Package 'AntsNet'

May 6, 2026

Title Unified Simulation of Isomorphisms Between Ant Colony Intelligence and Machine Learning

Version 1.0.0

Description Implements the full suite of simulation, visualization, and analysis tools for exploring the mathematical isomorphisms between ant colony decision-making and three major paradigms of machine learning: random forests (Part I: variance reduction through decorrelation), boosting (Part II: bias reduction through adaptive recruitment), and neural networks (Part III: gradient-based generational learning). Accompanies the trilogy ``Isomorphic Functionalities between Ant Colony and Ensemble Learning'' (Fokoué, Babbitt, and Levental, 2026, <doi:10.48550/arXiv.2603.20328>, <doi:10.48550/arXiv.2604.00038>).

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/ylevental/IsomorphismSim_Full

BugReports https://github.com/ylevental/IsomorphismSim_Full/issues

Depends R (>= 4.1.0)

Imports stats, graphics, grDevices, ranger, ggplot2, dplyr, tidyr, patchwork, viridis, ggpubr

Suggests shiny, cowplot, knitr, rmarkdown, testthat (>= 3.0.0)

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

Author Yuval Levental [aut, cre], Gregory Babbitt [aut], Ernest Fokoué [aut]

Maintainer Yuval Levental <yhl3051@rit.edu>

Repository CRAN

Date/Publication 2026-04-21 19:02:41 UTC

Contents

`AntsNet-package` *AntsNet: Unified Simulation of Ant Colony / Machine Learning Isomorphisms*

Description

Implements the full suite of simulation, visualization, and analysis tools for exploring the mathematical isomorphisms between ant colony intelligence and three major paradigms of machine learning: random forests (Part I), boosting (Part II), and neural networks (Part III).

Part I — Random Forest Isomorphism

`generate_regression_data` Synthetic regression data

`simulate_ant_colony` Agent-based colony simulation

`within_colony_correlation` Pairwise ant correlation

`variance_decomposition_experiment` RF variance decomposition

`colony_variance_experiment` Colony variance decomposition

`isomorphism_test` Direct isomorphism comparison

`optimal_decorrelation_experiment` Optimal theta sweep

`sensitivity_analysis` Robustness across conditions

`create_isomorphism_schematic` Figure 1: schematic

`plot_variance_decomposition` Figure 2: variance

`plot_correlation_decay` Figure 3: correlation decay

`plot_optimal_decorrelation` Figure 4: optimal theta

`plot_sensitivity_heatmap` Figure 5: sensitivity

Part II — Boosting Isomorphism

`generate_classification_data` Synthetic classification data

`adaboost` AdaBoost with decision stumps

`predict_adaboost` Predict from AdaBoost ensemble

`acar` Ant Colony Adaptive Recruitment

`calculate_margins` Boosting margin computation

`calculate_quorum_margin` Colony quorum margin

`weak_learnability_experiment` Weak learnability test

`convergence_experiment_boost` Convergence comparison

`noise_experiment_boost` Noise robustness comparison

`plot_weak_learnability` Figure 1: weak learnability

`plot_weight_pheromone` Figure 2: weight/pheromone

`plot_margin_quorum` Figure 3: margin/quorum

`plot_convergence_boost` Figure 4: convergence

`plot_noise_robustness_boost` Figure 5: noise

Part III — Neural Network Isomorphism

`generate_synthetic_data` Synthetic data with complexity

`gacl` Generational Ant Colony Learning

`simple_neural_network` Single-layer neural network

`plot_isomorphism` Figure 1: gradient isomorphism

`plot_learning_curves` Figure 2: learning curves

`plot_pheromone_weight` Figure 3: pheromone/weight

`plot_learning_rate_sensitivity` Figure 4: LR sensitivity

`plot_noise_robustness_nn` Figure 5: noise robustness

`plot_convergence_complexity` Figure 6: complexity

`plot_gradient_dynamics` Figure 7: gradient dynamics

`plot_plasticity` Figure 8: plasticity/adaptation

Shiny Apps

`launch_app` Interactive explorer (Part I or Part II)

Author(s)

Maintainer: Yuval Levental <yhl3051@rit.edu>

Authors:

- Gregory Babbitt <gabsbi@rit.edu>
- Ernest Fokoué <epfeqa@rit.edu>

See Also

Useful links:

- https://github.com/ylevental/IsomorphismSim_Full
- Report bugs at https://github.com/ylevental/IsomorphismSim_Full/issues

acar *Ant Colony Adaptive Recruitment (ACAR)*

Description

Simulates the adaptive recruitment dynamics of an ant colony, implementing Algorithm 3 (ACAR) from the Part II manuscript.

Usage

```
acar(
  site_qualities,
  n_ants = 50,
  n_waves = 100,
  rho = 0.1,
  gamma = 0.5,
  alpha = 1,
  beta = 1,
  noise_sd = 0.5,
  early_stop = TRUE
)
```

Arguments

`site_qualities`	Numeric vector of true site qualities.
`n_ants`	Number of ants released per wave.
`n_waves`	Maximum number of recruitment waves.
`rho`	Evaporation rate in $(0, 1]$.
`gamma`	Pheromone deposition rate.
`alpha`	Pheromone influence exponent.
`beta`	Heuristic influence exponent.
`noise_sd`	Standard deviation of observation noise.
`early_stop`	Logical; stop early when consensus is reached.

Value

A list with:

pheromone_history Matrix (waves x K) of pheromone concentrations.

fitness_history Vector of colony fitness per wave.

decision_history Integer vector of colony decisions per wave.

final_decision Index of the chosen site.

adaboost *AdaBoost with Decision Stumps*

Description

Implements the AdaBoost algorithm using decision stumps as weak learners, following Freund & Schapire (1997).

Usage

```
adaboost(X, y, M = 100)
```

Arguments

X	Numeric matrix of predictors (n x p).
y	Numeric vector of labels in {-1, +1}.
M	Integer number of boosting iterations.

Value

A list with components `models` (list of stumps), `alpha` (learner weights), and `weight_history` (M x n matrix of instance weights at each iteration).

calculate_margins *Calculate Boosting Margins*

Description

Computes the normalized margin of each training instance.

Usage

```
calculate_margins(result, X, y)
```

Arguments

result	Output of `adaboost()`.
X	Predictor matrix.
y	Label vector.

Value

Numeric vector of margins.

`calculate_quorum_margin` *Calculate Quorum Margin*

Description

Computes the normalized pheromone margin between the best and second-best sites, analogous to boosting margins.

Usage

```
calculate_quorum_margin(result)
```

Arguments

`result`	Output of `acar()`.

Value

Numeric scalar: the quorum margin.

`colony_variance_experiment` *Experiment 2: Ant Colony Variance Decomposition*

Description

Sweeps colony size and exploration probability, computing decision accuracy and the empirical variance/correlation of ant assessments.

Usage

```
colony_variance_experiment(
  n_ants_values = c(10, 20, 30, 50, 100),
  p_explore_values = seq(0, 1, by = 0.2),
  n_replicates = 50,
  n_sites = 5,
  site_qualities = c(10, 8, 6, 4, 2)
)
```

Arguments

`n_ants_values`	Integer vector of colony sizes
`p_explore_values`	Numeric vector of exploration probabilities
`n_replicates`	Monte Carlo replicates
`n_sites`	Number of candidate nest sites
`site_qualities`	True site quality vector

Value

A data.frame with columns `n_ants`, `p_explore`, `accuracy_mean`, `accuracy_sd`, `var_mean`, `cor_mean`

`convergence_experiment_boost` *Convergence Rate Experiment*

Description

Compares convergence of AdaBoost and ACAR across iterations. For AdaBoost: test-set accuracy using first m iterations. For ACAR: P(correct final decision) using m waves, averaged across independent replicates.

Usage

```
convergence_experiment_boost(
  n_boost_reps = 30,
  n_acar_reps = 200,
  max_iters = 80,
  verbose = TRUE
)
```

Arguments

`n_boost_reps`	Number of Monte Carlo replicates for AdaBoost.
`n_acar_reps`	Number of Monte Carlo replicates for ACAR.
`max_iters`	Maximum iterations / waves.
`verbose`	Logical; if `TRUE` (the default), progress messages are emitted via `message` and can be suppressed with `suppressMessages`.

Value

A data.frame with columns `iteration`, `accuracy`, `system`, and `rep`.

`create_isomorphism_schematic` *Figure 1: Isomorphism Schematic*

Description

Plots theoretical $\rho = \rho_{max}(1 - \theta)$ alongside empirical values from both random forest and ant colony experiments.

Usage

```
create_isomorphism_schematic(rf_results, colony_results)
```

Arguments

`rf_results`	Output of `variance_decomposition_experiment`
`colony_results`	Output of `colony_variance_experiment`

Value

A ggplot object

`find_best_stump` *Find the Best Decision Stump*

Description

Searches over all features and thresholds to find the stump that minimizes the weighted classification error.

Usage

```
find_best_stump(X, y, D)
```

Arguments

`X`	Numeric matrix of predictors.
`y`	Label vector in {-1, +1}.
`D`	Weight vector (sums to 1).

Value

A list describing the best stump: `feature`, `threshold`, and `direction`.

`gacl` *Generational Ant Colony Learning (GACL)*

Description

Implements the full GACL algorithm with configurable parameters. Pheromone evolution across generations follows update equations isomorphic to stochastic gradient descent.

Usage

```
gacl(
  site_qualities,
  n_ants = 100,
  n_generations = 50,
  n_waves = 10,
  rho_wave = 0.3,
  rho_gen = 0.1,
  gamma = 0.5,
  alpha = 1,
  beta = 1,
  noise_sd = 0.2
)
```

Arguments

`site_qualities`	Numeric vector of true qualities for K sites.
`n_ants`	Number of ants per generation (default 100).
`n_generations`	Number of generations to simulate (default 50).
`n_waves`	Number of recruitment waves per generation (default 10).
`rho_wave`	Within-generation evaporation rate (default 0.3).
`rho_gen`	Between-generation evaporation rate, analogous to the learning rate η in neural networks (default 0.1).
`gamma`	Pheromone deposition rate (default 0.5).
`alpha`	Pheromone influence exponent (default 1).
`beta`	Heuristic influence exponent (default 1).
`noise_sd`	Standard deviation of observation noise (default 0.2).

Value

A list with components:

pheromone_history Matrix (generations x K) of pheromone levels.

fitness_history Numeric vector of colony fitness per generation.

error_signal_history Numeric vector of negative fitness (loss analog).

decision_history Integer vector; best site per generation.

final_decision Best site at the last generation.

Examples

```
result <- gacl(c(10, 7, 5, 4, 3), n_generations = 30)
plot(result$fitness_history, type = "l")
```

---

generate_all_figures *Generate All Manuscript Figures*

Description

Runs all experiments and saves Figures 1–5 as PDFs in the specified output directory.

Usage

```
generate_all_figures(output_dir, verbose = TRUE)
```

Arguments

output_dir	Path to directory for saved figures. No default is provided to avoid writing to the user's home filespace; use `tempdir()` or another user-chosen directory.
verbose	Logical; if `TRUE` (the default), progress messages are emitted via `message` and can be suppressed with `suppressMessages`.

Value

Invisible NULL. Figures are saved as side effects.

---

generate_classification_data *Generate Synthetic Classification Data*

Description

Creates a binary classification problem where the true boundary depends on the first three features: $y = sign(x_1 + x_2 x_3 + 0.5x_3^2)$, with optional label noise.

Usage

```
generate_classification_data(n = 500, p = 10, noise = 0.2)
```

Arguments

n	Number of observations.
p	Number of features.
noise	Fraction of labels flipped.

Value

A list with `X` (n x p matrix), `y` (label vector), and `true_labels`.

`generate_regression_data` *Generate Synthetic Regression Data*

Description

Creates data with a known sparse signal (first 5 features active) and controllable signal-to-noise ratio. Used to benchmark both random forest and ant colony simulation experiments.

Usage

```
generate_regression_data(n = 1000, p = 50, signal_strength = 1, noise_sd = 1)
```

Arguments

`n`	Number of observations
`p`	Number of features
`signal_strength`	Multiplier for the true signal
`noise_sd`	Standard deviation of additive Gaussian noise

Value

A list with components:

X A data.frame of features (`n` rows, `p` columns)

y Numeric response vector

f_true True function values (without noise)

Examples

```
d <- generate_regression_data(n = 200, p = 20, signal_strength = 2, noise_sd = 1)
plot(d$f_true, d$y, xlab = "True signal", ylab = "Observed response")
```

generate_synthetic_data *Generate Synthetic Classification Data*

Description

Creates a binary classification dataset with a non-linear decision boundary whose complexity can be varied.

Usage

```
generate_synthetic_data(n = 1000, p = 5, noise = 0.1, complexity = 2)
```

Arguments

n	Number of samples (default 1000).
p	Number of features (default 5).
noise	Label-flip noise rate, between 0 and 0.5 (default 0.1).
complexity	Complexity of the boundary: 1 = linear, 2 = quadratic, 3 = complex with interactions (default 2).

Value

A list with components X (matrix), y (labels with noise), and true_labels.

Examples

```
d <- generate_synthetic_data(n = 500, p = 5, complexity = 2)
table(d$y)
```

isomorphism_test *Experiment 3: Direct Isomorphism Test*

Description

Runs random forest and ant colony under matched θ values (m_try/p = p_explore) and compares the resulting pairwise correlation and ensemble variance.

Usage

```
isomorphism_test(n_replicates = 50, p_vals = c(0.1, 0.3, 0.5, 0.7, 0.9))
```

Arguments

n_replicates	Monte Carlo replicates per θ
p_vals	Numeric vector of θ values

Value

A data.frame with theta, system, correlation, cor_sd, ensemble_var, var_sd

---

launch_app *Launch an Interactive Shiny App*

Description

Opens an interactive Shiny application for exploring the ant colony / machine learning isomorphisms.

Usage

```
launch_app(part = c("part1", "part2"), ...)
```

Arguments

part	Which part of the trilogy to explore: "part1" for the random forest isomorphism (decorrelation, variance decomposition) or "part2" for the boosting isomorphism (adaptive recruitment, weak learnability).
...	Additional arguments passed to runApp.

Value

No return value, called for side effects (launches a Shiny app).

Examples

```
if (interactive()) launch_app("part1")
if (interactive()) launch_app("part2")
```

noise_experiment_boost

Noise Robustness Experiment

Description

Compares how AdaBoost and ACAR degrade under increasing noise. Both systems are measured as P(correct) across independent replicates. Parameters are calibrated so both show visible, parallel degradation.

Usage

```
noise_experiment_boost(
  noise_levels = seq(0, 0.45, by = 0.05),
  n_replicates = 50,
  verbose = TRUE
)
```

Arguments

noise_levels	Numeric vector of noise fractions (0 to 0.45).
n_replicates	Number of Monte Carlo replicates per level.
verbose	Logical; if TRUE (the default), progress messages are emitted via message and can be suppressed with suppressMessages.

Value

A data.frame with columns noise, accuracy, and system.

optimal_decorrelation_experiment

Experiment 4: Optimal Decorrelation

Description

Sweeps ensemble size and θ to find the performance-optimal decorrelation parameter. Uses MSE for random forests and 1-accuracy for ant colonies so both metrics point downward.

Usage

```
optimal_decorrelation_experiment(
  ensemble_sizes = c(10, 30, 100),
  theta_values = seq(0.1, 0.9, by = 0.1),
  n_replicates = 30
)
```

Arguments

ensemble_sizes	Integer vector of ensemble sizes
theta_values	Numeric vector of θ values
n_replicates	Monte Carlo replicates

Value

A data.frame with M, theta, system, performance, se

---

plot_colony_accuracy *Supplementary: Colony Accuracy vs Size*

Description

Supplementary: Colony Accuracy vs Size

Usage

```
plot_colony_accuracy(colony_results)
```

Arguments

colony_results	Output of colony_variance_experiment

Value

A ggplot object

---

plot_convergence_boost *Plot Figure 4: Convergence Rates*

Description

Plot Figure 4: Convergence Rates

Usage

```
plot_convergence_boost(results)
```

Arguments

results	Output of convergence_experiment_boost().

Value

A ggplot object.

`plot_convergence_complexity` *Plot Convergence Across Complexity (Figure 6)*

Description

Three-panel plot showing convergence for linear, quadratic, and complex decision boundaries.

Usage

```
plot_convergence_complexity(n_replicates = 15, n_generations = 50)
```

Arguments

`n_replicates`	Replicates per complexity (default 15).
`n_generations`	Generations/epochs (default 50).

Value

Invisibly, `NULL`.

`plot_correlation_decay` *Figure 3: Correlation Decay Comparison*

Description

Figure 3: Correlation Decay Comparison

Usage

```
plot_correlation_decay(iso_results)
```

Arguments

`iso_results`	Output of `isomorphism_test`

Value

A ggplot object

plot_gradient_dynamics *Plot Gradient Dynamics (Figure 7)*

Description

Two-panel plot comparing the error signal and gradient magnitude in both systems.

Usage

```
plot_gradient_dynamics(site_qualities = c(10, 7, 5, 4, 3), n_generations = 50)
```

Arguments

site_qualities	Numeric vector of site qualities.
n_generations	Generations/epochs (default 50).

Value

Invisibly, a list with both results.

plot_isomorphism *Plot the Gradient Descent Isomorphism (Figure 1)*

Description

Runs one GACL and one neural-network simulation and overlays their normalised error/loss trajectories.

Usage

```
plot_isomorphism(site_qualities = c(10, 7, 5, 4, 3), n_generations = 50, ...)
```

Arguments

site_qualities	Numeric vector of site qualities (default c(10, 7, 5, 4, 3)).
n_generations	Number of generations/epochs (default 50).
...	Additional graphical parameters passed to plot.

Value

Invisibly, a list with the GACL and NN results.

plot_learning_curves *Plot Learning Curves with Replicates (Figure 2)*

Description

Runs multiple GACL and neural-network replicates, plots individual traces, mean curves, and SE ribbons.

Usage

```
plot_learning_curves(
  site_qualities = c(10, 7, 5, 4, 3),
  n_replicates = 20,
  n_generations = 50,
  ...
)
```

Arguments

site_qualities	Numeric vector of site qualities.
n_replicates	Number of independent replicates (default 20).
n_generations	Number of generations/epochs (default 50).
...	Additional graphical parameters.

Value

Invisibly, a list with gacl_mat and nn_mat.

plot_learning_rate_sensitivity *Plot Learning Rate Sensitivity (Figure 4)*

Description

Sweeps the evaporation rate / learning rate and plots final normalised performance for both systems with error bars.

Usage

```
plot_learning_rate_sensitivity(
  site_qualities = c(10, 7, 5, 4, 3),
  learning_rates = c(0.02, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5),
  n_replicates = 15,
  n_generations = 30
)
```

Arguments

site_qualities	Numeric vector of site qualities.
learning_rates	Numeric vector of rates to test.
n_replicates	Replicates per rate (default 15).
n_generations	Generations/epochs per run (default 30).

Value

Invisibly, a data frame of summary statistics.

---

plot_margin_quorum *Plot Figure 3: Margin vs Quorum*

Description

Plot Figure 3: Margin vs Quorum

Usage

```
plot_margin_quorum(boost_result, X, y, acar_result)
```

Arguments

boost_result	Output of adaboost().
X	Predictor matrix used in boosting.
y	Label vector used in boosting.
acar_result	Output of acar().

Value

A combined patchwork plot.

plot_noise_robustness_boost

Plot Figure 5: Noise Robustness

Description

Plot Figure 5: Noise Robustness

Usage

```
plot_noise_robustness_boost(results)
```

Arguments

results	Output of noise_experiment_boost().

Value

A ggplot object.

plot_noise_robustness_nn

Plot Noise Robustness (Figure 5)

Description

Sweeps noise levels and compares degradation in both systems.

Usage

```
plot_noise_robustness_nn(
  site_qualities = c(10, 7, 5, 4, 3),
  noise_levels = seq(0, 0.5, by = 0.05),
  n_replicates = 15
)
```

Arguments

site_qualities	Numeric vector of site qualities.
noise_levels	Numeric vector of noise levels to test.
n_replicates	Replicates per level (default 15).

Value

Invisibly, a summary data frame.

`plot_optimal_decorrelation` *Figure 4: Optimal Decorrelation*

Description

Figure 4: Optimal Decorrelation

Usage

```
plot_optimal_decorrelation(optimal_results)
```

Arguments

`optimal_results`	Output of `optimal_decorrelation_experiment`

Value

A ggplot object

`plot_pheromone_weight` *Plot Pheromone vs Weight Evolution (Figure 3)*

Description

Two-panel plot comparing the evolution of pheromone concentrations across sites with the evolution of representative neural-network weights across epochs.

Usage

```
plot_pheromone_weight(site_qualities = c(10, 7, 5, 4, 3), n_generations = 50)
```

Arguments

`site_qualities`	Numeric vector of site qualities.
`n_generations`	Number of generations/epochs (default 50).

Value

Invisibly, the GACL result.

plot_plasticity *Plot Plasticity and Adaptation (Figure 8)*

Description

Simulates an environmental shift at the midpoint and compares the recovery dynamics of both systems.

Usage

```
plot_plasticity(n_replicates = 15)
```

Arguments

n_replicates	Replicates (default 15).

Value

Invisibly, NULL.

plot_sensitivity_heatmap *Figure 5: Sensitivity Heat-map*

Description

Figure 5: Sensitivity Heat-map

Usage

```
plot_sensitivity_heatmap(sensitivity_results)
```

Arguments

sensitivity_results	Output of sensitivity_analysis

Value

A ggplot object

`plot_variance_decomposition`

Figure 2: Variance Decomposition

Description

Side-by-side panels showing (A) correlation vs `m_try/p` and (B) theoretical vs empirical ensemble variance.

Usage

```
plot_variance_decomposition(rf_results)
```

Arguments

`rf_results`	Output of `variance_decomposition_experiment`

Value

A combined ggplot (via `ggpubr::ggarrange`)

`plot_weak_learnability`

Plot Figure 1: Weak Learnability Theorem

Description

Plot Figure 1: Weak Learnability Theorem

Usage

```
plot_weak_learnability(results)
```

Arguments

`results`	Output of `weak_learnability_experiment()`.

Value

A `ggplot` object.

plot_weight_pheromone *Plot Figure 2: Weight vs Pheromone Evolution*

Description

Plot Figure 2: Weight vs Pheromone Evolution

Usage

```
plot_weight_pheromone(boost_result, acar_result, site_qualities)
```

Arguments

boost_result	Output of adaboost().
acar_result	Output of acar().
site_qualities	Numeric vector used in the ACAR call.

Value

A combined patchwork plot.

predict_adaboost *Predict with an AdaBoost Ensemble*

Description

Predict with an AdaBoost Ensemble

Usage

```
predict_adaboost(result, X, up_to = NULL)
```

Arguments

result	Output of adaboost().
X	Numeric predictor matrix.
up_to	Optional integer; use only the first up_to iterations.

Value

Numeric prediction vector in {-1, +1}.

predict_stump *Predict with a Decision Stump*

Description

Predict with a Decision Stump

Usage

```
predict_stump(stump, X)
```

Arguments

stump	A stump list from find_best_stump.
X	Numeric predictor matrix.

Value

Numeric prediction vector in {-1, +1}.

sensitivity_analysis *Experiment 5: Sensitivity Analysis*

Description

Tests whether the isomorphism holds under varying signal strengths, noise levels, and θ values.

Usage

```
sensitivity_analysis(
  signal_strengths = c(0.5, 1, 2, 5),
  noise_levels = c(0.5, 1, 2, 4),
  n_replicates = 20
)
```

Arguments

signal_strengths	Numeric vector of signal multipliers
noise_levels	Numeric vector of noise standard deviations
n_replicates	Monte Carlo replicates

Value

A data.frame with signal, noise, theta, system, correlation, cor_sd

`simple_neural_network` *Simple Neural Network with Stochastic Gradient Descent*

Description

Implements a single-hidden-layer perceptron trained with mini-batch SGD and binary cross-entropy loss. Under the isomorphism, weight updates correspond to pheromone updates in `gacl()`.

Usage

```
simple_neural_network(
  X,
  y,
  n_epochs = 50,
  batch_size = 32,
  learning_rate = 0.1,
  n_hidden = 10,
  validation_split = 0.2
)
```

Arguments

`X`	Numeric matrix of input features (n x p).
`y`	Numeric vector of labels ({-1, 1} or {0, 1}).
`n_epochs`	Number of training epochs (default 50).
`batch_size`	Mini-batch size (default 32).
`learning_rate`	Learning rate η, analogous to the evaporation rate ρ in GACL (default 0.1).
`n_hidden`	Number of hidden units (default 10).
`validation_split`	Fraction of data held out for validation (default 0.2).

Value

A list with components:

train_loss Numeric vector of training loss per epoch.

val_loss Numeric vector of validation loss per epoch.

val_acc Numeric vector of validation accuracy per epoch.

gradient_norm Numeric vector of gradient L2 norm per epoch.

final_weights List of weight matrices (W1, b1, W2, b2).

Examples

```
d <- generate_synthetic_data(n = 300, p = 5)
nn <- simple_neural_network(d$X[1:240, ], d$y[1:240], n_epochs = 30)
plot(nn$val_acc, type = "l")
```

`simulate_ant_colony` *Simulate an Ant Colony Decision Process*

Description

Agent-based simulation of *Temnothorax*-style nest-site selection. Each ant probabilistically explores or follows pheromone trails, makes noisy observations of site quality, and contributes to recruitment. The colony reaches a decision when recruitment exceeds a quorum threshold.

Usage

```
simulate_ant_colony(
  n_ants = 50,
  n_sites = 5,
  site_qualities = c(10, 8, 6, 4, 2),
  p_explore = 0.3,
  n_steps = 100,
  noise_sd = 2,
  quorum_threshold = 20,
  alpha = 0.1,
  beta = 0.05,
  gamma = 0.2
)
```

Arguments

`n_ants`	Number of ants in the colony
`n_sites`	Number of candidate nest sites
`site_qualities`	Numeric vector of true site quality values
`p_explore`	Probability that an ant explores independently (vs following pheromone). This is the decorrelation parameter, analogous to `m_try/p` in random forests.
`n_steps`	Maximum number of simulation time steps
`noise_sd`	Standard deviation of observation noise
`quorum_threshold`	Recruitment level that triggers a colony decision
`alpha`	Learning rate for pheromone updates
`beta`	Pheromone evaporation rate (0 to 1)
`gamma`	Recruitment strength multiplier

Value

A list with components:

history List of per-step snapshots (pheromone, recruitment, preferences)

final_preferences Colony-average quality estimates per site

final_recruitment Final recruitment levels per site

decision Index of chosen site

ant_states Matrix of visit counts (`n_ants` x `n_sites`)

ant_preferences Matrix of estimated qualities (`n_ants` x `n_sites`)

Examples

```
sim <- simulate_ant_colony(n_ants = 30, p_explore = 0.4)
cat("Colony chose site", sim$decision, "\n")
```

`sim_boost_recruitment` *Boosting and Adaptive Recruitment Simulation*

Description

Runs AdaBoost and ant colony adaptive recruitment in parallel.

Usage

```
sim_boost_recruitment(n_iterations = 100, n_ants = 50,
                      n_sites = 10, evaporation_rate = 0.1,
                      noise_level = 0.15, n_rep = 100)
```

Arguments

`n_iterations`	Number of boosting iterations / recruitment waves.
`n_ants`	Number of ants.
`n_sites`	Number of candidate sites.
`evaporation_rate`	Pheromone evaporation rate.
`noise_level`	Noise level for data generation.
`n_rep`	Monte Carlo replicates for convergence curves.

Value

An S3 object of class `"boost_recruitment"` with elements `$boost` and `$colony` (numeric learning curves), and `print()` and `plot()` methods.

sim_colony_convergence *Colony Convergence Simulation*

Description

Tracks the colony's probability estimates over time, demonstrating convergence to the best site via Thompson sampling and recruitment.

Usage

```
sim_colony_convergence(n_ants = 50, n_sites = 5,
                       site_qualities = c(0.9, 0.7, 0.5, 0.3, 0.1),
                       n_steps = 200, p_explore = 0.3)
```

Arguments

n_ants	Number of ants.
n_sites	Number of candidate sites.
site_qualities	True quality vector.
n_steps	Number of time steps.
p_explore	Exploration probability.

Value

An S3 object of class "colony_convergence" with print() and plot() methods.

sim_decorrelation *Decorrelation Parameter Sweep*

Description

Sweeps the decorrelation parameter theta for both random forests and ant colonies and returns empirical pairwise correlations.

Usage

```
sim_decorrelation(theta_range = seq(0, 1, by = 0.05),
                  n_sites = 20, n_ants = 50, n_trees = 50,
                  n_features = 20, n_rep = 200)
```

Arguments

theta_range	Numeric vector of theta values.
n_sites	Number of candidate sites (ant colony).
n_ants	Number of ants.
n_trees	Number of trees.
n_features	Number of features (p).
n_rep	Monte Carlo replicates per theta.

Value

An S3 object of class "decorrelation" with print() and plot() methods.

---

sim_gradient_colony *Gradient Descent and Generational Colony Learning*

Description

Simulates generational pheromone evolution alongside neural network training via SGD.

Usage

```
sim_gradient_colony(n_generations = 50, n_ants = 30,
                    n_trails = 10, evaporation_rate = 0.05,
                    learning_rate = 0.05,
                    hidden_units = c(16, 8),
                    task = "classification", n_rep = 50)
```

Arguments

n_generations	Number of generations / epochs.
n_ants	Number of ants per generation.
n_trails	Number of trails / sites.
evaporation_rate	Pheromone evaporation rate.
learning_rate	Neural network learning rate.
hidden_units	Integer vector of hidden layer sizes.
task	"classification" or "regression".
n_rep	Number of replicates for learning curves.

Value

An S3 object of class "gradient_colony" with elements $colony and $neural_net (numeric vectors of normalized performance), and print(), summary(), and plot() methods.

sim_margin_analysis *Margin Analysis*

Description

Computes margin distributions for boosting and ant colony quorum decisions.

Usage

```
sim_margin_analysis(n_iterations = 200, n_ants = 100,
                    noise_level = 0.2)
```

Arguments

n_iterations	Number of boosting iterations.
n_ants	Number of ants.
noise_level	Noise level.

Value

An S3 object of class "margin_analysis" with print() and plot() methods.

sim_plasticity *Plasticity and Environmental Adaptation*

Description

Demonstrates neural plasticity / colony adaptation correspondence, with optional environmental shift.

Usage

```
sim_plasticity(n_generations = 300, n_trails = 20,
               ltp_rate = 0.1, ltd_rate = 0.05,
               prune_threshold = 0.01, genesis_rate = 0.02,
               env_shift_at = NULL)
```

Arguments

n_generations	Number of generations.
n_trails	Number of trails.
ltp_rate	Long-term potentiation / reinforcement rate.
ltd_rate	Long-term depression / evaporation rate.
prune_threshold	Pruning threshold.
genesis_rate	New trail formation rate.
env_shift_at	Generation at which environment shifts (NULL = no shift).

Value

An S3 object of class "plasticity_sim" with print() and plot() methods.

---

sim_variance_decomp *Variance Decomposition Simulation*

Description

Simulates both an ant colony and a random forest across a range of ensemble sizes and correlation levels, validating the variance decomposition formula.

Usage

```
sim_variance_decomp(N_range = c(5, 10, 25, 50, 100),
                    rho_range = seq(0, 1, by = 0.2),
                    sigma2 = 1, n_rep = 500)
```

Arguments

N_range	Integer vector of ensemble sizes.
rho_range	Numeric vector of pairwise correlation levels.
sigma2	Individual unit variance.
n_rep	Number of Monte Carlo replicates.

Value

An S3 object of class "variance_decomp" with print(), summary(), and plot() methods.

---

test_isomorphism *Test Isomorphism Between Two Learning Curves*

Description

Compares two numeric vectors (e.g. colony fitness and NN loss) using a Kolmogorov-Smirnov test, correlation analysis, or dynamic time warping.

Usage

```
test_isomorphism(ant_result, ml_result,
                 method = c("ks", "correlation", "dtw"))
```

Arguments

ant_result	Numeric vector (colony learning curve).
ml_result	Numeric vector (ML learning curve).
method	One of "ks", "correlation", or "dtw".

Value

An S3 object of class `"iso_test"` with `print()` and `summary()` methods.

---

track_weights *Track AdaBoost Weight Evolution*

Description

Convenience wrapper that returns only the weight history.

Usage

```
track_weights(X, y, M = 50)
```

Arguments

`X`	Predictor matrix.
`y`	Label vector.
`M`	Number of iterations.

Value

An M x n matrix of instance weights.

---

variance_decomposition_experiment *Experiment 1: Random Forest Variance Decomposition*

Description

Validates that $\mathrm{Var}[\hat{f}_{\mathrm{rf}}] = \rho\sigma^2 + (1-\rho)\sigma^2/M$ holds empirically and that ρ varies with `m_try/p`.

Usage

```
variance_decomposition_experiment(
  n_train = 500,
  n_test = 1000,
  p = 50,
  m_try_values = c(1, 2, 5, 10, 20, 50),
  n_trees = 500,
  n_replicates = 100,
  signal_strength = 2,
  noise_sd = 1
)
```

Arguments

n_train	Training set size
n_test	Test set size
p	Number of features
m_try_values	Integer vector of mtry values to sweep
n_trees	Number of trees per forest
n_replicates	Number of Monte Carlo replicates
signal_strength	Signal multiplier for generate_regression_data
noise_sd	Noise level for generate_regression_data

Value

A data.frame with columns m_try, rep, rho, sigma2, ensemble_var, pred_error, theoretical_var

---

weak_learnability_experiment

Weak Learnability Experiment

Description

Tests whether gamma-weak ant colonies converge to correct decisions with sufficient recruitment waves, as predicted by Theorem 4. Parameters are calibrated so that small gamma is genuinely hard (slow convergence) while large gamma converges quickly.

Usage

```
weak_learnability_experiment(
  gamma_values = c(0.05, 0.1, 0.2, 0.3),
  wave_counts = c(3, 5, 8, 12, 18, 25, 40, 60, 90, 130, 200, 300),
  n_replicates = 200,
  verbose = TRUE
)
```

Arguments

gamma_values	Numeric vector of weakness parameters.
wave_counts	Integer vector of wave counts to test.
n_replicates	Number of Monte Carlo replicates per combination.
verbose	Logical; if TRUE (the default), progress messages are emitted via message and can be suppressed with suppressMessages.

Value

A data.frame with columns gamma, waves, rep, and accuracy.

`within_colony_correlation` *Compute Within-Colony Ant Correlation*

Description

Measures the mean pairwise correlation between ants' preference vectors over all candidate sites within a single colony. This is the correct analogue of tree-tree correlation in a random forest: each ant's preference vector over K sites corresponds to a tree's prediction vector over test points.

Usage

```
within_colony_correlation(ant_preferences)
```

Arguments

`ant_preferences` An `n_ants` x `n_sites` matrix of quality estimates, as returned by `simulate_ant_colony`.

Value

Mean pairwise correlation (scalar), or `NA` if fewer than 2 ants have visited at least 2 sites.

Examples

```
sim <- simulate_ant_colony(n_ants = 30, p_explore = 0.3)
within_colony_correlation(sim$ant_preferences)
```

Index