ADVI: Taxi Routes experiment

Release 2024

Nafissa Benali, Laura Fuentes, Rita Maatouk

Mar 19, 2024

CONTENTS:

CHAPTER ONE

DF_PROCESSING MODULE

df_processing.**extract_traj**(*df*)

Extracts the trajectories stored (as strings) in a dataframe.

Parameters
`df` (*pd.DataFrame*) – Dataframe gathering trajectories information

Returns
list of processed trajectories (as np.array instead of str elements)

Return type
list_trips (list)

df_processing.**interpolation**(*ls*, *num_points*)

Linear interpolation of the given data points.

Parameters

- **x** (*list*) – List of x-coordinates.
- **y** (*list*) – List of y-coordinates.
- **num_points** (*int*) – Number of points for interpolation.

Returns
Tuple containing the interpolated x and y coordinates.

Return type
tuple

df_processing.**new_df**(*trips*, *nb_points*, *mask*, *newlist*)

Create a new dataframe with the new interpolated version of trajectories (50 coordinates) and saves it outside gitHub

Parameters

- **trips** (*pd.DataFrame*) – Dataframe with a fixed row length containing: trajectory id and trajectories
- **nb_points** (*int*) – Number of rows selected on dataframe trips
- **mask** (*np.array*) – matrix indicating whether trajectories have more than one coordinate or not
- **newlist** (*list*) – interpolated trajectories

df_processing.**plot_trajectories_list**(*ls*)

Plot all trajectories from a list of coordinates

Parameters
`ls` (*list*) – contains a group of coordinates (x,y)

CHAPTER

TWO

ADVI_FCTS MODULE

class advi_fcts.**ADVI_algorithm**(*data_dim*, *latent_dim*, *num_datapoints*, *dataset*, *nb_samples*, *lr*)

Bases: *Model*

T_inv(*theta*)

Computes the inverse transform of zeta, allowing to go from R^{dim} into support of $ heta$. In this case, z_prior and w_prior values are in the same space support as theta, hence no transformation is needed. Regarding alpha_prior and sigma_prior, they return only positive values, hence, computing the probability of alpha_sample<0 or sigma_sample<0 would give a nan. To solve such a problem, we consider for those two parameters the exponential transformation.

Parameters
theta (*tf.Tensor*) – Vector gathering the parameters to extract (z, w, sigma, alpha)

Returns
tf.concat([first_part, last_n_elements], axis=0) has same support as all prior variables

Return type
T^1(theta) (tf.Tensor)

Tinv_jac(*theta*)

Computes the log absolute value of the determinant of the jacobian matrix of T^(-1)

Parameters
theta (*tf.Tensor*) – Vector gathering the parameters to extract (z, w, sigma, alpha)

Returns
log-absolute value of the determinant of the jacobian of T^(-1)

Return type
det (tf.Tensor value)

fct_obj(*nb_samples*)

Computes by Monte Carlo (MC) integration: nabla_mu and nabla_omega (for mu and omega update) and elbo. :param nb_samples: Number of samples for MC integration :type nb_samples: int

Returns
estimation of nabla_mu by MC integration nabla_omega/nb_samples: estimation of nabla_omega by MC integration elbo/ nb_samples + entropy: estimation of elbo by MC integration

Return type
nabla_mu/ nb_samples

gradient_Tinv(*zeta*)

Compute the gradient of Tinv (T^(-1)) and the log-abs value of determinant of jacobian

Parameters
`zeta` (*tf.Tensor*) – Vector gathering the parameters to extract (z, w, sigma, alpha) before transformation (T_inv)

Returns
gradient of T_inv grad_log_jac_Tinv (tf.Tensor): gradient of log-abs value of determinant of jacobian

Return type
grad_Tinv (tf.Tensor)

`gradient_log_joint`(*theta*)

Compute the gradient of log-joint probability

Parameters
`theta` (*tf.Tensor*) – Vector gathering the parameters to extract (z, w, sigma, alpha)

Returns
gradient of log-joint value

Return type
grad (tf.Tensor)

`run_ADVI`()

Run the whole ADVI algorithm

Returns
estimated parameter of the distributional distribution: mu self.omega.numpy(): estimated parameter of the distributional distribution: omega

Return type
self.mu.numpy()

`step_size`(*i_value*, *lr*, *s*, *grad*, *tau=1*, *alpha=0.1*)

Computes the step-size for updating mu and omega

Parameters

- `i_value` (*int*) – iteration number
- `lr` (*float*) – learning rate
- `s` (*tf.tensor*) – Value
- `grad` (*tf.tensor*) – _description_
- `tau` (*int, optional*) – Defaults to 1.
- `alpha` (*int, optional*) – Defaults to 0.1.

Returns
estimation of step-size parameters

Return type
rho, s

class `advi_fcts.Model`(*data_dim*, *latent_dim*, *num_datapoints*, *dataset*)

Bases: `object`

z ~ Normal(0, I) w ~ Normal(0, I) sigma ~ LogNormal(1, 1) alpha ~ InvGamma(1, 1) `log_joint`(*theta*)

Compute log-joint probability of z, w, alpha, sigma, data according to the precised model

Parameters
`theta` (*tf.Tensor*) – Vector gathering the parameters to extract (z, w, sigma, alpha)

Returns
tf.reduce_sum(log_lik) + tf.reduce_sum(w_log_prior) + tf.reduce_sum(z_log_prior) + tf.reduce_sum(sigma_log_prior)

Return type
Value (tf.Tensor)

`params`(*theta*)

Extracts the samples from $ heta = (z, w, sigma, alpha)$

Parameters
`theta` (*tf.Tensor*) – tensor of dimension (self.dim) in R^{self.dim} gathering all parameters to extract (z, w, sigma, alpha)

Returns
extracted parameters

Return type
sigma, alpha, z, w (tf.Tensors)

CHAPTER

THREE

CLUSTERING MODULE

`clustering.`**`extract_from_VI`**`(`*mu*, *omega*, *advi_model*`)`

Extracts model parameters from output of PPCA (variational distribution results)

Parameters

- **mu** (*tf.Tensor*) – mean of variational distribution q
- **omega** (*tf.Tensor*) – standard deviation of variational distribution q
- **advi_model** (*class*) – model used for dimension reduction

Returns

model parameters

Return type

z, w, sigma (tf.Tensors)

`clustering.`**`perform_BGMM`**`(`*n_clusters*, *trajectories*, *x*`)`

Function that performs Bayesian GMM over a set of reduced dimension trajectories for a predefined number of clusters

Parameters

- **n_clusters** (*int*) – number of clusters
- **trajectories** (*np.array*) – projected trajectories of shape (num_datapoints, latent_dim)
- **x** (*pd.DataFrame*) – dataset with trajectories and IDs

Returns

description

Return type

type

CHAPTER FOUR

FIGS MODULE

figs.**ELBO_fig**(*elbo_evol*)

Function that creates the figure of ELBO evolution

Parameters

elbo_evol (*np.array*) – array containing ELBO values over iterations

figs.**clusters_fig**(*reshaped*, *num_datapoints*, *nb_clusters*, *colors_nclusters*, *cm*)

Function that produces an image that will be saved in images/

Parameters

- **reshaped** (*np.array*) – reshaped trajectories (vector format)
- **num_datapoints** (*int*) – number of datapoints in dataset
- **nb_clusters** (*list of integers*) – list of 3 elements containing the number of produced clustering
- **colors_nclusters** (*list*) – list of color palettes for presenting clusters
- **cm** (*list*) – list containing cluster memberships for each element in dataset for each predefined number of clusters

figs.**generate_colors**(*n*)

Function that generates a palette of n colors for labeling in final figure

Parameters

n (*int*) – number of clusters for generating a palette

Returns

color palette

Return type

sns.color_palette("Spectral", n)

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX