Package 'FastLZeroSpikeInference'

December 21, 2018

Maintainer Sean Jewell <swjewell@uw.edu>

Author Sean Jewell, Toby Dylan Hocking

Version 2018.12.10

License GPL-3

Title Fast Nonconvex Deconvolution of Calcium Imaging Data

Description Estimate spike times from calcium imaging data using an L0 penalty.

Depends R (>= 2.10)

Suggests testthat

RoxygenNote 6.1.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-12-21 13:50:12 UTC

R topics documented:

`estimate_calcium` *Estimate underlying calcium concentration based on estimated spikes*

Description

Estimate underlying calcium concentration based on estimated spikes

Usage

```
estimate_calcium(fit)
```

Arguments

`fit`	object created by running estimate_spikes

Details

This algorithm solves the optimization problems

AR(1) model:

minimize_c1,...,cT 0.5 sum_t=1^T (y_t - c_t)^2 + lambda sum_t=2^T 1_[c_t != max(gam c_t-1, EPS)]

for the global optimum, where y_t is the observed fluorescence at the tth timestep.

Constrained AR(1) model:

minimize_c1,...,cT 0.5 sum_t=1^T (y_t - c_t)^2 + lambda sum_t=2^T 1_[c_t != max(gam c_t-1, EPS)]

subject to c_t >= max(gam c_t-1, EPS), t = 2, ..., T

We introduce the constant EPS > 0, to avoid arbitrarily small calcium concentrations that would result in numerical instabilities. In practice, this means that the estimated calcium concentration decays according to the AR(1) model for values greater than EPS and is equal to EPS thereafter.

When estimating the spikes, it is not necessary to explicitly compute the calcium concentration. Therefore, if only the spike times are required, the user can avoid this computation cost by setting the estimate_calcium boolean to false. Because estimating the calcium requires additional computation time, we suggest estimating the calcium only if it is needed.

Given the set of estimated spikes produced from the estimate_spike, the calcium concentration can be estimated with the estimate_calcium function (see examples below).

For additional information see:

1. Jewell, Hocking, Fearnhead, and Witten (2018) <arXiv:1802.07380> and

2. Jewell, Sean; Witten, Daniela. Exact spike train inference via l0 optimization. Ann. Appl. Stat. 12 (2018), no. 4, 2457–2482. doi:10.1214/18-AOAS1162. https://projecteuclid.org/euclid.aoas/1542078052

Value

Returns a list with elements:

`spikes` the set of estimated spikes

`estimated_calcium` estimated calcium concentration

`change_pts` the set of changepoints

`cost` the cost at each time point

`n_intervals` the number of intervals at each point

See Also

Estimate spikes: `estimate_spikes` `estimate_calcium`

Simulate: `simulate_ar1`

Examples

```
sim <- simulate_ar1(n = 500, gam = 0.95, poisMean = 0.009, sd = 0.05, seed = 1)
plot(sim)

## Fits for a single tuning parameter

# AR(1) model
fit <- estimate_spikes(dat = sim$fl, gam = 0.95, lambda = 1)
print(fit)

# compute fitted values from prev. fit
fit <- estimate_calcium(fit)
plot(fit)

# or
fit <- estimate_spikes(dat = sim$fl, gam = 0.95, lambda = 1, estimate_calcium = TRUE)
plot(fit)

# Constrained AR(1) model
fit <- estimate_spikes(dat = sim$fl, gam = 0.95, lambda = 1, constraint = TRUE,
                                                    estimate_calcium = TRUE)
print(fit)
plot(fit)
```

`estimate_spikes` *Estimate spike train, underlying calcium concentration, and changepoints based on a fluorescence trace.*

Description

Estimate spike train, underlying calcium concentration, and changepoints based on a fluorescence trace.

Usage

```
estimate_spikes(dat, gam, lambda, constraint = FALSE,
  estimate_calcium = FALSE, EPS = 1e-04)
```

Arguments

dat	fluorescence data
gam	a scalar value for the AR(1) decay parameter
lambda	tuning parameter lambda
constraint	boolean specifying constrained or unconstrained optimization problem (see below)
estimate_calcium	boolean specifying whether to estimate the calcium
EPS	double specifying the minimum calcium value

Details

This algorithm solves the optimization problems

AR(1) model:

minimize_c1,...,cT 0.5 sum_t=1^T (y_t - c_t)^2 + lambda sum_t=2^T 1_[c_t != max(gam c_t-1, EPS)]

for the global optimum, where y_t is the observed fluorescence at the tth timestep.

Constrained AR(1) model:

minimize_c1,...,cT 0.5 sum_t=1^T (y_t - c_t)^2 + lambda sum_t=2^T 1_[c_t != max(gam c_t-1, EPS)]

subject to c_t >= max(gam c_t-1, EPS), t = 2, ..., T

We introduce the constant EPS > 0, to avoid arbitrarily small calcium concentrations that would result in numerical instabilities. In practice, this means that the estimated calcium concentration decays according to the AR(1) model for values greater than EPS and is equal to EPS thereafter.

When estimating the spikes, it is not necessary to explicitly compute the calcium concentration. Therefore, if only the spike times are required, the user can avoid this computation cost by setting the estimate_calcium boolean to false. Because estimating the calcium requires additional computation time, we suggest estimating the calcium only if it is needed.

Given the set of estimated spikes produced from the estimate_spike, the calcium concentration can be estimated with the estimate_calcium function (see examples below).

For additional information see:

1. Jewell, Hocking, Fearnhead, and Witten (2018) <arXiv:1802.07380> and

2. Jewell, Sean; Witten, Daniela. Exact spike train inference via l0 optimization. Ann. Appl. Stat. 12 (2018), no. 4, 2457–2482. doi:10.1214/18-AOAS1162. https://projecteuclid.org/euclid.aoas/1542078052

Value

Returns a list with elements:

`spikes` the set of estimated spikes

`estimated_calcium` estimated calcium concentration

`change_pts` the set of changepoints

`cost` the cost at each time point

`n_intervals` the number of intervals at each point

See Also

Estimate spikes: `estimate_spikes` `estimate_calcium`

Simulate: `simulate_ar1`

Examples

```
sim <- simulate_ar1(n = 500, gam = 0.95, poisMean = 0.009, sd = 0.05, seed = 1)
plot(sim)

## Fits for a single tuning parameter

# AR(1) model
fit <- estimate_spikes(dat = sim$fl, gam = 0.95, lambda = 1)
print(fit)

# compute fitted values from prev. fit
fit <- estimate_calcium(fit)
plot(fit)

# or
fit <- estimate_spikes(dat = sim$fl, gam = 0.95, lambda = 1, estimate_calcium = TRUE)
plot(fit)

# Constrained AR(1) model
fit <- estimate_spikes(dat = sim$fl, gam = 0.95, lambda = 1, constraint = TRUE,
                                                    estimate_calcium = TRUE)
print(fit)
plot(fit)
```

`estimate_spike_paths` *Estimate spike train, underlying calcium concentration, and changepoints based on a fluorescence trace. Automatic tuning parameter selection within a range of values [lambda_min, lambda_max]*

Description

Estimate spike train, underlying calcium concentration, and changepoints based on a fluorescence trace. Automatic tuning parameter selection within a range of values [lambda_min, lambda_max]

Usage

```
estimate_spike_paths(dat, gam, lambda_min = 0.01, lambda_max = 10,
  constraint = FALSE, EPS = 1e-04, max_iters = 10)
```

Arguments

dat	fluorescence data
gam	a scalar value for the AR(1) decay parameter
lambda_min	minimum lamba value
lambda_max	maximum lamba value
constraint	boolean specifying constrained or unconstrained optimization problem (see below)
EPS	double specifying the minimum calcium value
max_iters	maximum number of tuning parameters attempted

Details

This algorithm solves the optimization problems

AR(1) model:

minimize_c1,...,cT 0.5 sum_t=1^T (y_t - c_t)^2 + lambda sum_t=2^T 1_[c_t != max(gam c_t-1, EPS)]

for the global optimum, where y_t is the observed fluorescence at the tth timestep.

Constrained AR(1) model:

minimize_c1,...,cT 0.5 sum_t=1^T (y_t - c_t)^2 + lambda sum_t=2^T 1_[c_t != max(gam c_t-1, EPS)]

subject to c_t >= max(gam c_t-1, EPS), t = 2, ..., T

We introduce the constant EPS > 0, to avoid arbitrarily small calcium concentrations that would result in numerical instabilities. In practice, this means that the estimated calcium concentration decays according to the AR(1) model for values greater than EPS and is equal to EPS thereafter.

When estimating the spikes, it is not necessary to explicitly compute the calcium concentration. Therefore, if only the spike times are required, the user can avoid this computation cost by setting the estimate_calcium boolean to false. Because estimating the calcium requires additional computation time, we suggest estimating the calcium only if it is needed.

Given the set of estimated spikes produced from the estimate_spike, the calcium concentration can be estimated with the estimate_calcium function (see examples below).

For additional information see:

1. Jewell, Hocking, Fearnhead, and Witten (2018) <arXiv:1802.07380> and

2. Jewell, Sean; Witten, Daniela. Exact spike train inference via l0 optimization. Ann. Appl. Stat. 12 (2018), no. 4, 2457–2482. doi:10.1214/18-AOAS1162. https://projecteuclid.org/euclid.aoas/1542078052

Value

Returns a list with elements:

`path_stats` a dataframe with summary statistics (number of spikes, tuning parameters, cost)

`path_fits` a list with estimated_spikes object for each tuning parameter

`approximate_path` a boolean indicating whether an early stopping criterion condition occurred

See Also

Estimate spikes: `estimate_spikes` `estimate_calcium`

Simulate: `simulate_ar1`

Examples

```
sim <- simulate_ar1(n = 500, gam = 0.95, poisMean = 0.009, sd = 0.05, seed = 1)
plot(sim)

## Fits for tuning parameters between [0.1, 10]
fits <- estimate_spike_paths(dat = sim$fl, gam = 0.95, lambda_min = 0.1, lambda_max = 10)
print(fits)
plot(fits)
print(fits$path_fits[[1]])
plot(fits$path_fits[[1]])

## Fits for a single tuning parameter

# AR(1) model
fit <- estimate_spikes(dat = sim$fl, gam = 0.95, lambda = 1)
print(fit)

# compute fitted values from prev. fit
fit <- estimate_calcium(fit)
plot(fit)

# or
fit <- estimate_spikes(dat = sim$fl, gam = 0.95, lambda = 1, estimate_calcium = TRUE)
plot(fit)

# Constrained AR(1) model
fit <- estimate_spikes(dat = sim$fl, gam = 0.95, lambda = 1, constraint = TRUE,
                                                    estimate_calcium = TRUE)
print(fit)
plot(fit)
```

`FastLZeroSpikeInference` *FastLZeroSpikeInference: FastLZeroSpikeInference: A package for estimating spike times from calcium imaging data using an L0 penalty*

Description

This package implements an algorithm for deconvolving calcium imaging data for a single neuron in order to estimate the times at which the neuron spikes.

Details

This algorithm solves the optimization problems

AR(1) model:

minimize_c1,...,cT 0.5 sum_t=1^T (y_t - c_t)^2 + lambda sum_t=2^T 1_[c_t != max(gam c_t-1, EPS)]

for the global optimum, where y_t is the observed fluorescence at the tth timestep.

Constrained AR(1) model:

minimize_c1,...,cT 0.5 sum_t=1^T (y_t - c_t)^2 + lambda sum_t=2^T 1_[c_t != max(gam c_t-1, EPS)]

subject to c_t >= max(gam c_t-1, EPS), t = 2, ..., T

We introduce the constant EPS > 0, to avoid arbitrarily small calcium concentrations that would result in numerical instabilities. In practice, this means that the estimated calcium concentration decays according to the AR(1) model for values greater than EPS and is equal to EPS thereafter.

When estimating the spikes, it is not necessary to explicitly compute the calcium concentration. Therefore, if only the spike times are required, the user can avoid this computation cost by setting the estimate_calcium boolean to false. By default, the calcium concentration is not estimated.

Given the set of estimated spikes produced from the estimate_spike, the calcium concentration can be estimated with the estimate_calcium function (see examples below).

For additional information see:

1. Jewell, Hocking, Fearnhead, and Witten (2018) <arXiv:1802.07380> and

2. Jewell, Sean; Witten, Daniela. Exact spike train inference via l0 optimization. Ann. Appl. Stat. 12 (2018), no. 4, 2457–2482. doi:10.1214/18-AOAS1162. https://projecteuclid.org/euclid.aoas/1542078052

See Also

Estimate spikes: `estimate_spikes` `estimate_calcium`

Simulate: `simulate_ar1`

Examples

```
sim <- simulate_ar1(n = 500, gam = 0.95, poisMean = 0.009, sd = 0.05, seed = 1)
plot(sim)

## Fits for a single tuning parameter

# AR(1) model
fit <- estimate_spikes(dat = sim$fl, gam = 0.95, lambda = 1)
print(fit)

# compute fitted values from prev. fit
fit <- estimate_calcium(fit)
plot(fit)

# or
fit <- estimate_spikes(dat = sim$fl, gam = 0.95, lambda = 1, estimate_calcium = TRUE)
plot(fit)

# Constrained AR(1) model
fit <- estimate_spikes(dat = sim$fl, gam = 0.95, lambda = 1, constraint = TRUE,
                                                    estimate_calcium = TRUE)
print(fit)
plot(fit)
```

---

plot.estimated_spikes *Plot the solution to an L0 segmentation problem*

Description

Plot the solution to an L0 segmentation problem

Usage

```
## S3 method for class 'estimated_spikes'
plot(x, xlims = NULL, ...)
```

Arguments

x	output from running estimate_spikes
xlims	optional parameter to specify the x-axis limits
...	arguments to be passed to methods

See Also

estimate_spikes, estimate_calcium,

plot.estimated_spike_paths *Plot number of spikes vs. tuning parameter*

Description

Plot number of spikes vs. tuning parameter

Usage

```
## S3 method for class 'estimated_spike_paths'
plot(x, xlims = NULL, ...)
```

Arguments

x	output from running estimate_spike_paths
xlims	optional parameter to specify the x-axis limits
...	arguments to be passed to methods

See Also

estimate_spike_paths, estimate_spikes, estimate_calcium,

plot.simdata *Plot simulated data*

Description

Plot simulated data

Usage

```
## S3 method for class 'simdata'
plot(x, xlims = NULL, ...)
```

Arguments

x	output data from simulate_ar1
xlims	optional parameter to specify the x-axis limits
...	arguments to be passed to methods

Value

Plot with simulated fluorescence (dark grey circles), calcium concentration (dark green line) and spikes (dark green tick marks on x-axis)

See Also

estimate_spikes, estimate_calcium,

Examples

```
sim <- simulate_ar1(n = 500, gam = 0.998, poisMean = 0.009, sd = 0.05, seed = 1)
plot(sim)
```

---

print.estimated_spikes

Print estimated spikes

Description

Print estimated spikes

Usage

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

Arguments

x	estimated spikes
...	arguments to be passed to methods

---

print.estimated_spike_paths

Print estimated spike path

Description

Print estimated spike path

Usage

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

Arguments

x	estimated spikes path
...	arguments to be passed to methods

print.simdata *Print simulated data*

Description

Print simulated data

Usage

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

Arguments

x	simulated data
...	arguments to be passed to methods

simulate_ar1 *Simulate fluorescence trace based on simple AR(1) generative model*

Description

Simulate fluorescence trace based on simple AR(1) generative model

Usage

```
simulate_ar1(n, gam, poisMean, sd, seed)
```

Arguments

n	number of timesteps
gam	AR(1) decay rate
poisMean	mean for Poisson distributed spikes
sd	standard deviation
seed	random seed

Details

Simulate fluorescence trace based on simple AR(1) generative model

y_t = c_t + eps, eps ~ N(0, sd)

c_t = gam * c_t-1 + s_t

s_t ~ Pois(poisMean)

Value

spikes, fluorescence, and calcium concentration

Examples

```
sim <- simulate_ar1(n = 500, gam = 0.998, poisMean = 0.009, sd = 0.05, seed = 1)
plot(sim)
```

Index