
mu	Numeric d x K matrix. Column k is mean of component k.
Sigma	Numeric d x d x K array. Sigma[,,k] is the covariance of component k.
seed	Integer. Random seed. Default 42.

Value

Numeric n x d matrix of observations.

Examples

```
## Univariate 2-component mixture
X1 <- simulate_gmm(
  n     = 300,
  K     = 2,
  pi    = c(0.5, 0.5),
  mu    = matrix(c(-2, 2), nrow = 1),
  Sigma = array(rep(matrix(1), 2), dim = c(1, 1, 2)),
  seed  = 1
)
hist(X1, breaks = 30, main = "Simulated 2-component mixture")

## Bivariate 3-component mixture
X2 <- simulate_gmm(
  n     = 500,
  K     = 3,
  pi    = c(0.3, 0.4, 0.3),
  mu    = matrix(c(-4, 0,  0, 0,  4, 0), nrow = 2),
  Sigma = array(rep(diag(2), 3), dim = c(2, 2, 3)),
  seed  = 42
)
plot(X2, pch = 16, cex = 0.5, col = "gray50",
     main = "Simulated 3-component mixture")
```

`summary.tamd` *Summarize a tamd Object*

Description

Detailed report of a fitted TAMD model including all component parameters and pairwise Hellinger affinities.

Usage

```
## S3 method for class 'tamd'
summary(object, ...)
```

Arguments

`object`	A tamd object returned by `tamd`.
`...`	Ignored.

Value

Invisibly returns `object`.

See Also

tamd, print.tamd, plot.tamd

Examples

```
set.seed(1)
X <- simulate_gmm(200, K = 2,
  pi    = c(0.5, 0.5),
  mu    = matrix(c(-3, 0, 3, 0), nrow = 2),
  Sigma = array(rep(diag(2), 2), dim = c(2, 2, 2)))
fit <- tamd(X, K = 2, seed = 1)
summary(fit)
```

---

tamd *Fit a Gaussian Mixture Model via TAMD*

Description

The Transcendental Algorithm for Mixtures of Distributions (TAMD) fits a K-component Gaussian mixture model via penalized likelihood maximization. Analytic barrier terms built from the Hellinger affinity between components prevent coalescence and weight degeneracy.

This work is dedicated to the memory of Professor D.M. Titterington (1945–2023), University of Glasgow.

Usage

```
tamd(X, K,
     lambda_n   = NULL,
     lambda_wt  = 0.01,
     lambda_sc  = 0.0,
     max_iter   = 200,
     tol        = 1e-6,
     init       = c("kmeans", "random", "user"),
     theta_init = NULL,
     n_restarts = 1L,
     seed       = 42L,
     verbose    = FALSE)
```

Arguments

X	Numeric matrix of shape n x d. Rows are observations.
K	Integer. Number of mixture components.
lambda_n	Numeric. Penalty strength. Default: sqrt(log(n)/n) as prescribed by Theorem C.
lambda_wt	Numeric >= 0. Weight barrier coefficient. Default 0.01.
lambda_sc	Numeric >= 0. Scale barrier coefficient. Default 0.

`max_iter`	Integer. Maximum EM iterations. Default 200.
`tol`	Numeric. Convergence tolerance on objective improvement. Default 1e-6.
`init`	Character. Initialization method: `"kmeans"` (default), `"random"`, or `"user"`.
`theta_init`	List. User-supplied initial parameters when `init = "user"`. Must contain `pi` (length-K), `mu` (d x K matrix), `Sigma` (d x d x K array).
`n_restarts`	Integer. Number of random restarts. The run with the highest objective is returned. Default 1.
`seed`	Integer. Random seed for reproducibility. Default 42.
`verbose`	Logical. Print iteration progress. Default FALSE.

Details

TAMD maximizes the penalized objective:

$$\hat{\theta}_n = \arg\max_{\theta} \frac{1}{n} \sum_{i=1}^{n} \log p_\theta(X_i) - \lambda_n \mathcal{R}_T(\theta)$$

where the transcendental regularizer is

$$\mathcal{R}_T(\theta) = \sum_{k<j} -\log(1 - \mathsf{A}_{kj}) - \lambda_{wt} \sum_{k} \log \pi_k$$

and A_{kj} is the Hellinger affinity between components k and j. The barrier diverges precisely when components coalesce (Theorem A) and its gradient is transverse to the singular locus with repulsion law $\|\nabla \mathcal{R}_T\| \geq C_0 / d(\theta, \mathcal{S}_K)$ (Theorem B). The estimator is confined to a compact regular stratum with a positive-definite penalized Hessian (Theorem C).

The penalty schedule lambda_n -> 0 ensures asymptotic efficiency: as n grows, TAMD recovers the Cramer-Rao bound.

Use `tamd_select` to automatically choose K via the TAC criterion (Theorem E).

Value

An object of class `"tamd"`, a list containing:

`theta` Fitted parameters: list with `pi` (mixing weights), `mu` (d x K mean matrix), `Sigma` (d x d x K covariance array).

`loglik` Final normalized log-likelihood.

`objective` Final TAMD penalized objective value.

`affinities` K x K matrix of pairwise Hellinger affinities at convergence.

`rho` Regularity index rho in (0,1). Values above 0.90 indicate reliable, well-separated components. See `regularity_index` and the Titterington Theorem.

`tac` TAC criterion value (lower is better). Use `tamd_select` for automated K selection.

`tic` TIC criterion value (lower is better).

`bic` BIC value for comparison with TAC.

`aic` AIC value for comparison.

`history` Numeric vector of objective values per iteration. Useful for convergence diagnostics.

`converged` Logical. TRUE if convergence criterion was met within max_iter iterations.

`n_iter` Number of iterations run.

`K`, `n`, `d` Problem dimensions.

`lambda_n` Penalty strength actually used.

References

Fokoue, E. (2024). The Transcendental Algorithm for Mixtures of Distributions. *Annals of Statistics*, submitted.

Titterington, D.M., Smith, A.F.M., and Makov, U.E. (1985). *Statistical Analysis of Finite Mixture Distributions*. Wiley.

Watanabe, S. (2009). *Algebraic Geometry and Statistical Learning Theory*. Cambridge University Press.

See Also

`tamd_select`, `regularity_index`, `compute_tac`, `simulate_gmm`, `plot.tamd`, `print.tamd`, `hellinger_affinity`, `separation_gradient_mean`

Examples

```
## ------------------------------------------------------------
## Example 1: Basic fit --- two well-separated Gaussians
## ------------------------------------------------------------
set.seed(42)
X <- simulate_gmm(
  n     = 300,
  K     = 2,
  pi    = c(0.4, 0.6),
  mu    = matrix(c(-3, 0, 3, 0), nrow = 2),
  Sigma = array(rep(diag(2), 2), dim = c(2, 2, 2))
)
fit <- tamd(X, K = 2, seed = 42)
print(fit)

## ------------------------------------------------------------
## Example 2: Inspect the regularity index and criteria
## ------------------------------------------------------------
cat("Regularity index rho =", round(fit$rho, 4), "\n")
cat("TAC =", round(fit$tac, 2), "\n")
cat("BIC =", round(fit$bic, 2), "\n")
cat("TAC < BIC:", fit$tac < fit$bic, "\n")

## ------------------------------------------------------------
## Example 3: Automated model selection via TAC
## ------------------------------------------------------------
sel <- tamd_select(X, K_min = 1, K_max = 5, seed = 42)
cat("TAC selects K =", sel$K_hat, "\n")
print(sel$table)
```

```
## ------------------------------------------------------------
## Example 4: Three-component mixture in 2D
## ------------------------------------------------------------
set.seed(7)
X3 <- simulate_gmm(
  n     = 500,
  K     = 3,
  pi    = c(0.3, 0.4, 0.3),
  mu    = matrix(c(-4, 0,  0, 0,  4, 0), nrow = 2),
  Sigma = array(rep(diag(2), 3), dim = c(2, 2, 3))
)
fit3 <- tamd(X3, K = 3, seed = 7)
plot(fit3, X = X3)

## ------------------------------------------------------------
## Example 5: Univariate mixture (well-separated for reliability)
## ------------------------------------------------------------

set.seed(99)
X1 <- simulate_gmm(
  n     = 500,
  K     = 2,
  pi    = c(0.5, 0.5),
  mu    = matrix(c(-3, 3), nrow = 1),
  Sigma = array(rep(matrix(1), 2), dim = c(1, 1, 2))
)
fit1 <- tamd(X1, K = 2, lambda_n = 0.05, seed = 99)
plot(fit1, X = X1, which = "fit")


## ------------------------------------------------------------
## Example 6: Real data --- Old Faithful
## ------------------------------------------------------------
data("faithful")
X_f <- scale(faithful)
sel_f <- tamd_select(X_f, K_min = 1, K_max = 5, seed = 42)
cat("Old Faithful: TAC selects K =", sel_f$K_hat, "\n")
plot(sel_f$fit, X = X_f, which = "fit")
```

`tamd_select` *Automated Model Order Selection via TAC*

Description

Fits TAMD for K = K_min, ..., K_max and selects the number of components minimizing the Transcendental Affinity Criterion (TAC). TAC is consistent for model order selection and dominates BIC (Theorem E, Fokoue 2024).

Usage

```
tamd_select(X, K_min = 1L, K_max = 6L, ...)
```

Arguments

X	Numeric n x d data matrix.
K_min	Integer. Minimum components to try. Default 1.
K_max	Integer. Maximum components to try. Default 6.
...	Additional arguments passed to tamd. Commonly: seed, lambda_n, n_restarts.

Value

A list with:

K_hat Selected number of components (TAC minimum).

fit The tamd object at K_hat.

table Data frame with columns K, TAC, BIC, AIC, rho, loglik for each fitted model.

fits List of all tamd fits (K_min to K_max).

References

Fokoue, E. (2024). The Transcendental Algorithm for Mixtures of Distributions. *Annals of Statistics*, submitted. Theorem E: TIC and TAC consistency.

See Also

tamd, compute_tac, regularity_index

Examples

```
## Simulate 3-component mixture
set.seed(42)
X <- simulate_gmm(
  n     = 400,
  K     = 3,
  pi    = c(0.3, 0.4, 0.3),
  mu    = matrix(c(-4, 0,  0, 0,  4, 0), nrow = 2),
  Sigma = array(rep(diag(2), 3), dim = c(2, 2, 3))
)

## Automated selection
sel <- tamd_select(X, K_min = 1, K_max = 6, seed = 42)
cat("TAC selects K =", sel$K_hat, "\n")

## Full table: TAC vs BIC vs rho
print(sel$table)

## Plot the selected fit
plot(sel$fit, X = X, which = "fit")
```

```
## The selected fit has rho diagnostic
cat("Regularity index rho =", round(sel$fit$rho, 4), "\n")
```

Index