When features go missing, Bayes' comes to the rescue

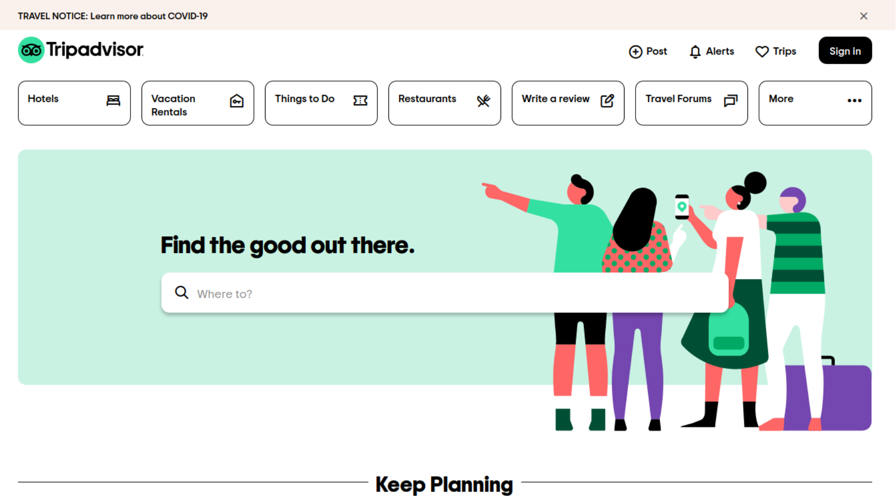
Narendra Mukherjee

Machine Learning Scientist, Tripadvisor

About myself

- Machine Learning Scientist in Tripadvisor's B2C data science team
- Work in a variety of domains from sort & recommendations to NLP
- Previously: PhD in Neuroscience from Brandeis University; studied taste processing in the brain
- Long time Bayesian
- Extras: avid bicyclist, >10k miles logged in the Boston/Massachusetts area
- narendramukherjee.github.io

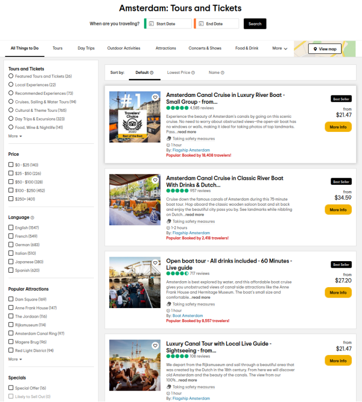
About Tripadvisor



A good product sort is essential for a smooth user experience

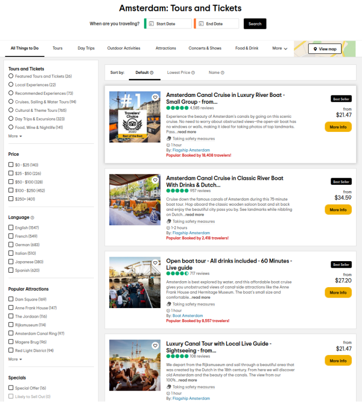
Tripadvisor

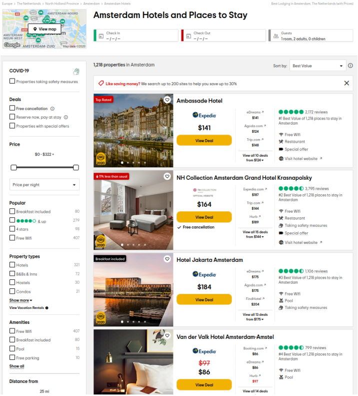
A good product sort is essential for a smooth user experience



Tripadvisor

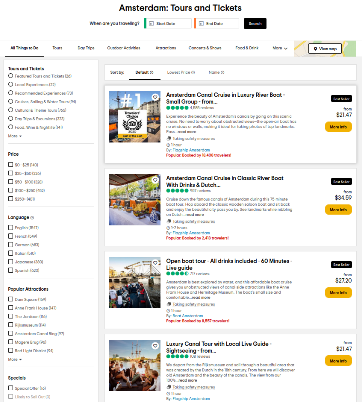
A good product sort is essential for a smooth user experience

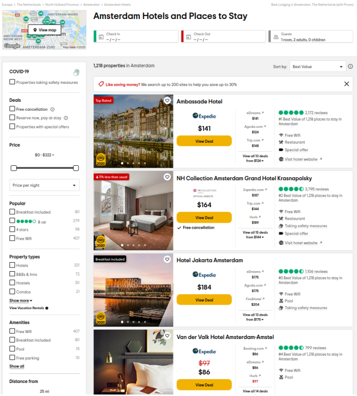


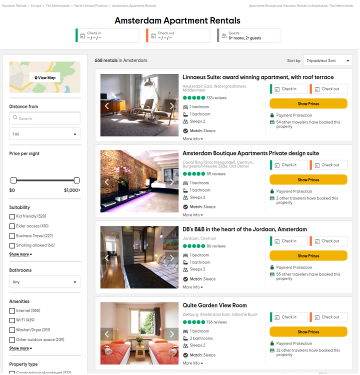


Tripadvisor

A good product sort is essential for a smooth user experience







Tripadvisor

Today's story

Today's story

- **Task:** Sorting lists of products in Tripadvisor's Experiences business

Tripadvisor

Today's story

- **Task:** Sorting lists of products in Tripadvisor's Experiences business
- **Features:** Mix of different types

Today's story

- **Task:** Sorting lists of products in Tripadvisor's Experiences business
- **Features:** Mix of different types
  - Product specific (price, star rating, CTR, CVR, etc)

Tripadvisor

Today's story

- **Task:** Sorting lists of products in Tripadvisor's Experiences business
- **Features:** Mix of different types
  - Product specific (price, star rating, CTR, CVR, etc)
  - User specific (prior browsing history, location, etc)

Tripadvisor

Today's story

- **Task:** Sorting lists of products in Tripadvisor's Experiences business
- **Features:** Mix of different types
  - Product specific (price, star rating, CTR, CVR, etc)
  - User specific (prior browsing history, location, etc)
  - Seasonality/time-of-year, etc

Tripadvisor

Today's story

- **Task:** Sorting lists of products in Tripadvisor's Experiences business
- **Features:** Mix of different types
  - Product specific (price, star rating, CTR, CVR, etc)
  - User specific (prior browsing history, location, etc)
  - Seasonality/time-of-year, etc
- **Model:** Gradient-boosted trees (XGBoost/LightGBM) that are great at capturing interactions between such diverse features

Tripadvisor

Today's story

- **Task:** Sorting lists of products in Tripadvisor's Experiences business
- **Features:** Mix of different types
  - Product specific (price, star rating, CTR, CVR, etc)
  - User specific (prior browsing history, location, etc)
  - Seasonality/time-of-year, etc
- **Model:** Gradient-boosted trees (XGBoost/LightGBM) that are great at capturing interactions between such diverse features
- **Problem:** Missing values in model features

Problem: Missing values in model features

Problem: Missing values in model features

- **How** do missing values arise in a sorting task?

Tripadvisor

Problem: Missing values in model features

- **How** do missing values arise in a sorting task?
- **Why** do they hurt performance in a sorting task?

Tripadvisor

How do missing values arise in a sorting task?

Let's sort some products!

Products	Views	Bookings	CVR	Price (USD)	# Reviews	Rating
Old Faithful	1000	200	0.2	20	500	4.5
Expensive 'n Dazzling	1000	50	0.05	200	600	4.5
New Kid on the Block	100	10	0.1	50	0	???
Out of Season	0	0	???	40	500	4.5

Tripadvisor

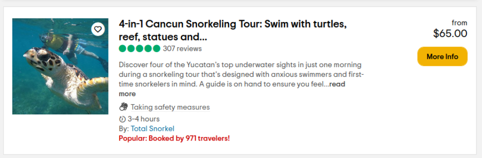
How do missing values arise in a sorting task?

Let's sort some products!

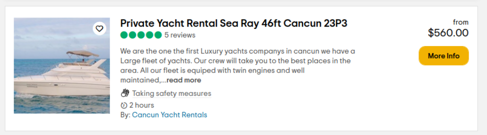
Products	Views	Bookings	CVR	Price (USD)	# Reviews	Rating
Old Faithful	1000	200	0.2	20	500	4.5
Expensive 'n Dazzling	1000	50	0.05	200	600	4.5
New Kid on the Block	100	10	0.1	50	0	???
Out of Season	0	0	???	40	500	4.5

Tripadvisor

How do missing values arise in a sorting task? (contd)



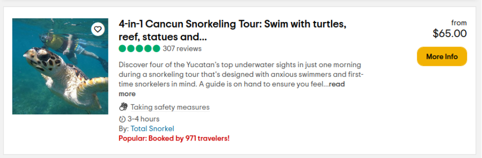
Views: 5000
Books: 971
CVR: 0.19



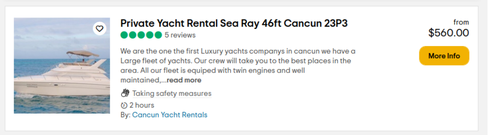
Views: 5000
Books: 50
CVR: 0.01

Tripadvisor

How do missing values arise in a sorting task? (contd)



Views: 5000
Books: 971
CVR: 0.19

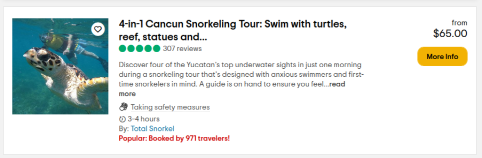


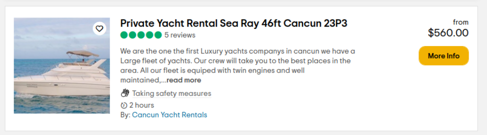
Views: 5000
Books: 50
CVR: 0.01

1. What if only 100 of the 5000 views were from users who selected a **Yacht Rentals** filter?

Tripadvisor

How do missing values arise in a sorting task? (contd)



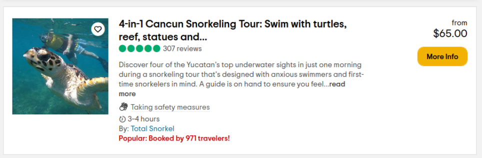


Views: 5000
Books: 971
CVR: 0.19

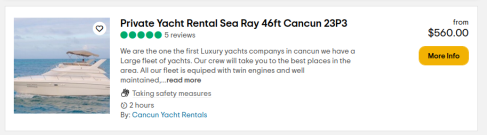
Views: 5000
Books: 50
CVR: 0.01

1. What if only 100 of the 5000 views were from users who selected a **Yacht Rentals** filter?
2. CVR$=\frac{50}{5000}=0.01$ *or* CVR$=\frac{50}{100}=0.5$ if the list is being served to a user who has selected the **Yacht Rentals** filter?

How do missing values arise in a sorting task? (contd)



Views: 5000
Books: 971
CVR: 0.19



Views: 5000
Books: 50
CVR: 0.01

1. What if only 100 of the 5000 views were from users who selected a **Yacht Rentals** filter?
2. CVR$=\frac{50}{5000}=0.01$ *or* CVR$=\frac{50}{100}=0.5$ if the list is being served to a user who has selected the **Yacht Rentals** filter?
3. CVR is missing for product × filter combinations that aren't viewed by users.

Tripadvisor

Why do missing values hurt performance in sorting task?

Why do missing values hurt performance in sorting task?

1. Training data is built using user impressions/views $\Rightarrow$ many products/user features (like filters) are missing from the training data

Why do missing values hurt performance in sorting task?

1. Training data is built using user impressions/views $\Rightarrow$ many products/user features (like filters) are missing from the training data
2. Model has to serve predictions for all combinations of product $\times$ user features $\Rightarrow$ lots of missing features at prediction time and model has to be extrapolate beyond training data

Tripadvisor

Why do missing values hurt performance in sorting task?

1. Training data is built using user impressions/views $\Rightarrow$ many products/user features (like filters) are missing from the training data
2. Model has to serve predictions for all combinations of product $\times$ user features $\Rightarrow$ lots of missing features at prediction time and model has to be extrapolate beyond training data
3. Tree-based models *cannot* extrapolate beyond their training data - they usually make constant predictions outside the regime of what they've seen during training

Tripadvisor

Why do missing values hurt performance in sorting task?

1. Training data is built using user impressions/views $\Rightarrow$ many products/user features (like filters) are missing from the training data
2. Model has to serve predictions for all combinations of product $\times$ user features $\Rightarrow$ lots of missing features at prediction time and model has to be extrapolate beyond training data
3. Tree-based models *cannot* extrapolate beyond their training data - they usually make constant predictions outside the regime of what they've seen during training
4. Bad predictions are *extremely visible* in a list of sorted recommendations - it is not enough to be doing well "on average" as in other ML tasks

Tripadvisor

So, what can we do with the missing values?

So, what can we do with the missing values?

- **Drop all rows with missing values**

Tripadvisor

So, what can we do with the missing values?

- **Drop all rows with missing values**
  - Sorting Tripadvisor Experiences with large number of features drops significant fraction of rows

Tripadvisor

So, what can we do with the missing values?

- **Drop all rows with missing values**
  - Sorting Tripadvisor Experiences with large number of features drops significant fraction of rows
- **Replace missing values with each feature's mean**

Tripadvisor

So, what can we do with the missing values?

- **Drop all rows with missing values**
  - Sorting Tripadvisor Experiences with large number of features drops significant fraction of rows
- **Replace missing values with each feature's mean**
  - The mean of a feature is dominated by extreme values, esp when % of missing values is high

So, what can we do with the missing values?

- **Drop all rows with missing values**
  - Sorting Tripadvisor Experiences with large number of features drops significant fraction of rows
- **Replace missing values with each feature's mean**
  - The mean of a feature is dominated by extreme values, esp when % of missing values is high
- **Let XGBoost/LightGBM natively handle missing values**

Tripadvisor

So, what can we do with the missing values?

- **Drop all rows with missing values**
  - Sorting Tripadvisor Experiences with large number of features drops significant fraction of rows
- **Replace missing values with each feature's mean**
  - The mean of a feature is dominated by extreme values, esp when % of missing values is high
- **Let XGBoost/LightGBM natively handle missing values**
  - GB algorithms make best split, "on average", for features with missing values, but disregard previous splits while doing so. Bad when almost all features have missing values. HousingAnywhere blog post

Tripadvisor

So, what can we do with the missing values?

- **Drop all rows with missing values**
  - Sorting Tripadvisor Experiences with large number of features drops significant fraction of rows
- **Replace missing values with each feature's mean**
  - The mean of a feature is dominated by extreme values, esp when % of missing values is high
- **Let XGBoost/LightGBM natively handle missing values**
  - GB algorithms make best split, "on average", for features with missing values, but disregard previous splits while doing so. Bad when almost all features have missing values. HousingAnywhere blog post
- **K-nearest-neighbors(KNN) imputation of missing values**

So, what can we do with the missing values?

- **Drop all rows with missing values**
  - Sorting Tripadvisor Experiences with large number of features drops significant fraction of rows
- **Replace missing values with each feature's mean**
  - The mean of a feature is dominated by extreme values, esp when % of missing values is high
- **Let XGBoost/LightGBM natively handle missing values**
  - GB algorithms make best split, "on average", for features with missing values, but disregard previous splits while doing so. Bad when almost all features have missing values. HousingAnywhere blog post
- **K-nearest-neighbors(KNN) imputation of missing values**
  - KNN finding is impractical given the strict latency requirements of a recommender system. AirBnB blog post

Tripadvisor

Sklearn *IterativeImputer*

Install User Guide API Examples More

Prev Up Next

scikit-learn 0.23.2
Other versions

Please **cite us** if you use the software.

sklearn.impute.IterativeImputer
Examples using sklearn.impute.IterativeImputer

sklearn.impute.IterativeImputer

class `sklearn.impute.IterativeImputer`(*estimator=None, *, missing_values=nan, sample_posterior=False, max_iter=10, tol=0.001, n_nearest_features=None, initial_strategy='mean', imputation_order='ascending', skip_complete=False, min_value=None, max_value=None, verbose=0, random_state=None, add_indicator=False*) [source]

Multivariate imputer that estimates each feature from all the others.

A strategy for imputing missing values by modeling each feature with missing values as a function of other features in a round-robin fashion.

Read more in the User Guide.

New in version 0.21.

Require: Initial imputation of missing values in $\{\text{model features}\}$ by, say, their means

repeat

  for $\text{feature}_i \in \{\text{model features}\}$ **do**

    $$\text{feature}_i^{\text{Not Missing}} \xrightarrow{\textbf{Train}} f(\{\text{model features}\}_{-i})$$

    $$\text{feature}_i^{\text{Missing}} \xleftarrow{\textbf{Predict}} f(\{\text{model features}\}_{-i})$$

  end for

until $|\Delta \text{feature}_i^{\text{Missing}}| \leq \text{Tolerance}$

f and Tolerance are defined by the user

Tripadvisor

Sklearn *IterativeImputer*

Require: Initial imputation of missing values in {model features} by, say, their means

repeat

  for $\text{feature}_i \in \{\text{model features}\}$ **do**

    $\text{feature}_i^{\text{Not Missing}} \xrightarrow{\textbf{Train}} f(\{\text{model features}\}_{-i})$

    $\text{feature}_i^{\text{Missing}} \xleftarrow{\textbf{Predict}} f(\{\text{model features}\}_{-i})$

  end for

until $|\Delta\text{feature}_i^{\text{Missing}}| \leq \text{Tolerance}$

Sklearn *IterativeImputer*

Require: Initial imputation of missing values in $\{\text{model features}\}$ by, say, their means

repeat

for $\text{feature}_i \in \{\text{model features}\}$ **do**

$\text{feature}_i^{\text{Not Missing}} \xrightarrow{\textbf{Train}} f(\{\text{model features}\}_{-i})$

$\text{feature}_i^{\text{Missing}} \xleftarrow{\textbf{Predict}} f(\{\text{model features}\}_{-i})$

end for

until $|\Delta\text{feature}_i^{\text{Missing}}| \leq \text{Tolerance}$

- Takes into account the relationships between the different features (only KNN is able to do this)

Sklearn *IterativeImputer*

Require: Initial imputation of missing values in $\{\text{model features}\}$ by, say, their means

repeat

for $\text{feature}_i \in \{\text{model features}\}$ **do**

$$\text{feature}_i^{\text{Not Missing}} \xrightarrow{\textbf{Train}} f(\{\text{model features}\}_{-i})$$

$$\text{feature}_i^{\text{Missing}} \xleftarrow{\textbf{Predict}} f(\{\text{model features}\}_{-i})$$

end for

until $|\Delta\text{feature}_i^{\text{Missing}}| \leq \text{Tolerance}$

- Takes into account the relationships between the different features (only KNN is able to do this)
- Provides a model (f) that can scale up the imputation process

Tripadvisor

Sklearn *IterativeImputer*

Require: Initial imputation of missing values in $\{\text{model features}\}$ by, say, their means

repeat

for $\text{feature}_i \in \{\text{model features}\}$ **do**

$$\text{feature}_i^{\text{Not Missing}} \xrightarrow{\textbf{Train}} f(\{\text{model features}\}_{-i})$$

$$\text{feature}_i^{\text{Missing}} \xleftarrow{\textbf{Predict}} f(\{\text{model features}\}_{-i})$$

end for

until $|\Delta\text{feature}_i^{\text{Missing}}| \leq \text{Tolerance}$

- Takes into account the relationships between the different features (only KNN is able to do this)
- Provides a model (f) that can scale up the imputation process
- User can modify f and Tolerance to tradeoff between imputation speed and quality

Tripadvisor

Examples from Tripadvisor's Experiences sort

Examples from Tripadvisor's Experiences sort

Naive approach: Replace missing values for CTR/CVR of product $\times$ filter combinations by -1 (lower than the minimum possible value of 0). Happens when $\#$ Views $= 0$

Examples from Tripadvisor's Experiences sort

Naive approach: Replace missing values for CTR/CVR of product $\times$ filter combinations by -1 (lower than the minimum possible value of 0). Happens when # Views $= 0$

Top 5 products for one of the Experiences' pages in London

Products	Views	Bookings	CVR
Tower of London Entrance Ticket Including Crown Jewels and Beefeater Tour	$\gg 0$	$\gg 0$	> 0
Christmas Lights Tour of London	$\gg 0$	$\gg 0$	> 0
Stonehenge, Windsor Castle, and Bath from London	$\gg 0$	$\gg 0$	> 0
Private Walking Tour of London with Brazilian Portuguese Speaking Guide	0	0	-1
Big Bus London Hop-On Hop-Off Tour	$\gg 0$	$\gg 0$	> 0

SHAP analysis shows the product's predicted score is driven up *precisely because Views=0 and CVR=-1*

Examples from Tripadvisor's Experiences sort

Principled approach: Replace missing values using *IterativeImputer*

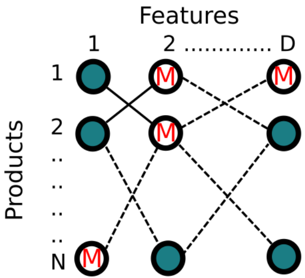
Top 5 products for one of the Experiences' pages in London

Products	Views	Bookings	CVR
Tower of London Entrance Ticket Including Crown Jewels and Beefeater Tour	$\gg 0$	$\gg 0$	> 0
Christmas Lights Tour of London	$\gg 0$	$\gg 0$	> 0
Stonehenge, Windsor Castle, and Bath from London	$\gg 0$	$\gg 0$	> 0
Big Bus London Hop-On Hop-Off Tour	$\gg 0$	$\gg 0$	> 0
Warner Bros. Studio: The Making of Harry Potter with Luxury Round-Trip Transport from London	$\gg 0$	$\gg 0$	> 0

How/why does it work? Enter Bayes!

Probabilistic understanding of imputation

Think of a probabilistic graphical model with missing values (M) as latent (hidden) variables



Two extreme views of imputation:

1. Forget about the relationships between the features, and model them as independent Gaussians $\Rightarrow$ replace missing values by MLE, aka, sample mean
2. Explicitly model the entire joint distribution of all features (Joint Modelling in statistics). Can only be done under restrictive assumptions

Tripadvisor

Can we do better and strike a middle ground?

Instead of the full joint distribution:

$$P(\text{Missing}, \text{Observed}, \text{Features}) \tag{1}$$

Model the *posterior* distribution of what we don't know (i.e, latent variables=missing values) *given* what we do know (i.e, the observed):

$$P(\text{Missing}|\text{Observed, Features}) \sim P(\text{Observed}|\text{Missing, Features}) \times P(\text{Missing, Features}) \tag{2}$$

Full machinery of Bayesian inference can be applied to this problem!

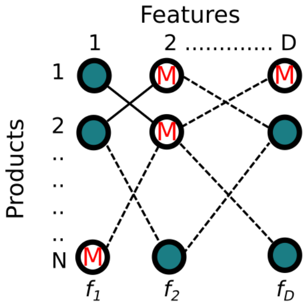
Posterior inference using MCMC

PyMC3, by default, performs posterior inference over any missing values in the data
Upcoming tutorial on this at PyMCon: Missing value tutorial

Task becomes even easier if we can sample from the conditional distributions of the features, i.e, $P(\text{Feature}_i | \{\text{Features}\}_{-i}) \rightarrow$ Gibbs sampling

Tripadvisor

Approximate Bayesian inference

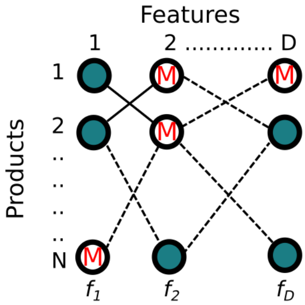


$$f_i \sim P(\text{Feature}_i | \{\text{Features}\}_{-i})$$

Ideal for expectation-maximization (EM): cycle between optimizing the "parameters" of conditional distributions, f_i, and latent variables (missing values, M) iteratively

Tripadvisor

Approximate Bayesian inference



$$f_i \sim P(\text{Feature}_i | \{\text{Features}\}_{-i})$$

Ideal for expectation-maximization (EM): cycle between optimizing the "parameters" of conditional distributions, f_i, and latent variables (missing values, M) iteratively

Guaranteed to find a local maximum of the posterior $P(\text{Missing}|\text{Observed, Features})$

Approximate Bayesian inference: different flavors of EM

1. **Variational inference ("fully" Bayesian):** Maintain the conditional distributions over the parameters of f_i and for the latent variables M at every iteration - all estimates are averaged wrt these conditionals

Tripadvisor

Approximate Bayesian inference: different flavors of EM

1. **Variational inference ("fully" Bayesian):** Maintain the conditional distributions over the parameters of f_i and for the latent variables M at every iteration - all estimates are averaged wrt these conditionals
2. **"Standard" EM ("partially" Bayesian):** Maintains conditional distributions only for the latent variables, not for the parameters of f_i. eg: Gaussian Mixture Modelling (GMM)

Approximate Bayesian inference: different flavors of EM

1. **Variational inference ("fully" Bayesian):** Maintain the conditional distributions over the parameters of f_i and for the latent variables M at every iteration - all estimates are averaged wrt these conditionals
2. **"Standard" EM ("partially" Bayesian):** Maintains conditional distributions only for the latent variables, not for the parameters of f_i. eg: Gaussian Mixture Modelling (GMM)
3. **Iterated Conditional Modes (ICM):** Set all random variables to the MAP/MLE estimate at each iteration, no distributions are maintained. eg: K-Means clustering

Tripadvisor

IterativeImputer: ICM in the probabilistic graph of missing values

Require: Initial imputation of missing values in $\{\text{model features}\}$ by, say, their means

repeat

    for $\text{feature}_i \in \{\text{model features}\}$ **do**

        $\text{feature}_i^{\text{Not Missing}} \xrightarrow{\textbf{Train}} f(\{\text{model features}\}_{-i})$

        $\text{feature}_i^{\text{Missing}} \xleftarrow{\textbf{Predict}} f(\{\text{model features}\}_{-i})$

    end for

until $|\Delta\text{feature}_i^{\text{Missing}}| \leq \text{Tolerance}$

IterativeImputer: ICM in the probabilistic graph of missing values

Require: Initial imputation of missing values in $\{\text{model features}\}$ by, say, their means (Setting a prior over the missing values)

repeat

  for $\text{feature}_i \in \{\text{model features}\}$ **do**

    $\text{feature}_i^{\text{Not Missing}} \xrightarrow{\textbf{Train}} f(\{\text{model features}\}_{-i})$

    $\text{feature}_i^{\text{Missing}} \xleftarrow{\textbf{Predict}} f(\{\text{model features}\}_{-i})$

  end for

until $|\Delta\text{feature}_i^{\text{Missing}}| \leq \text{Tolerance}$

Tripadvisor

IterativeImputer: ICM in the probabilistic graph of missing values

Require: Initial imputation of missing values in {model features} by, say, their means (Setting a prior over the missing values)

repeat

  for $\text{feature}_i \in \{\text{model features}\}$ **do**

    $\text{feature}_i^{\text{Not Missing}} \xrightarrow{\textbf{Train}} f(\{\text{model features}\}_{-i})$ (M step: Set the parameters of the conditionals, f_i to their MLE/MAP)

    $\text{feature}_i^{\text{Missing}} \xleftarrow{\textbf{Predict}} f(\{\text{model features}\}_{-i})$

  end for

until $|\Delta\text{feature}_i^{\text{Missing}}| \leq \text{Tolerance}$

IterativeImputer: ICM in the probabilistic graph of missing values

Require: Initial imputation of missing values in {model features} by, say, their means (Setting a prior over the missing values)

repeat

for $\text{feature}_i \in \{\text{model features}\}$ **do**

$\text{feature}_i^{\text{Not Missing}} \xrightarrow{\textbf{Train}} f(\{\text{model features}\}_{-i})$ (M step: Set the parameters of the conditionals, f_i to their MLE/MAP)

$\text{feature}_i^{\text{Missing}} \xleftarrow{\textbf{Predict}} f(\{\text{model features}\}_{-i})$ ("Hard" E step: Set the latent variables to the MLE/MAP, given the parameters)

end for

until $|\Delta \text{feature}_i^{\text{Missing}}| \leq \text{Tolerance}$

Tripadvisor

IterativeImputer: ICM in the probabilistic graph of missing values

Require: Initial imputation of missing values in {model features} by, say, their means (Setting a prior over the missing values)

repeat

  for $\text{feature}_i \in \{\text{model features}\}$ **do**

    $\text{feature}_i^{\text{Not Missing}} \xrightarrow{\textbf{Train}} f(\{\text{model features}\}_{-i})$ (M step: Set the parameters of the conditionals, f_i to their MLE/MAP)

    $\text{feature}_i^{\text{Missing}} \xleftarrow{\textbf{Predict}} f(\{\text{model features}\}_{-i})$ ("Hard" E step: Set the latent variables to the MLE/MAP, given the parameters)

  end for

until $|\Delta\text{feature}_i^{\text{Missing}}| \leq \text{Tolerance}$

Known in the statistics community as Multiple Imputation with Chained Equations (MICE)

StackOverflow, MICE

Tripadvisor

A word of caution

- ICM can get stuck in local optima - multiple random restarts are needed
- ML: pick the best (by MLE etc) fit amongst the multiple runs. Statistics: use the restarts to get confidence intervals on the estimates ("multiple" in MICE)
- We got reasonably good results in the missing value imputation problem with even just 1 run

Tripadvisor

Thank you!!

Visit narendramukherjee.github.io