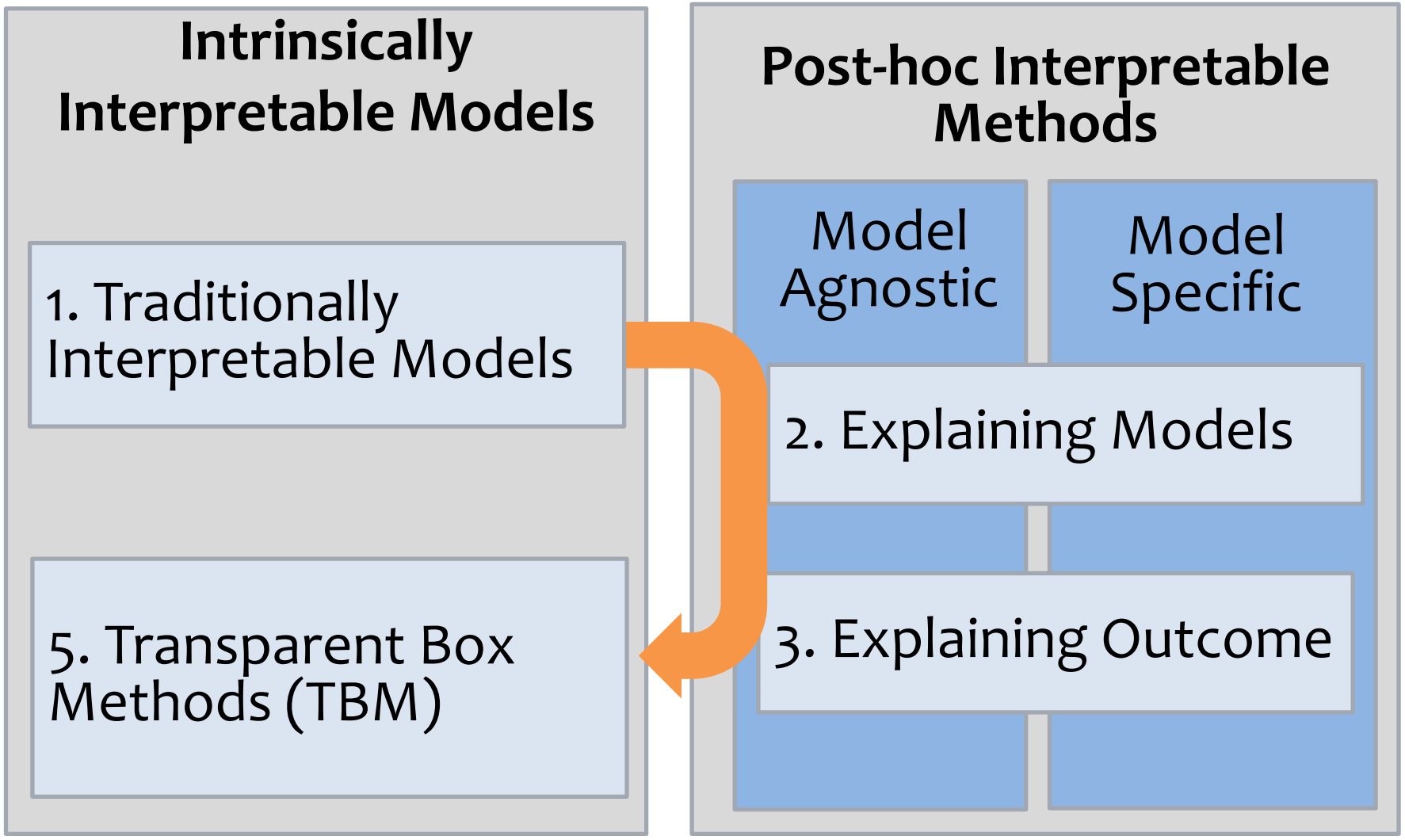
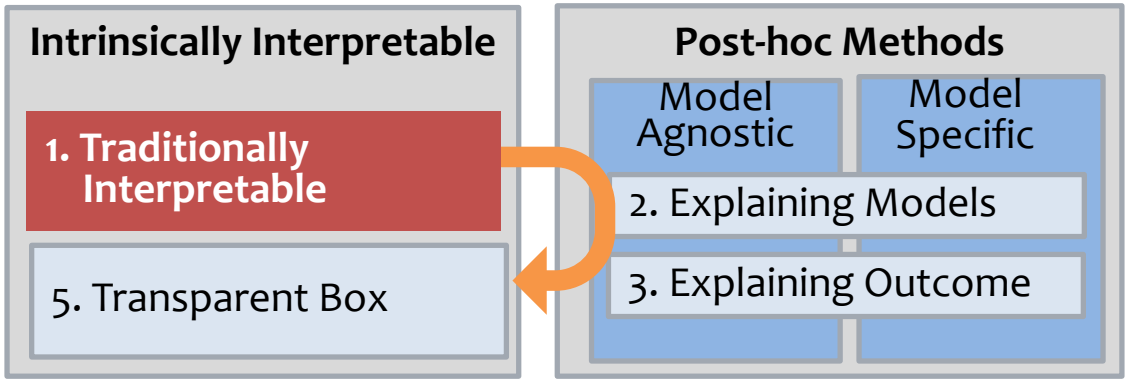
Interpretable ML Content Overview





Part 2: Interpretable ML Overview

- **Traditional Intrinsically Interpretable Models**

Most of the contents comes from

- "Interpretable Machine Learning *A Guide for Making Black Box Models Explainable." by Christoph Molnar 2018.*
- "Interpretable Machine Learning: The fuss, the concrete and the questions" B. Kim & F. Doshi-Velez, Tutorial, ICML 2017

[C. Molnar 2018]

Traditional Interpretable Models (TIM)

- Pros:
  - Often easier to understand how the model works
  - Often under express the complexity of the system
- Cons:
  - Often has lower accuracy compared to other ML
  - May not be the interpretability that you seek for
- Examples
  - Linear Models
  - Decision Tree
  - Decision Rules
  - RuleSets

Note: Some people argue that there are **no intrinsically interpretable models**

[C. Molnar 2018]

Linear Models

- Linear models learn linear (and therefore monotonic) relationships between the features and the target.

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i = \boldsymbol{\beta}^{\mathbf{T}} \boldsymbol{x_i} + \epsilon_i$$

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(y_i - \boldsymbol{\beta}^{\mathbf{T}} \boldsymbol{x_i}\right)^2$$

Given all other features stay the same.

- Numerical: Unit increase of x_k increases the expectation for y by β_k
- Categorical: A change from x_k's reference level to the other category increases the expectation for y by β_k

* Interpretation of a weight **can be unintuitive** because it depends on all other features

[C. Molnar 2018]

Interpretable Measures for Linear Models

- Measuring the **total variance** of your target outcomes explained by the model
  - i.e. Adjusted R-square given p # of features, n # of instances, correct labels y_i, and estimated labels $\hat{y}_i$:

$$\bar{R}^2 = R^2 - \frac{p}{n-p-1}(1-R^2)$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

SSE

SST

- Measuring importance of a feature in linear regression
  - i.e. T-statistics:

$$t_{\hat{\beta}} = \frac{\hat{\beta}}{std(\hat{\beta})/\sqrt{n}}$$

Feature weight

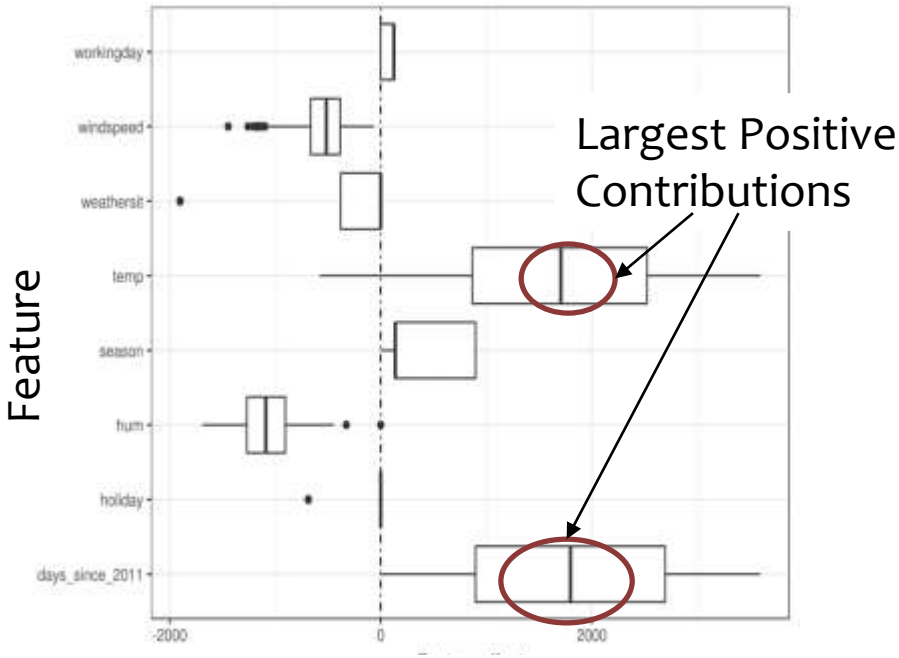
Standard error of feature weight

[C. Molnar 2018]

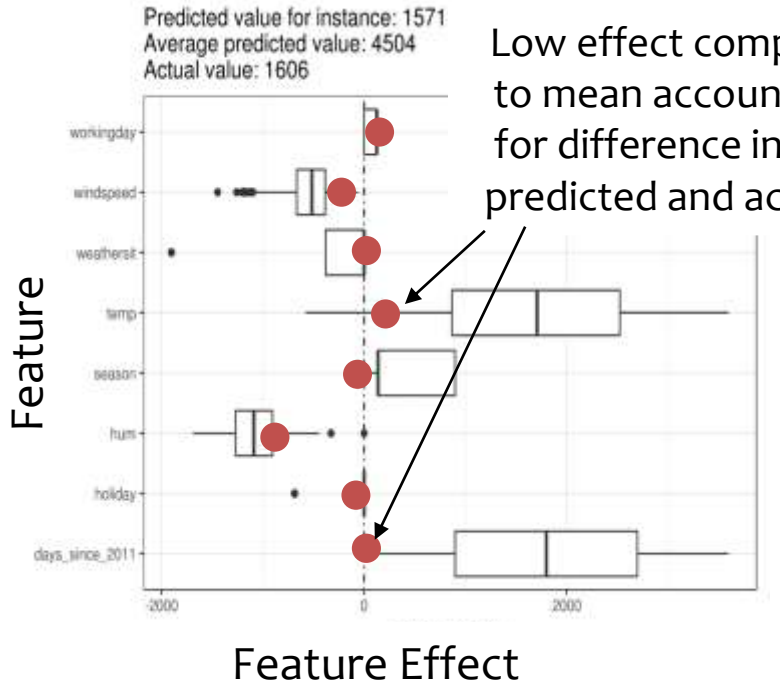
Effect Plot for Linear Model

Effects plot show how much the combination of a weight and a numerical feature contributes to the predictions. Feature Effect: $e_{ij} = w_i x_j$

Explaining Model



Explaining Single Predictions



Figures from [C. Molnar 2018]

[C. Molnar 2018]

Linear Models-Coding Categorical Features

- The choice of encode a categorical feature influences the interpretation of the β-weights.
- Effect coding example: a feature of three categories [A, B, C]

Feature matrix:

Intercept (represents the overall mean): β_0

$$\begin{matrix} A: \\ B: \\ C: \end{matrix} \begin{pmatrix} 1 & -1 & -1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \quad \begin{matrix} \text{Effect: } \beta_0 - (\beta_1 + \beta_2) \\ \text{Effect: } \beta_0 + \beta_1 \\ \text{Effect: } \beta_0 + \beta_2 \end{matrix}$$

* More encoding methods in:
https://stats.idre.ucla.edu/r/library/r-library-contrast-coding-systems-for-categorical-variables/
http://heidiseibold.github.io/page7/

[C. Molnar 2018]

Sparse Linear Models

- Linear models can be made more interpretable by making model **sparse**
- Simple solutions:
  - LASSO:

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(y_i - \boldsymbol{\beta}^{\mathbf{T}} \boldsymbol{x_i}\right)^2 + \lambda \|\boldsymbol{\beta}\|_1$$

  - Feature Selection:
    - Forward selection: iteratively add features to the model
    - Backward selection: iteratively delete features from the model

[C. Molnar 2018]

Logistic Regression

- Goal: model the **probability** of random variable Y being 0 or 1.

$$h_{\boldsymbol{\beta}}(\boldsymbol{x}_i) = \frac{1}{1-e^{-\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x_i}}} = \Pr(y_i = 1|\boldsymbol{x}_i;\boldsymbol{\beta})$$

$$1 - h_{\boldsymbol{\beta}}(\boldsymbol{x}_i) = \frac{e^{-\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x_i}}}{1-e^{-\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x_i}}} = \Pr(y_i = 0|\boldsymbol{x}_i;\boldsymbol{\beta})$$

- Interpretation: weights don't affect the probability linearly, but are squeezed through the logistic function -> reformulating odds and odds ratio

$$odds_i = \frac{h_{\boldsymbol{\beta}}(\boldsymbol{x}_i)}{1 - h_{\boldsymbol{\beta}}(\boldsymbol{x}_i)} = \exp\left(-\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x_i}\right)$$

Change of $x_{i,j}$ by +1 unit, changes the **odds ratio** by:

$$\frac{odds_{i,x_j+1}}{odds_i} = \exp\left(-\beta_j\left(x_{i,j}+1\right) - \beta_j\left(x_{i,j}\right)\right) = \boxed{\exp(\beta_j)}$$

[C. Molnar 2018]

Generalized Linear Model (GLM)

- **Problem:** Linear regression (LR) model assumes outcome follows a Gaussian. What if they don't?
- **Solution:** GLM extends the LR to model various types of outcomes using **link function g** and expected mean E_Y on the **assumed distribution**

$$g\left(E_Y(y_i|\boldsymbol{x_i})\right) = \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x_i}$$

- **Interpretation:** assumed distribution and link function determines how the estimated feature weights are interpreted.
  - EX> logistic regression is a GLM that assumes Bernoulli distribution and use logistic function as the link function.

[C. Molnar 2018]

Generalized Additive Model (GAM)

- **Problem:** What if the relationship between the features and y is not linear?

- **Solutions:**
  - Transform the feature (e.g. logarithm)
  - Categorization of the feature
  - GAMs that use regression splines

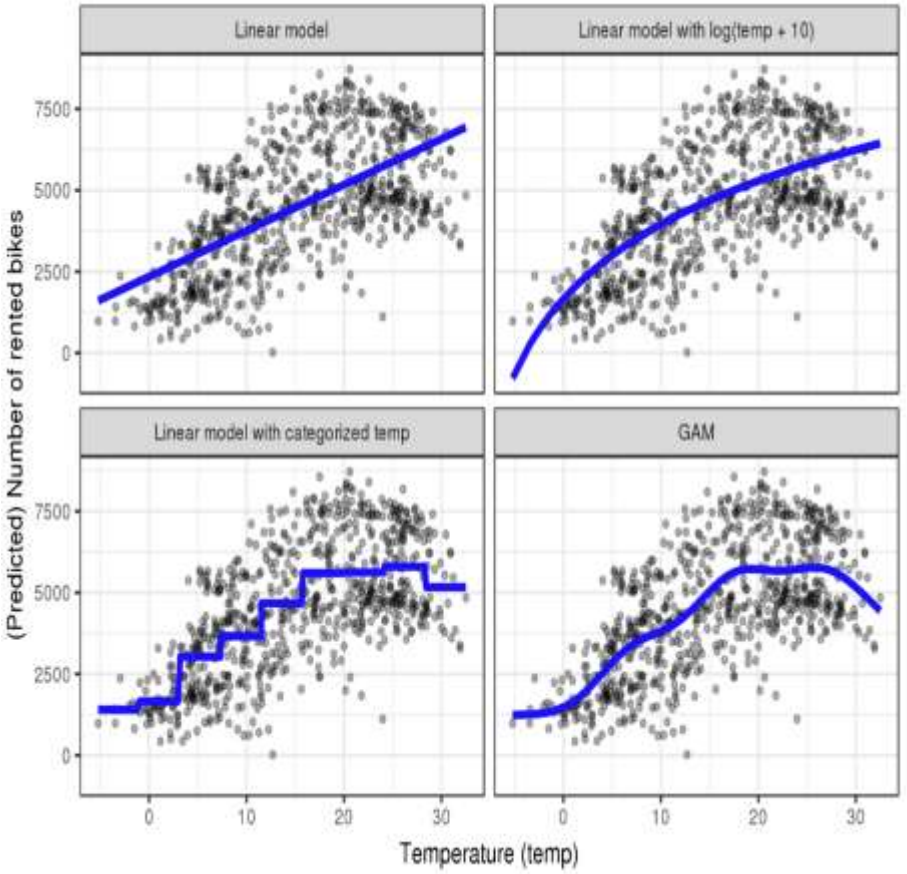


Figure 4.12 from [C. Molnar 2018]

[C. Molnar 2018]

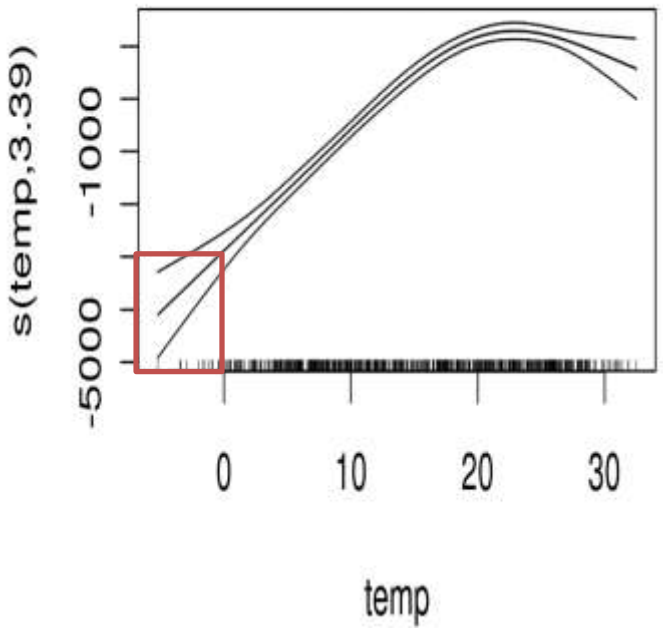
Generalized Additive Model (GAM)

- GAMs assume that the outcome can be modeled by a sum of arbitrary functions of each feature.

$$g\left(E_Y(y_i|\boldsymbol{x_i})\right) = \beta_o + f_1(x_{i1}) + \cdots + f_p(x_{ip})$$

and using weighted sum of "spline functions" to learn the nonlinear function.

Figure 4.14 from [C. Molnar 2018]



- **Interpretation via visual inspection:** Splines are usually centered around the mean prediction, so a point on the curve is the difference to the mean prediction.
  - Ex> At temp 0, predicted # is 3000 lower than the average prediction.

Decision Trees

Decision trees are non-linear models that can address features that interacting with each other

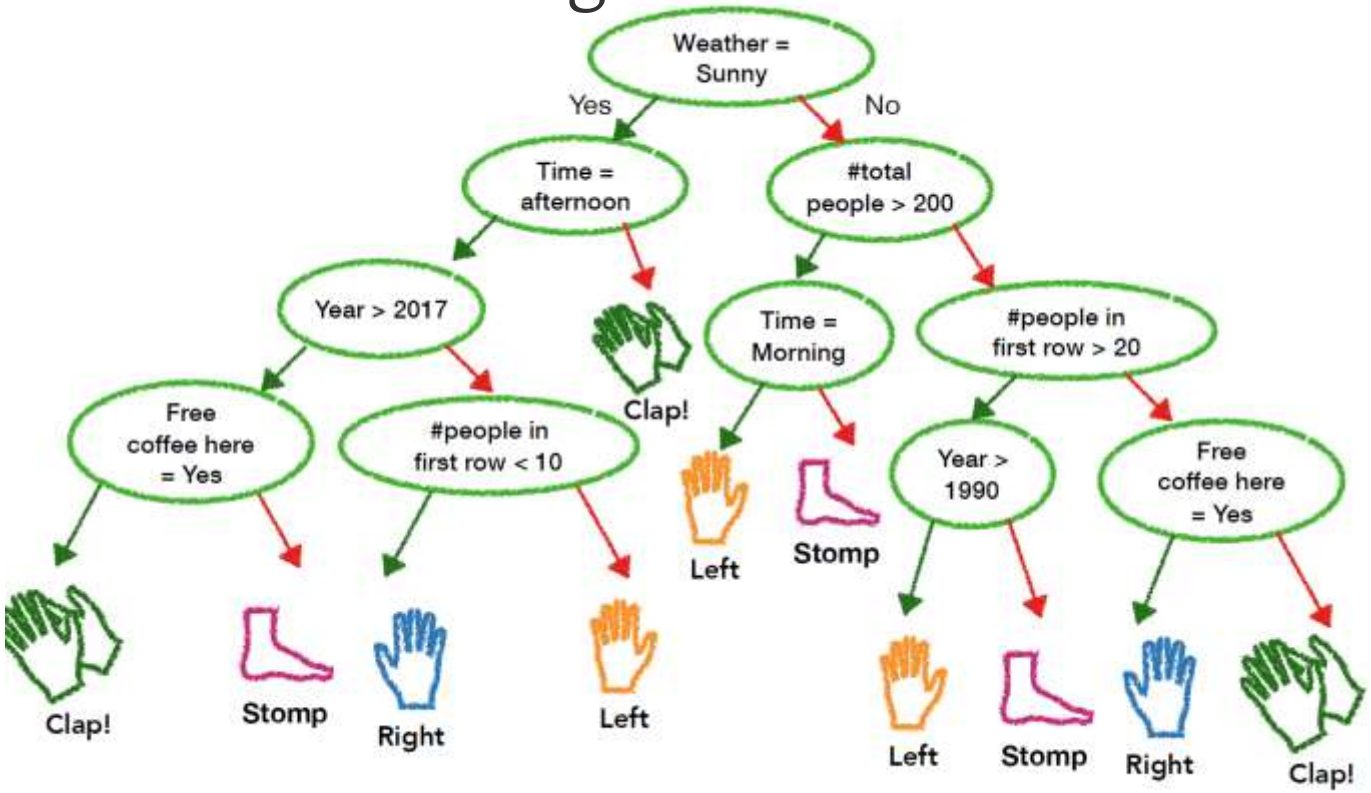


Figure from [Been CVPR18]

* Can be made more interpretable by pruning

[C. Molnar 2018]

Decision Trees Interpretation

- Reading the model:
  - "If feature x is [smaller/bigger] than threshold c AND …, then the predicted outcome is $\hat{y}_{\text{leafnode}}$."
- Importance of a feature in an instance $\boldsymbol{x_i}$
  - Go through all the splits for which the feature was used and add up how much it has improved the predictions in the child nodes compared to the parent node and scaled via **tree decomposition**
  - Tree decomposition:

$$\hat{f}(\boldsymbol{x_i}) = \bar{y} + \sum_{d=1}^{D} \text{split. contrib}(d, x)$$
$$= \bar{y} + \sum_{j=1}^{p} \text{feat. contrib}(j, x)$$

Contribution at the **root**

Decision Rules

More human language like using IF-THEN statement

If (sunny and hot)	then	go swim
Else if (sunny and cold)	then	go ski
Else if (wet and weekday)	then	go work
Else if (free coffee)	then	attend tutorial
Else if (cloudy and hot)	then	go swim
Else if (snowing)	then	go ski
Else if (New Rick and Morty)	then	watch TV
Else if (paper deadline)	then	go work
Else if (hungry)	then	go eat
Else if (tired)	then	watch TV
Else if (advisor might come)	then	go work
Else if (code running)	then	watch TV
Else	then	go work

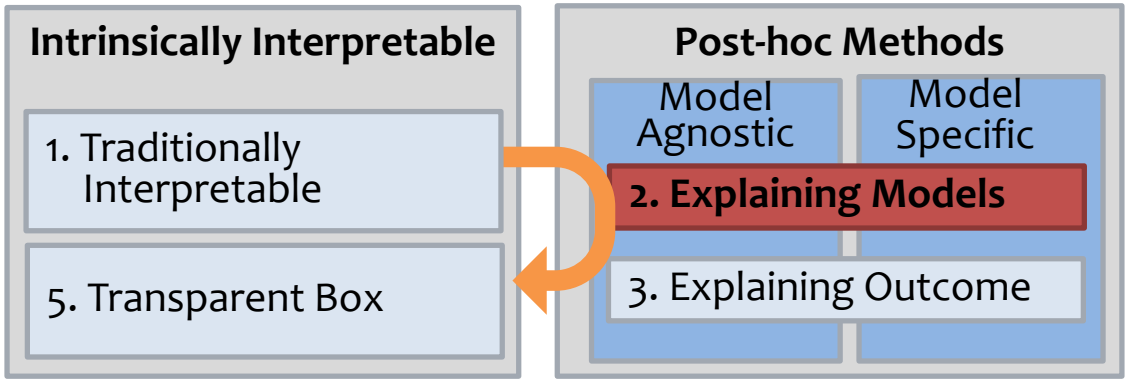
Figure from [Been CVPR18]

Rule Sets

Another human language like

IF (sunny and hot) OR (cloudy and hot) OR (sunny and thirsty and bored) OR (bored and tired) OR (thirty and tired) OR (code running) OR (friends away and bored) OR (sunny and want to swim) OR (sunny and friends visiting) OR (need exercise) OR (want to build castles) OR (sunny and bored) OR (done with deadline and hot) OR (need vitamin D and sunny) OR (just feel like it)
THEN go to beach
ELSE work

Figure from [Been CVPR18]



Part 2: Interpretable ML Overview

- **Post-hoc Interpretable Methods**
  - **Explaining Models**
  - Explaining Outcome
  - Model Inspection

Most of the contents comes from

- "Interpretable Machine Learning *A Guide for Making Black Box Models Explainable." by Christoph Molnar 2018.*

What is Explaining Models?

Explaining the overall behavior of a learning machine globally over all data.

- Guidotti et al. [2018] in their review distinguishes explaining models as two different problems of but we will consider both as model explanation problem. (shown in the following slide)

  [Guidotti, et al. *ACM Comput. Surv.* 2018.]

- Typical components of explaining models
  - Feature based – ex> find important features
  - Example based – ex> find prototypes

What is Explaining Models?

Guidotti et al. [2018] definitions:

"Given a black box predictor *b* and a set of instances *X*, the ***model explanation problem*** consists in finding an explanation $E \in \mathcal{E}$, belonging to a human-interpretable domain $\mathcal{E}$, through an interpretable global predictor $c_g = f(b, X)$ derived from the black box *b* and the instances *X* using some process $f(\cdot, \cdot)$. An explanation $E \in \mathcal{E}$ is obtained through c_g,if $\mathrm{E} = \varepsilon_g(c_g, X)$ for some explanation logic $\varepsilon_g(\cdot, \cdot)$, which reasons over c_g and X." [Guidotti et al. 2018]

"Given a black box predictor *b* and a set of instances *X*, the ***model inspection problem*** consists in providing a (visual or textual) representation $r = f(b, X)$ of some property of b using some process $f(\cdot, \cdot)$." [Guidotti et al. 2018]

Partial Dependence Plot [J. H. Friedman 2001]

Prediction function f(x) is fixed at a few values of the chosen features x_s and averaged over the other features x_C.

$$\hat{f}_{x_S}(x_S) = E_{x_C}\left[\hat{f}(x_S, x_C)\right]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\hat{f}(x_S, x_{Ci})$$

Shows the marginal effect of a feature on the predicted outcome

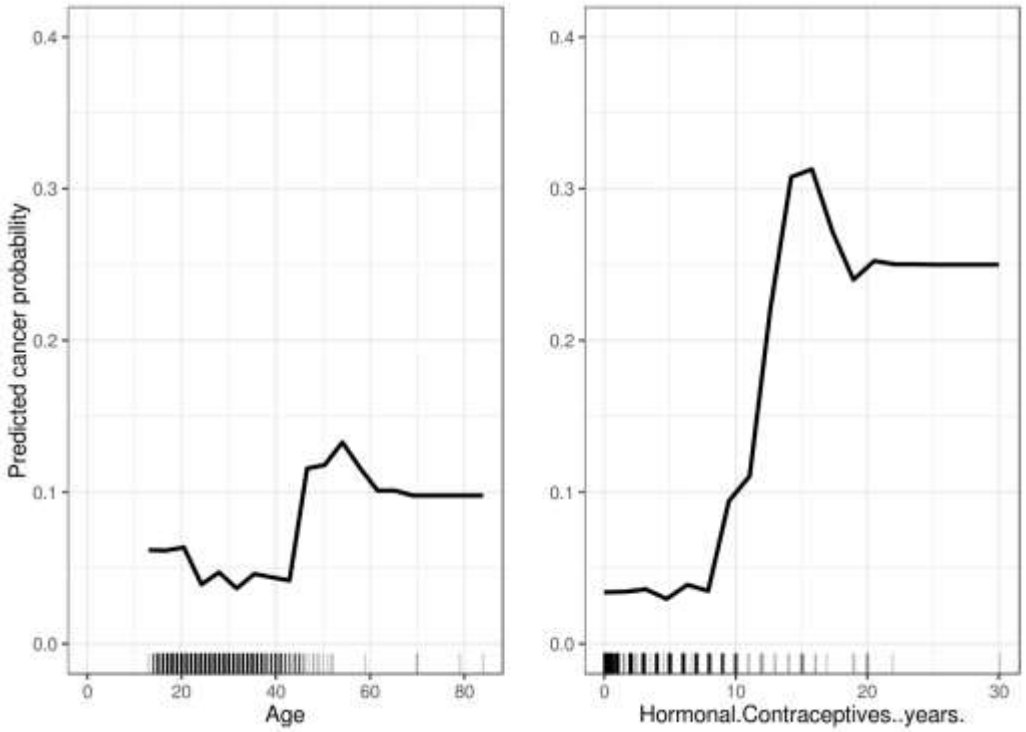
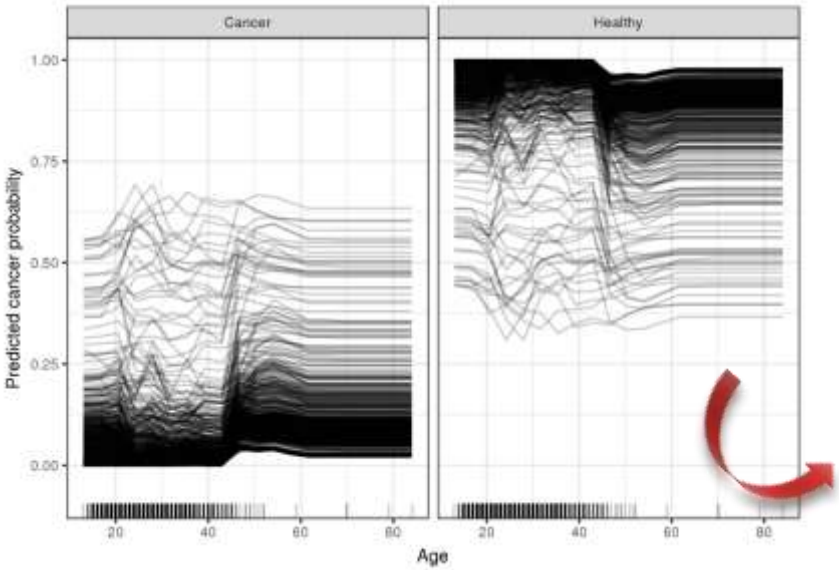


Figure from [C. Molnar, 2018]

Individual Conditional Expectation (ICE)

[Goldstein et al. 2013]

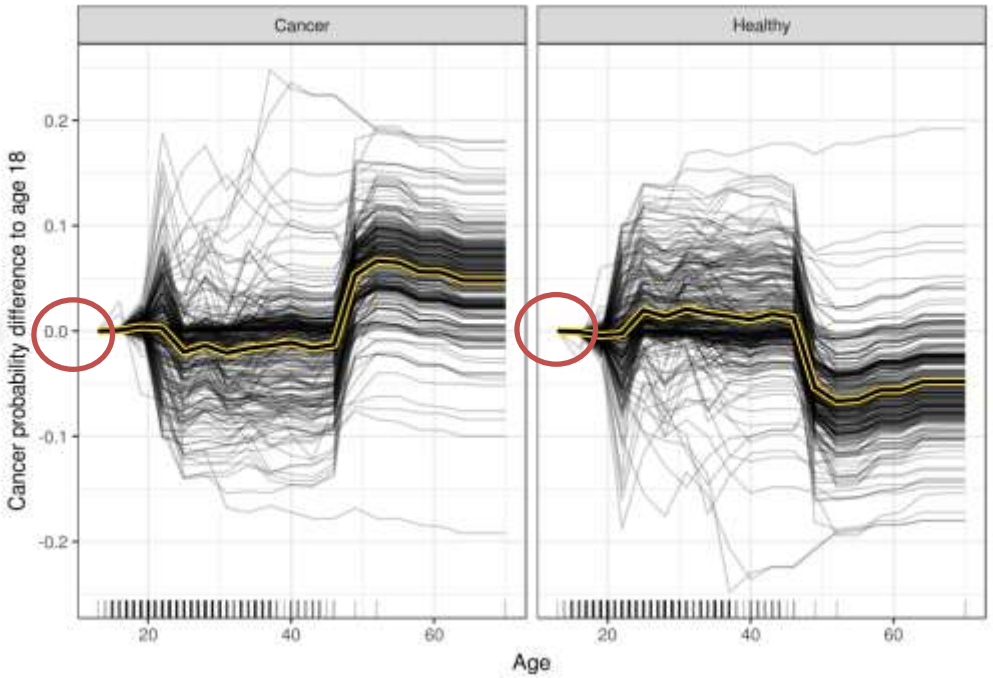
Draw one line *per instance*, representing how the instance's prediction changes when the feature changes



Figures from [C. Molnar, 2018]

Centered ICE plot

Centered at youngest observed age (13)



A PDP is the average of the lines of an ICE plot.

Feature Interaction

i.g., H-statistics estimate the *strength of interaction feature* x_j *to all other features* $\mathbf{x}_{-j}$ by measure how much of the ***variation*** *of the predicted* outcome depends on the interaction of the features.

[Friedman & Popescu 2008]

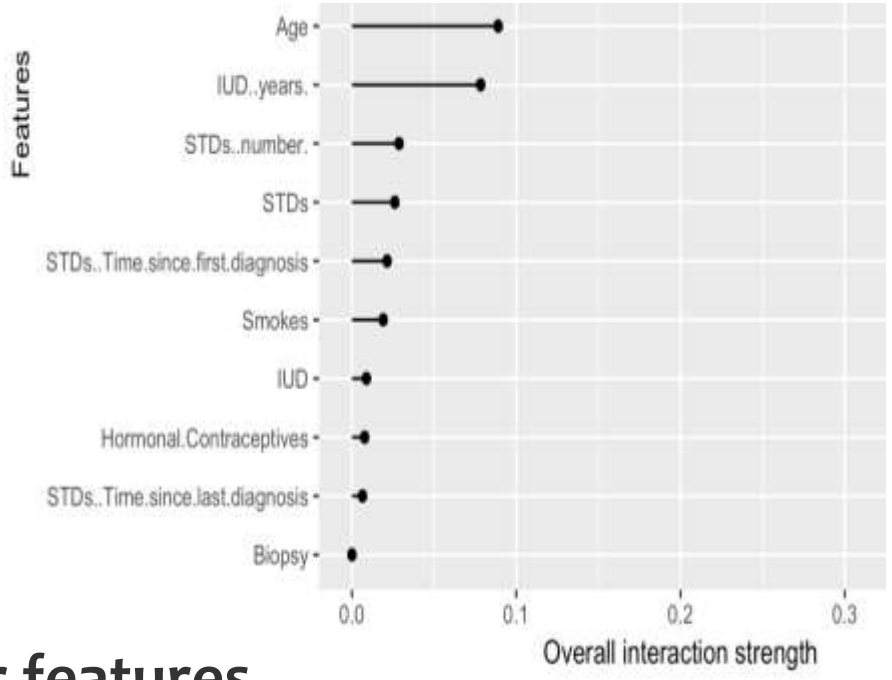


Figure from [C. Molnar, 2018]

H-statistics feature x_j vs all other features

$$H_j^2 = \sum_{i=1}^{n} \left[\hat{f}(x^{(i)}) - PD_j(x_j^{(i)}) - PD_{-j}(x_{-j}^{(i)}) \right] / \sum_{i=1}^{n} \hat{f}^2(x^{(i)})$$

where PD is partial dependence function

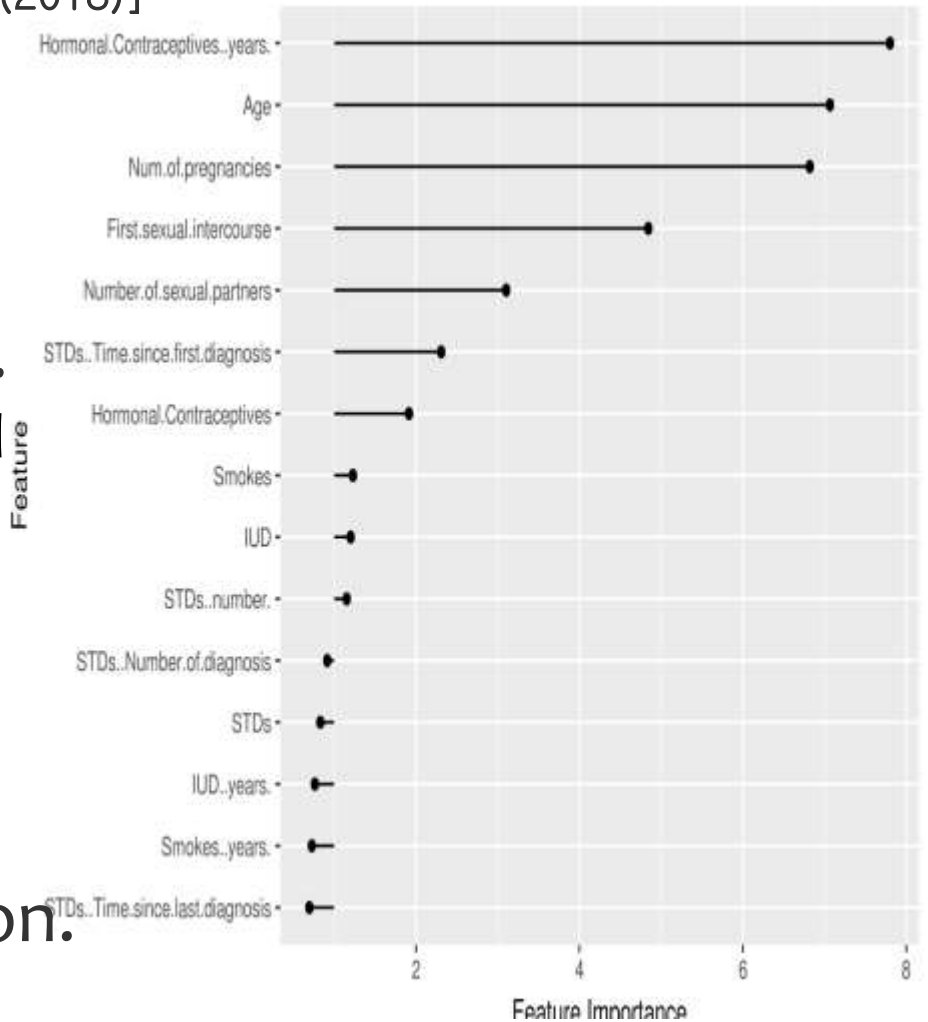
Feature Importance

[Breiman (2011) Fisher, Rudin, and Dominici (2018)]

Measures the increase of model error when the feature's information is destroyed (value permuted).

- A feature is "important" model error increases
- A feature is "unimportant" if model error unchanged

Requires labeled data and model with error computation.



[C. Molnar, 2018]

[C. Molnar, 2018]

Computing Feature Importance

Permutation feature importance algorithm based on
[Breiman 2011, Fisher et al. 2018]

Input: Trained model $\hat{f}$, feature matrix X, target vector Y, error measure $\mathrm{L}(\mathrm{Y}, \hat{Y})$

1. Estimate the original model error $\mathrm{e}_{\mathrm{org}}(\hat{f}) = \mathrm{L}(\mathrm{Y}, \hat{f}(X))$ (e.g rmse)
2. **For each** feature $j \in 1, \ldots, p$ **do**
   // break the association between X_j and Y by permuting value of X_j
3. Generate feature matrix X_{perm_j} by permuting feature X_j in X.
4. Estimate error $\mathrm{e}_{\mathrm{perm}} = \mathrm{L}\left(\mathrm{Y}, \hat{f}(X_{perm_j})\right)$ based on the predictions on the permuted data.
5. Calculate permutation feature importance $\mathrm{FI}_{\mathrm{j}} = \mathrm{e}_{\mathrm{perm}(\hat{f})} / e_{org}(\hat{f})$
   // alternatively, use difference $\mathrm{FI}_{\mathrm{j}} = \mathrm{e}_{\mathrm{perm}(\hat{f})} - e_{org}(\hat{f})$
6. Sort variables by descending *FI*

[C. Molnar, 2018]

Global Surrogate Models

Outputs a intrinsically interpretable models that is trained to approximate the predictions of a black box model

[Breiman 2011; Fisher, Rudin, and Dominici 2018]

e.g., Random forest model fit to decision tree model

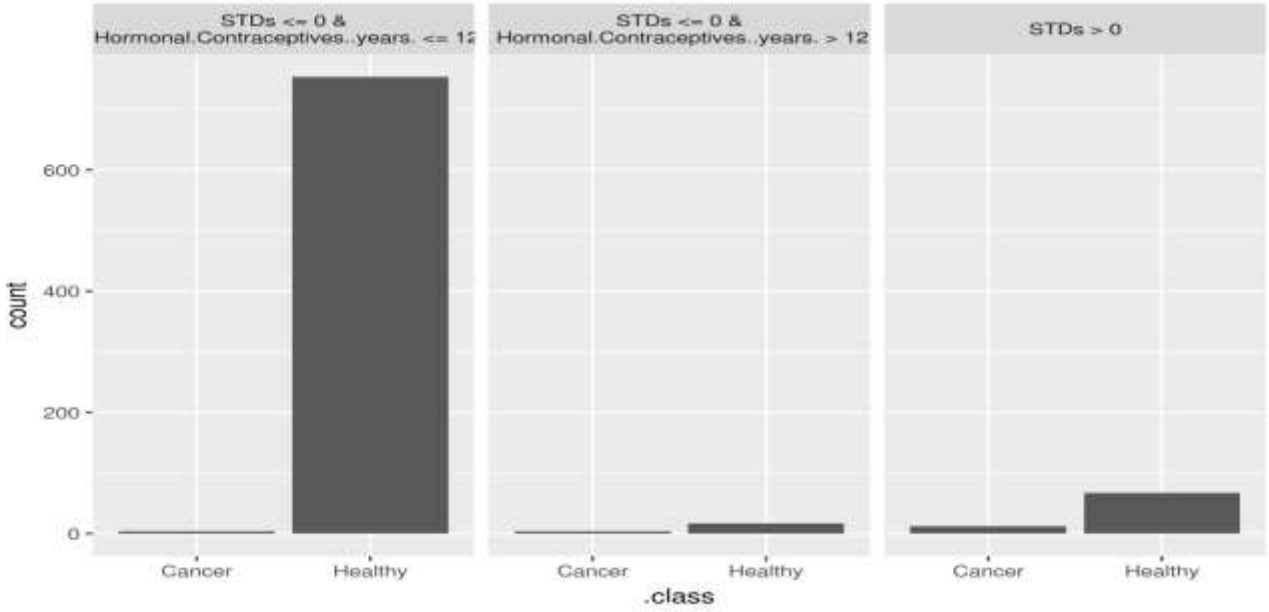


Figure from [C. Molnar, 2018]

[C. Molnar, 2018]

Global Surrogate Models

Typical Surrogate Models Used

- Decision Tree Modeling
- Decision Rules (surveyed in [Andrews et al. 1995] for neural net)
- Examples:
  - Explaining neural nets
    - *Trepan* [Graven et al. 1996] **enrich data** using NN as oracle for generating decision tree & *DecTex* [Boz 2002] uses pruning in addition to *Trepan* for generating simpler tree.
    - **Generate prototypes** [Krishnan et al 1999] or evolve tree [Johansson et al 2009] via genetic programming to generate small decision trees from small prototype dataset
  - Explaining tree ensembles
    - **Tree combination** using tree similarity measures [Chipman et al 1998]
    - **Data enrichment** + decision tree learning [Domingos et al 1998, Gibbons et al 2013, Zhou et al. 2016]
    - Generate **tree prototype** [Tan et al. 2016]

[C. Molnar, 2018]

Simple Steps for Generating a GSM

1. **Choose a dataset X.**
2. For the dataset X, get the predictions $\hat{Y}$ of the black box model.
3. Choose an interpretable model (linear model, decision tree, …).
4. Train the interpretable model on the dataset X and predictions $\hat{Y}$.
5. You now have a surrogate model.
6. Measure how well the surrogate model replicates the prediction of the black box model.
7. Interpret / visualize the surrogate model.

[C. Molnar, 2018]

Global Surrogate Models

- Strategies used for global surrogate model construction from base black box model
  - **Enrich data** by using base ML for generating label data.
  - **Generate prototype** data from the base ML data to generate simpler model.

- How well surrogate replicate the base model can be measured by **R squared measure** that evaluates the percentage of variance that is captured by the interpretable model

$$R^2 = 1 - \frac{SSE}{SST} = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i^{*} - \hat{y})^2}{\sum_{i=1}^{n}\left(\hat{y}_i - \overline{\hat{y}_i}\right)^2}$$

where $\hat{y}_i^{*}$ is the surrogate's prediction for the ith instance and $\hat{y}$ is the prediction of the black box model

[C. Molnar, 2018]

Example-Based Explanations

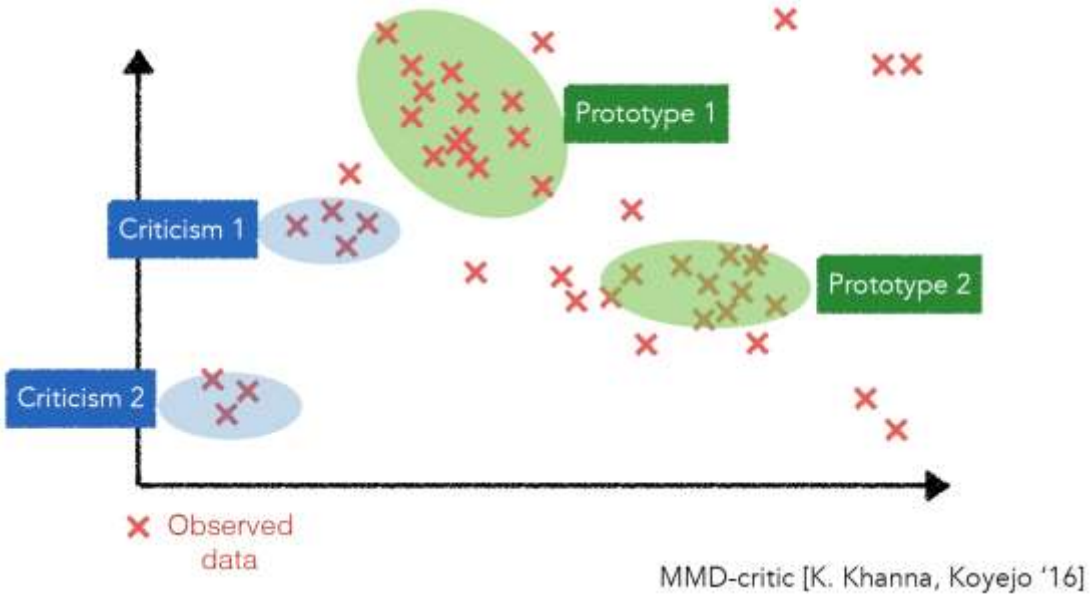
Selects **particular instances of the dataset** to explain the behavior of machine learning models or to explain the underlying data distribution

- Works well for data that have structure or when the number of features are few

Examples	What they explain
Prototypes and criticisms	model
Influential instances	model/outcome
Counterfactual explanations	outcome
Adversarial examples	outcome

Prototypes and Criticisms

- **Prototypes** are a selection of representative instances from the data
- **Criticisms** are instances that are not well represented by those prototypes.



MMD-critic [K. Khanna, Koyejo '16]

EX> KNN

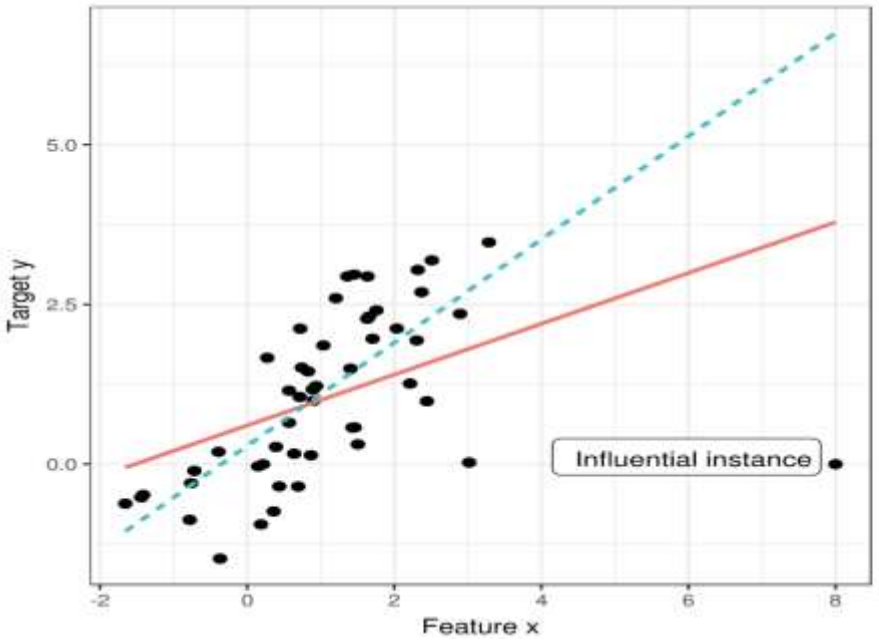
[C. Molnar, 2018]

Influential Instances

A data is '**influential**' when deleting it changes the parameter or prediction of a model.

* Useful for improving the model.

Figure form [C. Molnar, 2018]

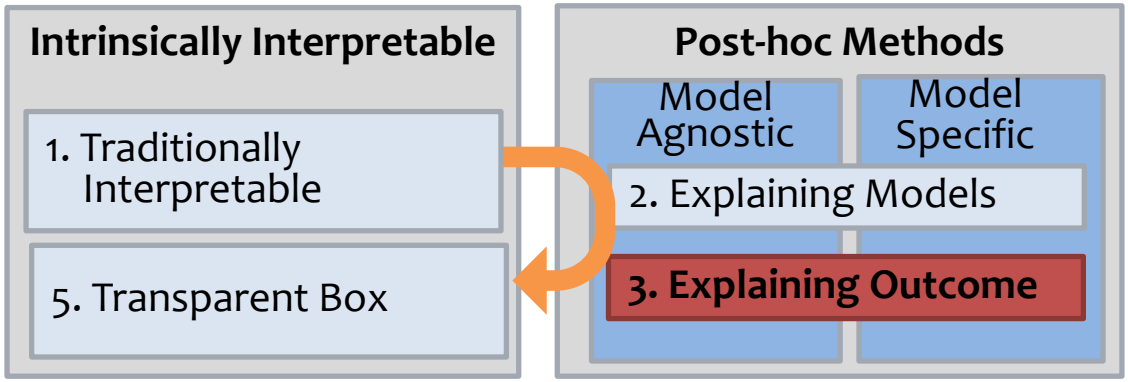


- Approaches:
  - Deletion diagnostics
    - EX> Cook's distance
  - Influence functions
    - Koh and Liang (2017)

    Utilize Hessian of the loss function

Reference

- R. Guidotti, et al. “A Survey of Methods for Explaining Black Box Models,” ACM Comput. Surv., vol. 51, no. 5, pp. 1–42, Aug. 2018.
- Friedman, Jerome H. 2001. “Greedy Function Approximation: A Gradient Boosting Machine.” Annals of Statistics. JSTOR, 1189–1232.
- Breiman, Leo. 2001. “Random Forests.” Machine Learning 45 (1). Springer: 5–32.
- Fisher et al. 2018. “Model Class Reliance: Variable Importance Measures for any Machine Learning Model Class, from the ‘Rashomon’ Perspective.” http://arxiv.org/abs/1801.01489.
- Cook, R. Dennis. “Detection of influential observation in linear regression.” Technometrics 19.1 (1977): 15-18.↩
- Koh, P. W., & Liang, P. (2017). Understanding Black-box Predictions via Influence Functions. http://arxiv.org/abs/1703.04730



Part 2: Interpretable ML Overview

- **Post-hoc Interpretable Methods**
  - Explaining Models
  - **Explaining Outcome**

Most of the contents comes from

- R. Guidotti, et al. "A Survey of Methods for Explaining Black Box Models," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1–42, Aug. 2018.

Explaining Outcome (EO)

"Given a black box predictor b and an instance x, the ***outcome explanation problem*** consists in finding an explanation $e \in \mathcal{E}$, belonging to a human-interpretable domain $\mathcal{E}$, through an interpretable local predictor $c_l = f(b, x)$ derived from the black box b and the instance x using some process $f(\cdot, \cdot)$. An explanation $e \in \mathcal{E}$ is obtained through c_l, if $e = \varepsilon_l(c_l, x)$ for some explanation logic $\varepsilon_l(\cdot, \cdot)$, which reasons over c_l and x." [Guidotti et al. 2018]

- Explains the outcome such as decisions or predictions made by the learning machine on an **instance**.
- Typical types:
  - Explain by returning (set of) **features** of the instance
  - Explain by returning associated **rules** of the instance

Shapley Value Explanations

Shapley value tells us how to assign **feature effects** $\phi_{ij}(\hat{f})$ to features **for single prediction** depending on their **contribution** towards the total output generated by learning model $\hat{f}$

* Original Shapely value calculation for game theory: Find each player's marginal contribution by simulating the arrival sequence and taking the average marginal contribution.

Shapley (1953)

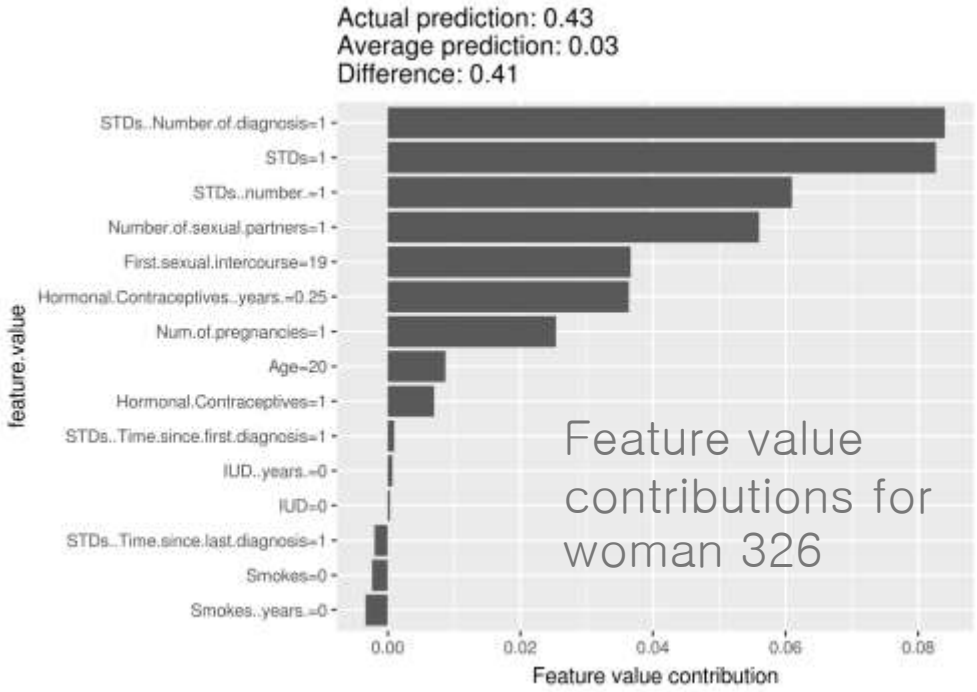


Figure form [C. Molnar, 2018]

[C. Molnar, 2018]

Computing Shapley Value

The **Shapley value of a feature value** x_{ij} is it's contribution to the payed outcome, weighted and summed over all possible feature value combinations:

$$\phi_{ij}(val) = \sum_{S \subseteq \{x_{i1},\dots,x_{ip}\} \{x_{ij}\}} \frac{|S|!\,(p-|S|-1)!}{p!} \left(val\left(S \cup \{x_{ij}\}\right) - val(S)\right)$$

[Lundberg &Lee NIPS'17]

where S is a subset of the features used in the model, x_i, is the feature values of instance i , and p is the number of features. $val_{x_i}(S)$ is the prediction for feature values in set S, marginalized over features not in S:

$$val_{x_i}(S) = \int \hat{f}(x_{i1}, \dots, x_{ip}) d\mathrm{P}_{X_i \notin S} - E_X(\hat{f}(\mathrm{X}))$$

If there are multiple features not in S, you actually do multiple integrations, for each features not in S.

[C. Molnar, 2018]

Local Surrogate Models

Method for fitting local, interpretable models that can **explain single prediction** of any black-box machine learning model by training on **variation of a instance** of interest and the model's output.

Figure from [lime R package]

Post-hoc | Model-Agnostic | Explaining Data | Feature-based

[Ribeiro et al, KDD'16]

LIME as Local Surrogate Model

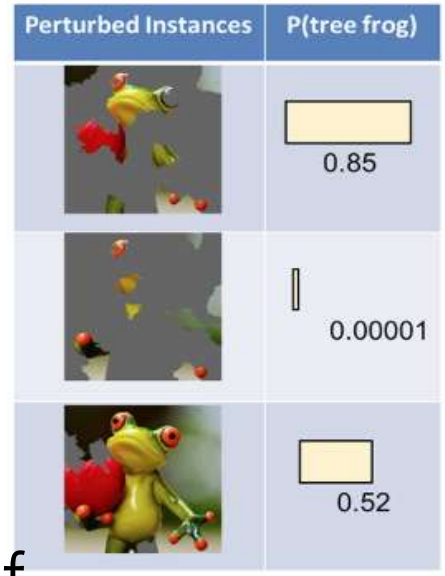
Local **I**nterpretable **M**odel-Agnostic **E**xplanations

1. Transforming input into **interpretable components.**

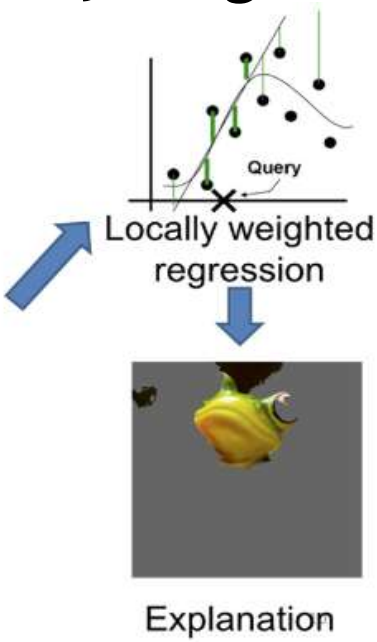
Original Image

Interpretable Components

P(tree frog) = 0.54



3. Learn a **simple (linear) model** on this data set, which is **locally weighted**



Explanation

2. Generate a data set of perturbed instances by turning some **components "off"**

4. Present the components with **highest positive weights** as an explanation.

Figure Sources: Marco Tulio Ribeiro, Pixabay.

Anchor-LIME

A model-agnostic system that explains the behavior of complex models with high-precision rules called **anchors**, representing local, "sufficient" conditions for predictions

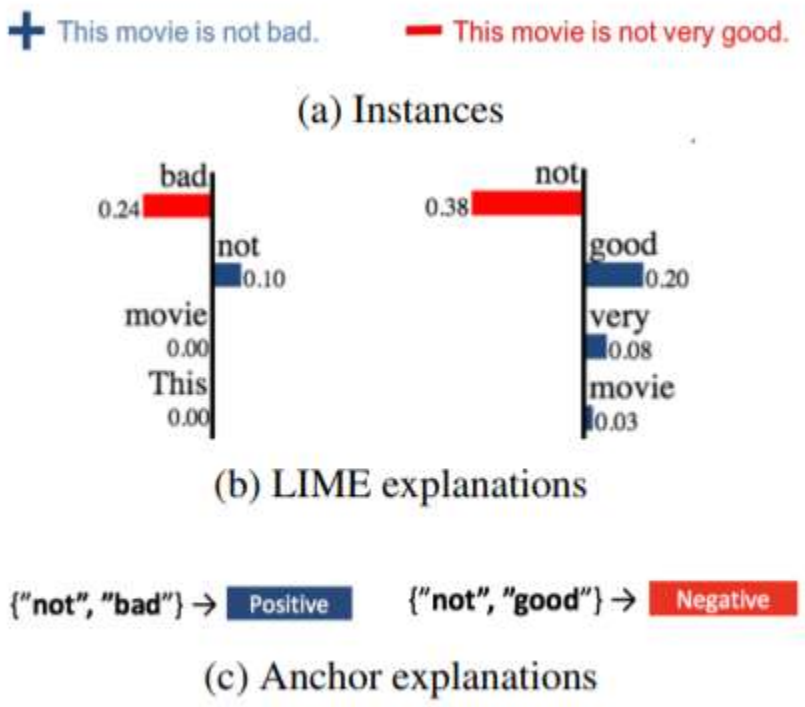


Figure 1: Sentiment predictions, LSTM

Figure from Ribeiro et al AAAI'18

Local Rule-based Explanations (LORE)

First learns local interpretable predictor on a synthetic neighborhood generated by a **genetic algorithm.**

Then it derives from the **logic** of the local a **decision rule,** which explains the reasons of the decision; and a set of **counterfactual rules,** suggesting the changes in the instance's features that lead to a different outcome.

Algorithm 1: $LORE(x, b)$

Input : x - instance to explain, b - black box, N - # of neighbors
Output: e - explanation of x

1 $G \leftarrow 10$; $pc \leftarrow 0.5$; $pm \leftarrow 0.2$; // init. parameters
2 $Z_{=} \leftarrow GeneticNeigh(x, fitness^{x}_{=}, b, N/2, G, pc, pm)$ // generate neigh.
3 $Z_{\neq} \leftarrow GeneticNeigh(x, fitness^{x}_{\neq}, b, N/2, G, pc, pm)$ // generate neigh.
4 $Z \leftarrow Z_{=} \cup Z_{\neq}$; // merge neighborhoods
5 $c \leftarrow BuildTree(Z)$; // build decision tree
6 $r = (p \rightarrow y) \leftarrow ExtractRule(c, x)$; // extract decision rule
7 $\Phi \leftarrow ExtractCounterfactuals(c, r, x)$; // extract counterfactuals
8 **return** $e = \langle r, \Phi \rangle$;

Counterfactual Explanation

- Tells how an instance has to change to significantly change its prediction
- Describes the **smallest change** to the feature values that changes the prediction to a **predefined output.**

"Starting from instance X that has output Y, change features A and B from X to get a counterfactual instance X' that outputs desired output Y' "

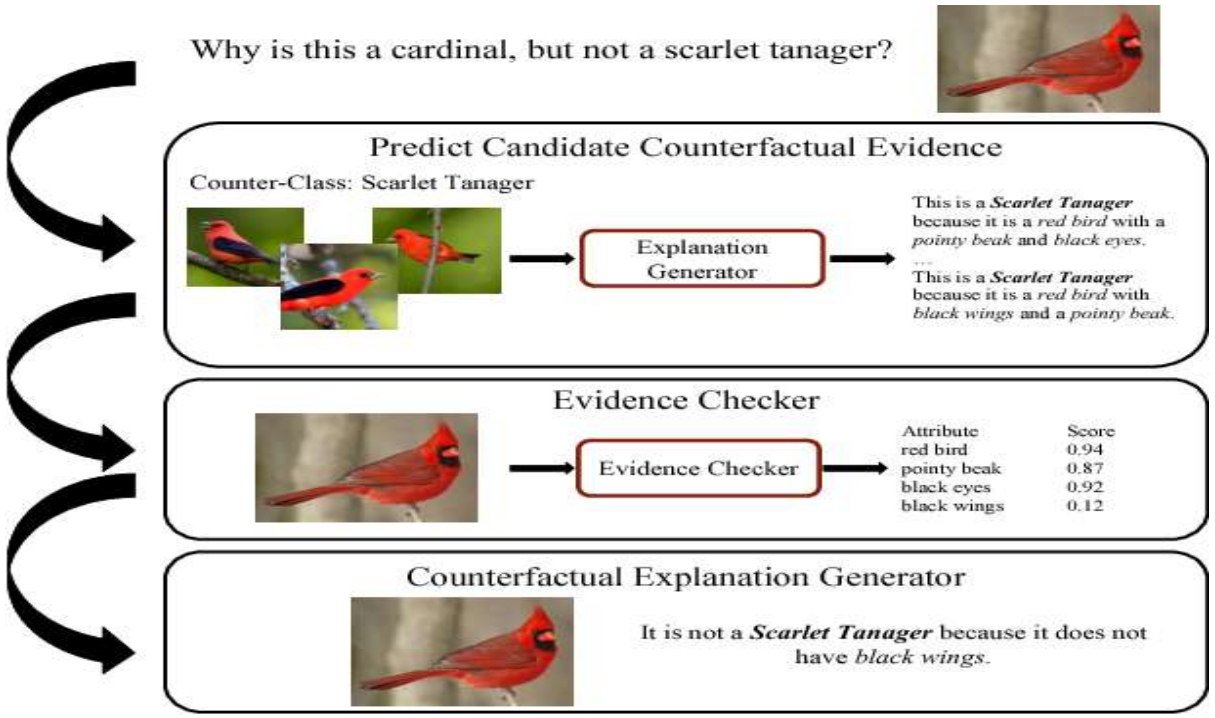


Figure from Hendricks et al 2018

Post-hoc | Model-Agnostic | Explaining Outcome | Example-based

[C. Molnar, 2018]

Computing Counterfactual Explanations

[Wachter et. al 2017]

An approach for generating counterfactuals:

Loss: $$L(x, x', y', \lambda) = \lambda \cdot \left(\hat{f}(x') - y'\right)^2 + d(x, x')$$

Optimization: $$\arg\min_{x'} \max_{\lambda} L(x, x', y', \lambda)$$

Input: Given an instance x to be explained, the desired outcome y', a tolerance ϵ, and a (low) initial value for λ.

1. Sample a random instance x' as initial counterfactual
2. //Optimize the loss with the initially sampled counterfactual as starting:
   While $\left|\hat{f}(x') - y'\right| > \epsilon$:
   Increase λ.
   Optimized the loss with the current counterfactual as staring point
   Return the counterfactual x' that minimizes the loss
3. Repeat steps 1-2 and return the list of counterfactual or the one that minimizes the loss

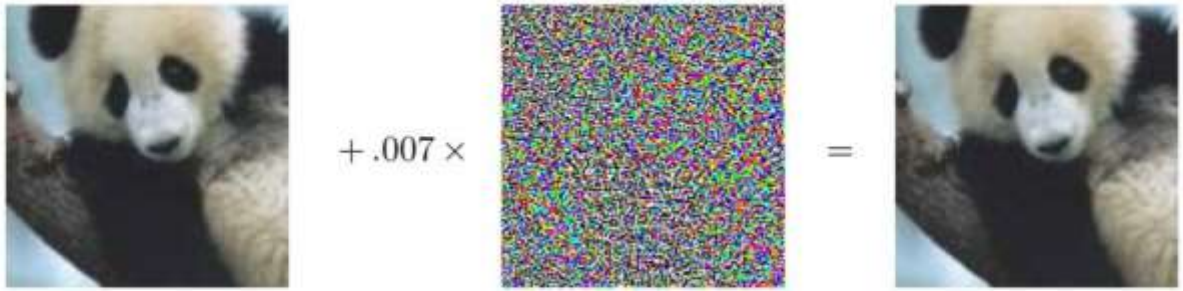
[C. Molnar, 2018]

Adversarial Examples

- **Counterfactuals** used to fool machine learning models

$$x^* = argmin_{\tilde{x}} \|x - \tilde{x}\|^2 \;\; s.t. f(x) \neq f(\tilde{x})$$

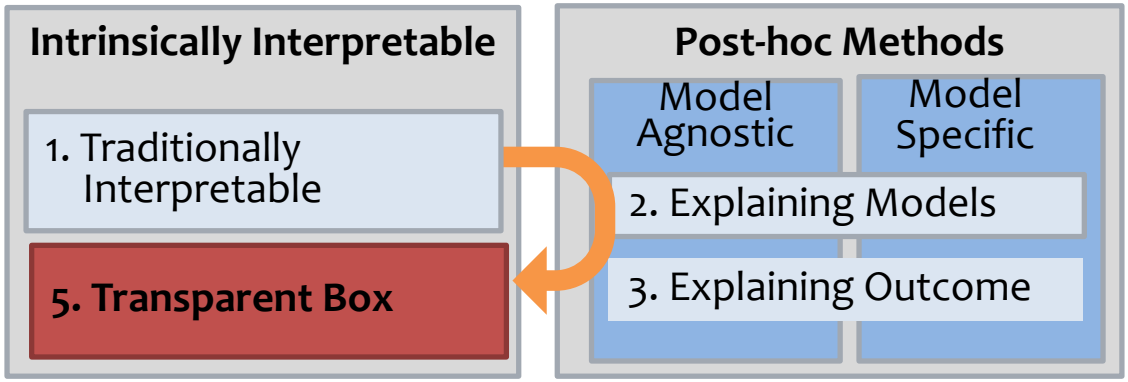
- Very similar to counterfactual examples but the aim is not to interpret a model but to deceive it



Goodfellow et al. 2015

Reference

- Shapley, Lloyd S. 1953. “A Value for N-Person Games.” Contributions to the Theory of Games 2 (28): 307–17.
- Strumbelj et al.. 2014. “Explaining prediction models and individual predictions with feature contributions.” Knowledge and Information Systems 41 (3): 647–65.
- Lundberg & Lee. 2016. “An unexpected unity among methods for interpreting model predictions,” no. Nips: 1–6.
- Ribeiro et al. 2016. Why should i trust you?: Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1135-1144). ACM.
- Wachter et al. (2017). Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR, (1), 1–47.
- Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. 2018. Local rule-based explanations of black box decision systems. arXiv preprint arXiv:1805.10820 (2018).



Part 2: Interpretable ML Overview

- **Transparent Box Models**

Transparent Box Models

- Methods that interpreted the models or predictions specific to the prediction model or application.
- Usually does this by
  - Making **intrinsically interpretable methods more interpretable**.
  - **Integrating interpretable component** to learning machine
- Examples:
  - Matrix/Tensor Sparse Decomposition
  - Simplified Rule Learning

[Wagstaff et al AAAI'13]

DEMUD: Matrix Factorization Based

- Motivated by need of methods in the era of large scientific data sets for:
  - Automatically **prioritize data** for review.
  - Make decisions that they can **understand** and trust

Proposed

- DEMUD: Discovery through Eigen basis Modeling of Uninteresting Data
- Uses principal components modeling and reconstruction error to prioritize data.

[Wagstaff et al AAAI'13]

DEMUD Example

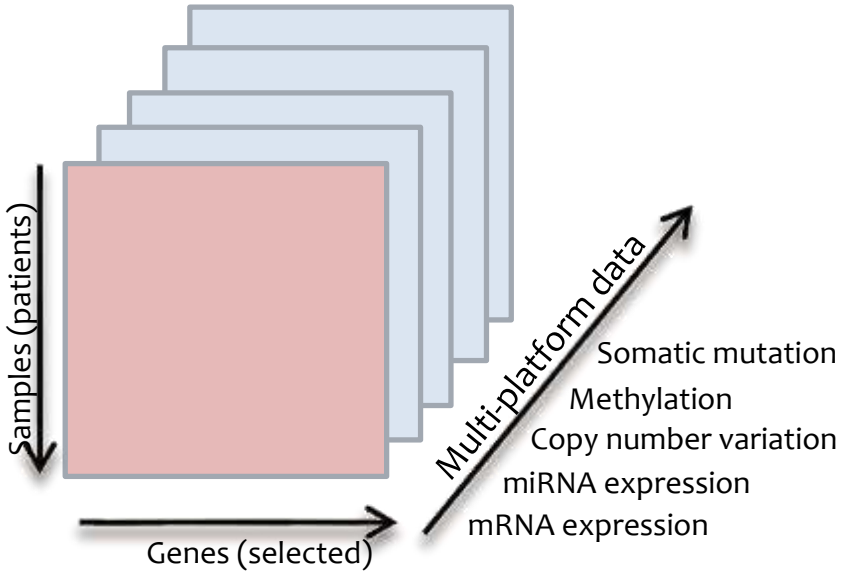
DEMUD result on Glass data set, expressed as residuals in original units (percent composition). Positive (negative) values are higher (lower) than expected;

Selection	Class (proportion)	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
1	container (6%)	−0.001	−1.60	−0.86	+0.79	−2.80	+5.40	−0.24	−0.88	−0.01
2	building window, non-float (36%)	0.000	−0.72	0.00	−2.00	−0.32	−6.10	+9.20	0.00	+0.24
3	tableware (4%)	+0.005	+4.60	0.00	−2.10	+5.00	−4.50	−2.90	0.00	−0.07
5	headlamp (14%)	−0.002	−2.80	−0.56	−0.41	+3.00	+1.30	−0.05	−0.46	−0.06
6	building window, float (33%)	+0.003	−0.28	+4.00	−0.50	−0.80	−1.80	−0.32	−0.37	−0.05
8	vehicle window, float (8%)	+0.002	+0.43	+2.90	−0.43	−1.20	−0.93	−0.02	−0.49	−0.07

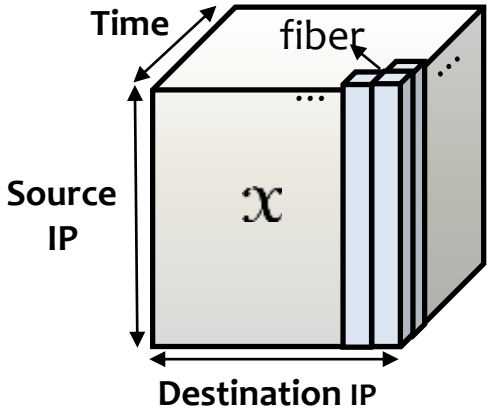
[Wagstaff et al AAAI'13]

Background: Tensor* as Data Structure

How can we represent multi-mode data?



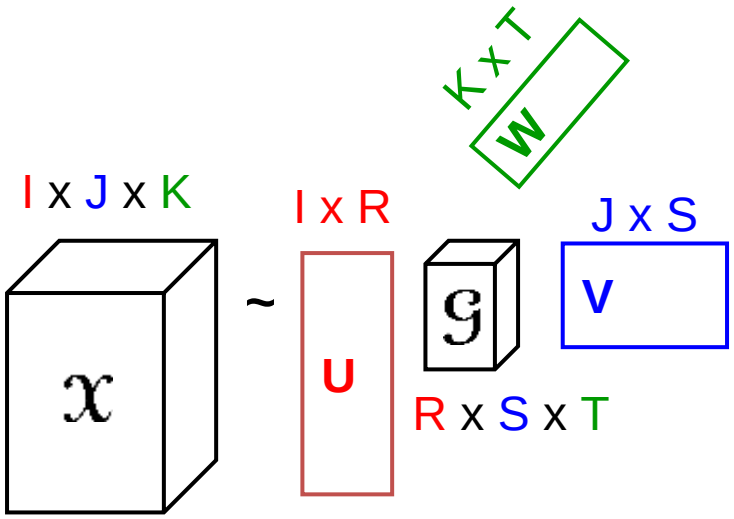
Multi-platform Bio-data
(patient – gene – platform)



Network traffic data
(src IP – dst IP - time)

***Tensor**: a multi-dim array. 1D is array, 2D matrix and 3D cube

Tucker Decomposition

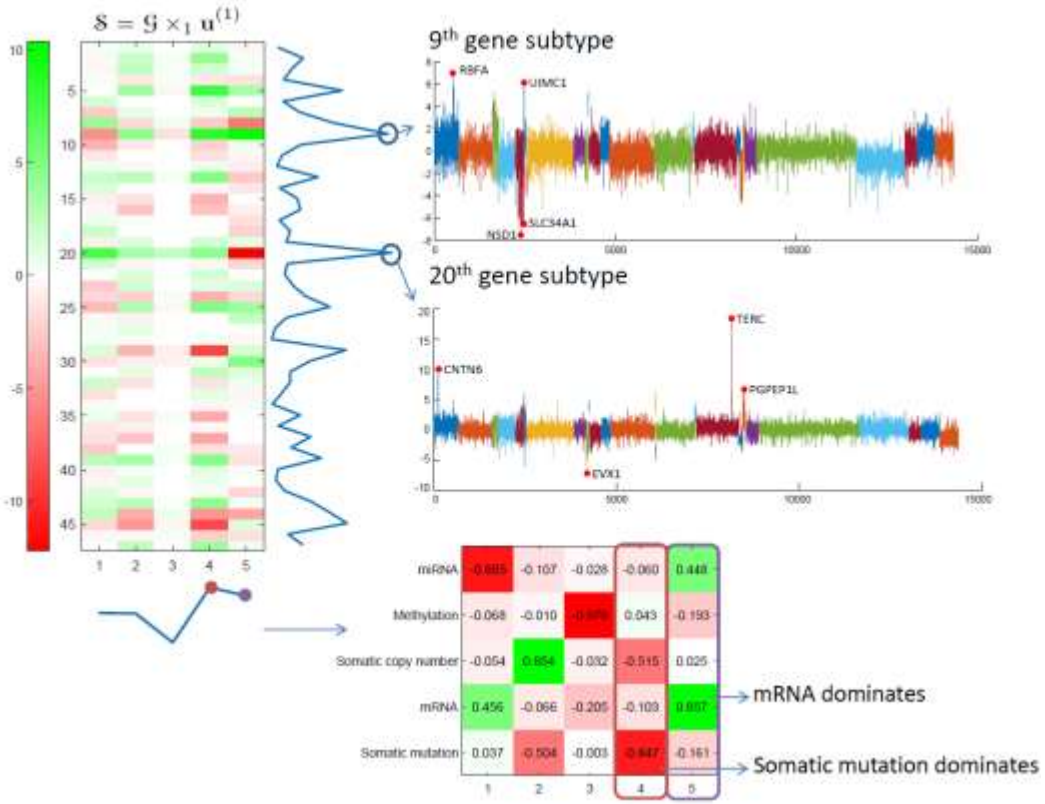


$$\mathcal{X} = \mathcal{G} \times_1 \mathbf{U} \times_2 \mathbf{V} \times_3 \mathbf{W}$$
$$= \sum_r \sum_s \sum_t g_{rst}\, \mathbf{u}_r \circ \mathbf{v}_s \circ \mathbf{w}_t$$
$$\equiv [\![\mathcal{G} ; \mathbf{U}, \mathbf{V}, \mathbf{W}]\!]$$

- Proposed by Tucker (1966)
- **U, V,** and **W** generally **assumed to be orthonormal**
- $\mathcal{G}$ is **not** diagonal
- **Not unique**

Why Interpretability?

- 3. Gain **insights** for advance in science
  - Ex> Detecting causality, detecting significant features

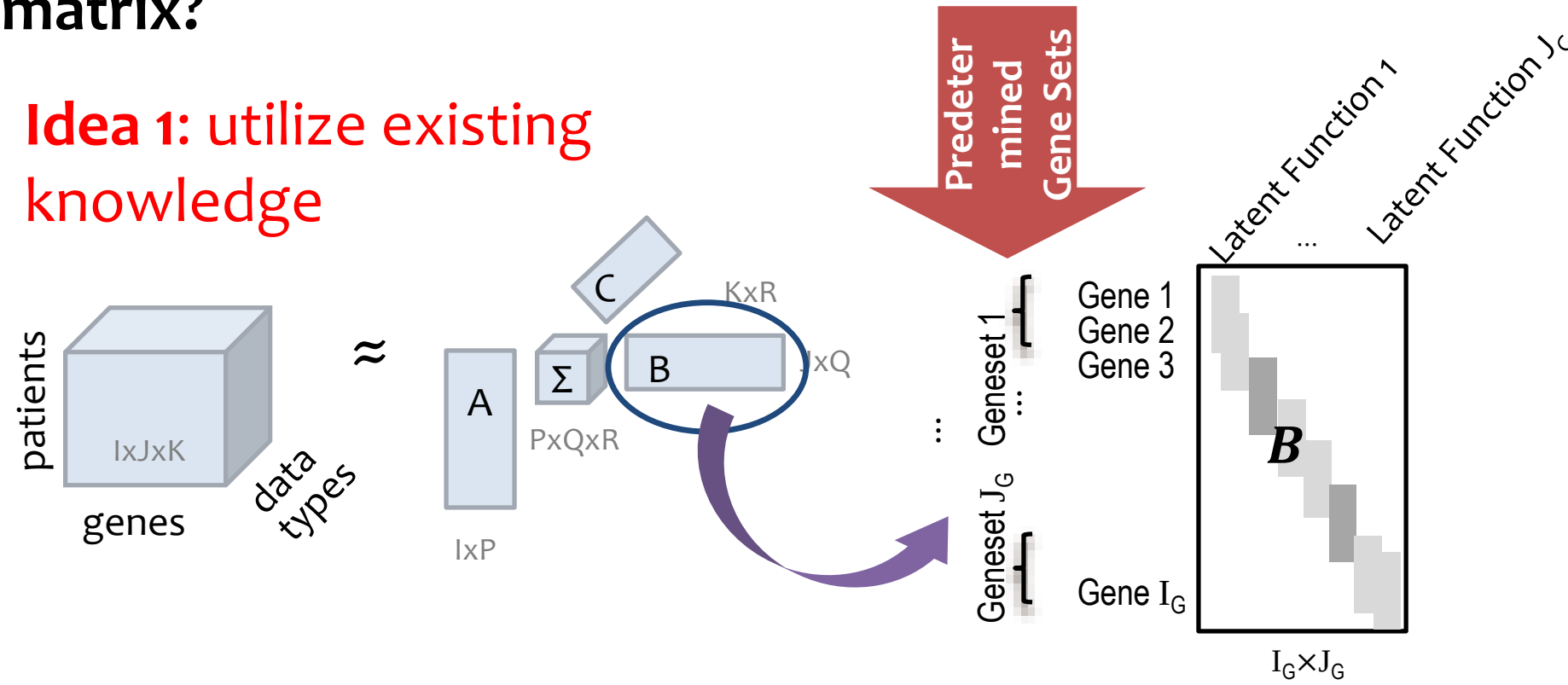


[D. Choi & L. Sael (in submission)]

Interpretable Factor Matrix: Idea 1

How can we enables natural **interpretation** of **factor matrix?**

Idea 1: utilize existing knowledge

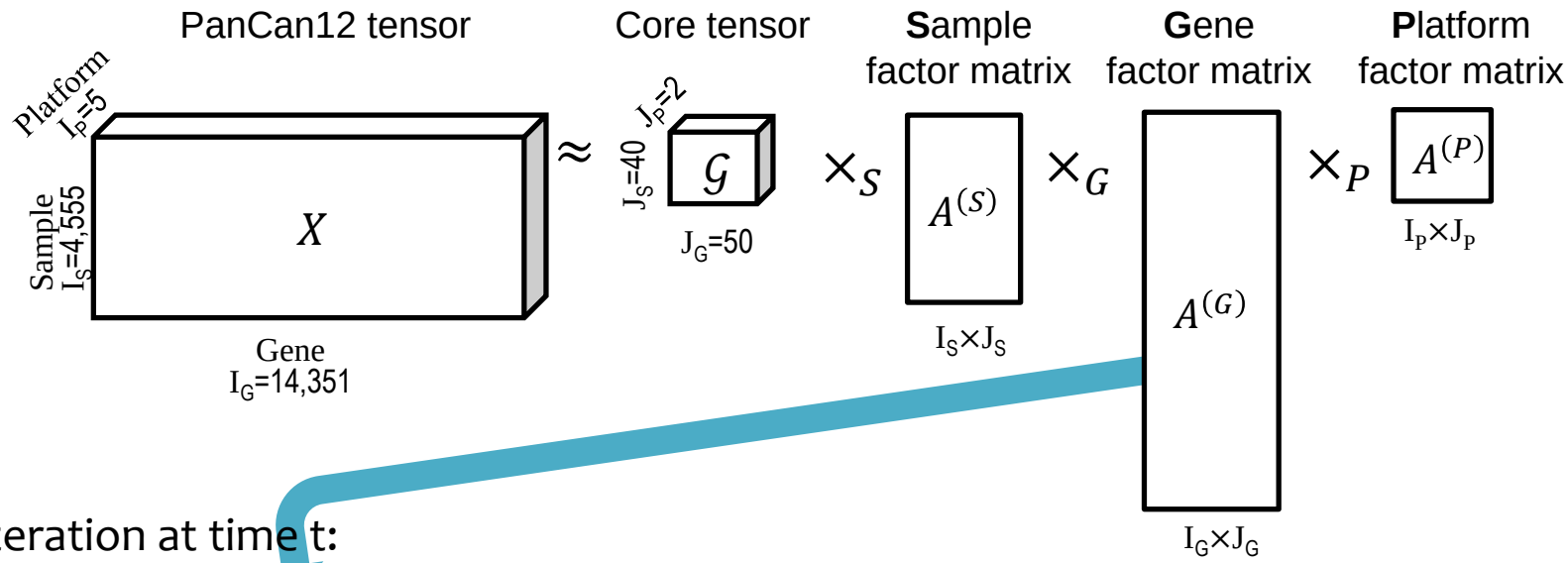


Interpretable Gene Factor Matrix B

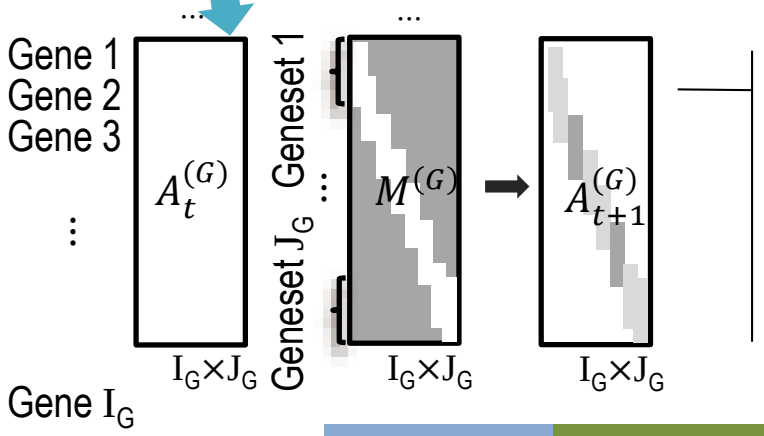
[S. Oh, J. Lee & L. Sael (2018) **Bioinformatics**]

Idea 1: Scheme of GIFT

GIFT: **G**uided and **I**nterpretable **F**actorization for **T**ensors



Iteration at time t:



Row-wise factor matrix **update** rule derived by computing a **gradient of loss function**

- Paralleled using openMP

[S. Oh, J. Lee & L. Sael **Bioinformatics**]

Proposed Method: Objective Function

- Regularize using Mask Matrix $\boldsymbol{M^{(2)}}$ on gene factor matrix

$$L\left(\mathcal{G}, A^{(1)}, A^{(2)}, A^{(3)}, M^{(1)}, M^{(2)}, M^{(3)}\right) =$$

$$\frac{1}{2}\sum_{\forall \alpha \in \Omega}\left(\boldsymbol{x}_{\alpha} - \sum_{\forall \beta \in \mathcal{G}} \mathcal{G}_{\beta} \prod_{n=1}^{N} \boldsymbol{a}_{i_n j_n}^{(n)}\right)^2$$

$$+\frac{\lambda}{2}\sum_{n \in \{1,2,3\}} \left\| M^{(n)} * A^{(n)} \right\|^2$$

Proposed Method: Parallelizable Update Rule

The row-wise update rules are derived by setting the gradient of object function to zero. [S. Oh, J. Lee & L. Sael (2018) **Bioinformatics**]

$$\arg\min_{[a_{i_n:}^{(n)}]} (L(\mathcal{G}, A^{(1)}, \dots, A^{(N)}, M^{(1)}, \dots, M^{(N)})$$

$$= c_{i_n:}^{(n)} \times \left[B_{i_n}^{(n)} + \lambda D_{i_n}^{(n)} \right]^{-1}$$

Intermediate data [S. Oh, L. Sael et al. ICDE 2018]

$$B_{i_n}^{(n)}(j_1, j_2) = \sum_{\forall \alpha \in \Omega} \delta_\alpha^{(n)}(j_1) \delta_\alpha^{(n)}(j_2) \qquad B_{i_n}^{(n)} \in \mathbb{R}^{J_n \times J_n}$$

$$c_{i_n}^{(n)}(j) = \sum_{\forall \alpha \in \Omega} \boldsymbol{x}_\alpha \, \delta_\alpha^{(n)}(j) \qquad c_{i_n}^{(n)} \in \mathbb{R}^{J_n}$$

$$\delta_\alpha^{(n)}(j) = \sum_{\forall (j_1 \dots j_n = j \dots j_N) \in \mathcal{G}} \mathcal{G}_{(j_1 \dots j_n = j \dots j_N)} \prod_{k \neq n} a_{i_k j_k}^{(k)} \qquad \delta_\alpha^{(n)} \in \mathbb{R}^{J_n}$$

[S. Oh, J. Lee & L. Sael **Bioinformatics**]

Proposed Method: GIFT Algorithm

Algorithm 1 3-order GIFT

Input: A tensor $\mathcal{X} \in \mathbb{R}^{I_S \times I_G \times I_P}$ with observable entries Ω, mask matrices $\mathbf{M}^{(S)}$,$\mathbf{M}^{(G)}$, $\mathbf{M}^{(P)}$, rank (J_S, J_G, J_P), and a regularization parameter λ.

Output: A core tensor $\mathcal{G}$ and factor matrices $\mathbf{A}^{(S)}$, $\mathbf{A}^{(G)}$, $\mathbf{A}^{(P)}$.

initialize $\mathcal{G}$ and $\mathbf{A}^{(S)}$, $\mathbf{A}^{(G)}$, $\mathbf{A}^{(P)}$ randomly
repeat
  for $n \in S, G, P$ **do**
    for $i_n = 1, \cdots, I_n$ **do**
      calculate intermediate data δ, $\mathbf{B}_{i_n}^{(n)}$, and $\mathbf{c}_{i_n:}^{(n)}$ by Eq. (2) – (4)
      calculate $\mathbf{D}_{i_n}$, where its (j_n, j_n)th entry is $\mathbf{M}_{i_n j_n}^{(n)}$
      update a row $a_{i_n:}^{(n)}$ by $\mathbf{c}_{i_n:}^{(n)} \times [\mathbf{B}_{i_n}^{(n)} + \lambda \mathbf{D}_{i_n}]^{-1}$
    end for
  end for
  compute reconstruction error by Eq. (5)
until error converges or exceeds maximum iteration

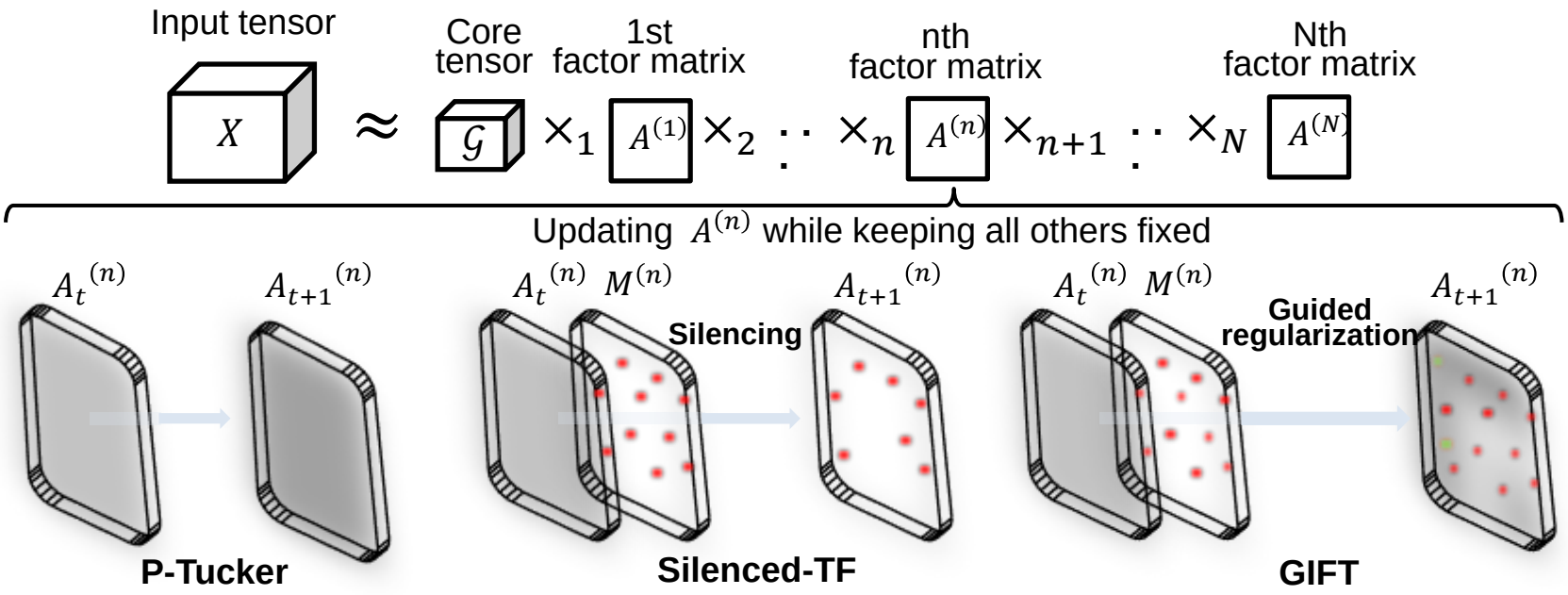
[S. Oh, J. Lee & L. Sael **Bioinformatics**]

GIFT Dataset : PanCan12

Table 2. Summary of dataset. M: million, K: thousand.

Dataset	Order	Size	Observable Entries
PanCan12 tensor	3	$(4,555 \times 14,351 \times 5)$	180M
Sampled-PanCan12	3	$(4,555 \times 14,351 \times 5)$	36−144M
Mask matrix $\mathbf{M}^{(G)}$	2	$(14,351 \times 50)$	7K

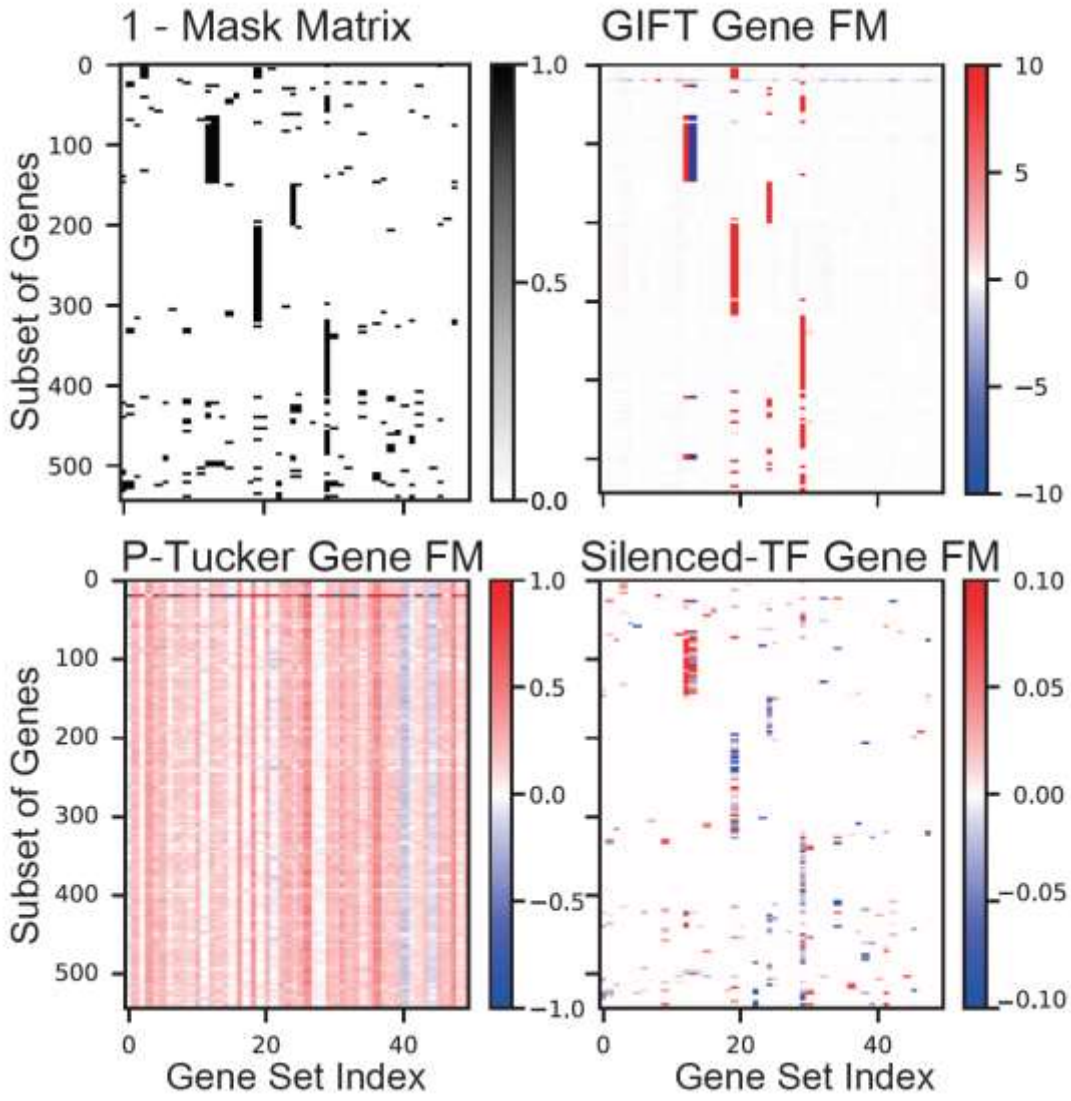
Results: GIFT vs Compared Methods



[S. Oh, L. Sael et al. ICDE 2018]

[S. Oh, J. Lee & L. Sael **Bioinformatics 2018**]

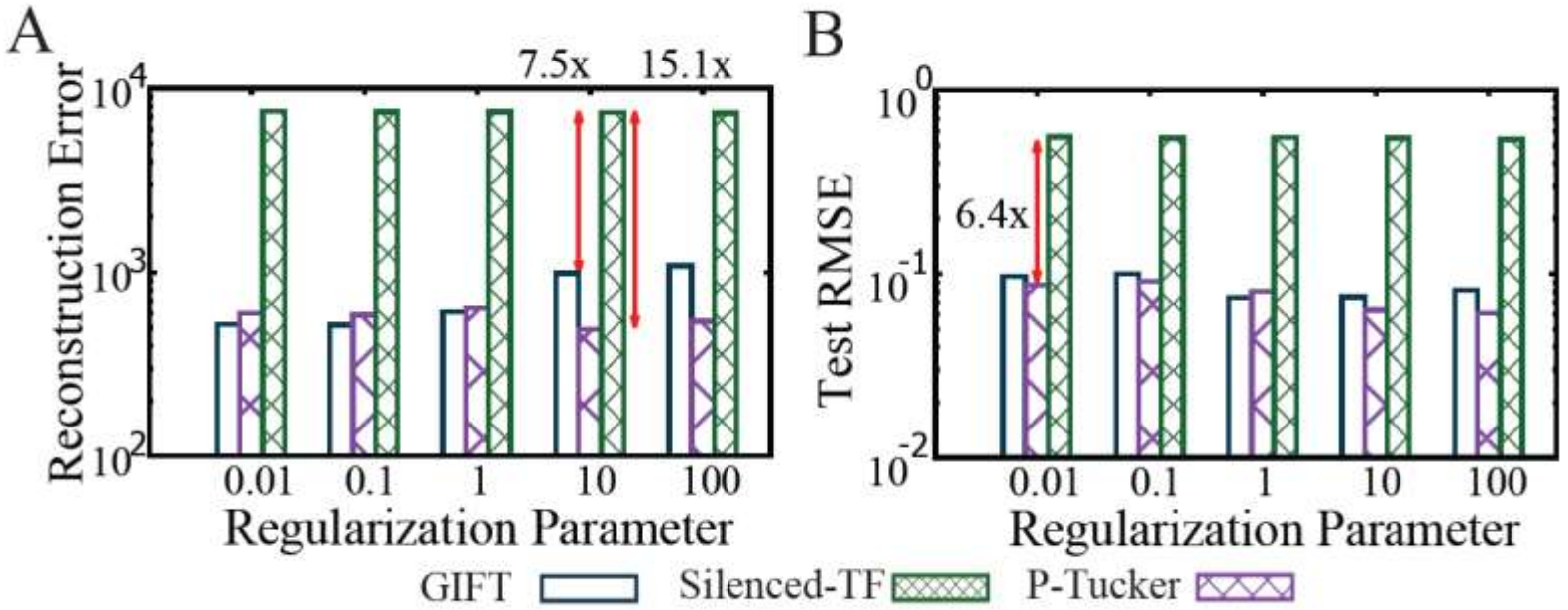
Interpretability



Mask matrix and gene factor matrices (FM) of GIFT, P-Tucker, and Silenced-TF.
Subset of genes are shown for better visualization.

[S. Oh, J. Lee & L. Sael **Bioinformatics**]

Accuracy



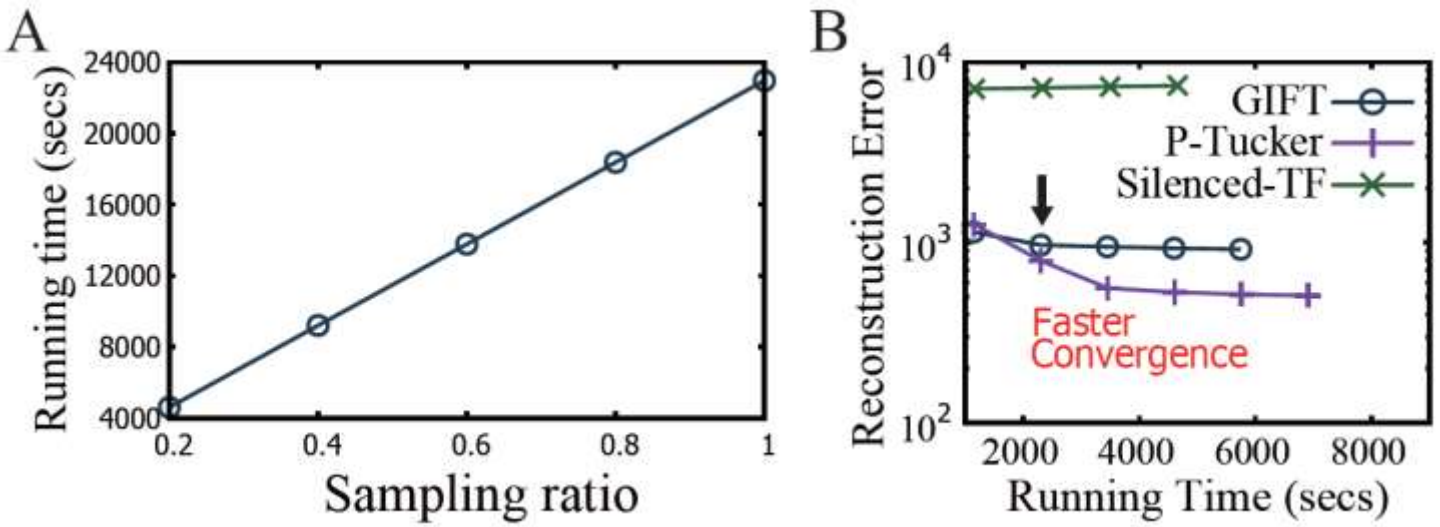
Performance comparisons of GIFT, Silenced-TF, and P-Tucker.

A. is reconstruction error plot.

B. is a test RMSE plot.

[S. Oh, J. Lee & L. Sael **Bioinformatics**]

R4: Scalability



Convergence and scalability of GIFT.

A. GIFT shows faster convergence than Ptucker and has higher accuracy than Silenced-TF.
B. Total running time of GIFT wrt the number of non-zeros.

[S. Oh, J. Lee & L. Sael **Bioinformatics**]

R5: Empirical Validation

Significant relations found on the PanCan12 dataset via GIFT.
(: important gene, +: not included in a gene set, but related).

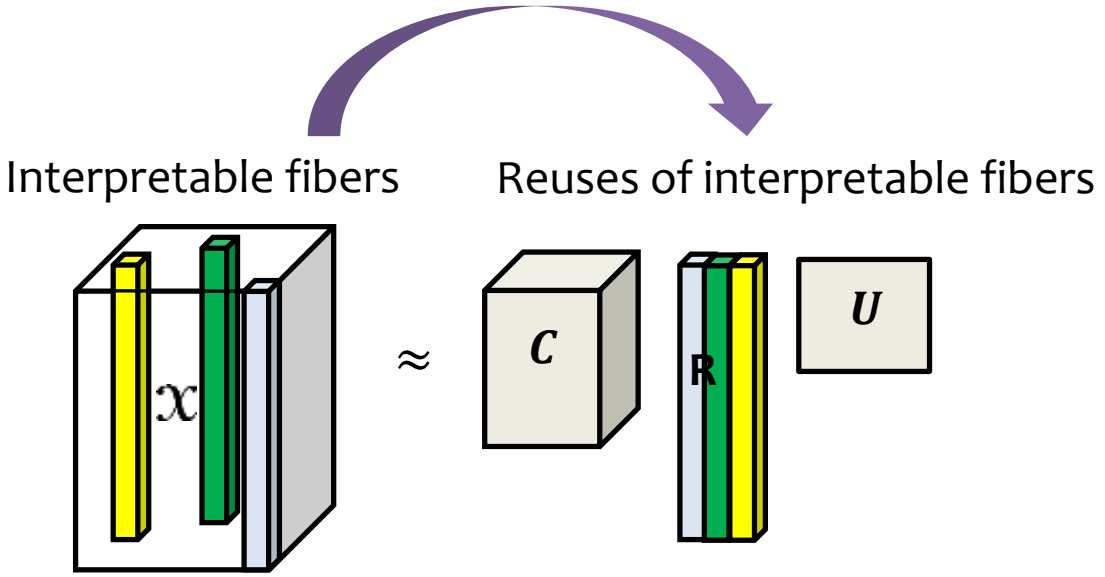
Cancer	Gene set	Genes	Evidence
HNSC, LUAD, LUSC, BLCA	TGF beta signaling	SKIL*	Encodes the SNON, negative regulators of **TGF-beta signaling** (Tecalco-Cruz *et al.*, 2012)
		FKBP1A*	Interacts with a type I **TGF-beta receptor**.
		LEFTY2*	Encodes a secreted ligand of the **TGF-beta family** of proteins.
GBM	Angiogenesis	PF4*	Inhibits cell proliferation and **angiogenesis** in vitro and in vivo (Bikfalvi, 2004).
		VCAN*	Encodes a protein involving in cell adhesion, and **angiogenesis** (Wight, 2002).
BRCA	Estrogen response late	IL17RB*	Involved in development and progression of **breast cancer** (Alinejad *et al.*, 2017).
		TFF3*	Promotes invasion and migration of **breast cancer** (May and Westley, 2015).
	Bile acid metabolism	APOA1*	**Breast cancer** risk factor (Martin *et al.*, 2015).
OV, UCEC	Interferon-gamma response	IRF7*	Encodes **interferon** regulatory factor 7.
		BST2*	High levels of BST2 have been identified in **ovarian cancer** (Shigematsu *et al.*, 2017).
	Apoptosis	CASP8AP2+	Associated with **apoptosis** of leukemic lymphoblasts (Flotho *et al.*, 2006).
			Encoded protein plays a regulatory role in Fas-mediated **apoptosis** (Imai *et al.*, 1999).
READ, COAD	Protein secretion	STX7*	Controls vesicle trafficking events involved in **cytokine secretion** (Achuthan *et al.*, 2008).
KIRC, LAML	Mitotic spindle	LATS1*	Binds phosphorylated zyxin and moves it to the **mitotic spindle**

[J. Lee, D. Choi, and **L. Sael** (2018) PLOS One]

Interpretable Factor Matrix: Idea 2

How can we enables natural **interpretation** of **factor matrix?**

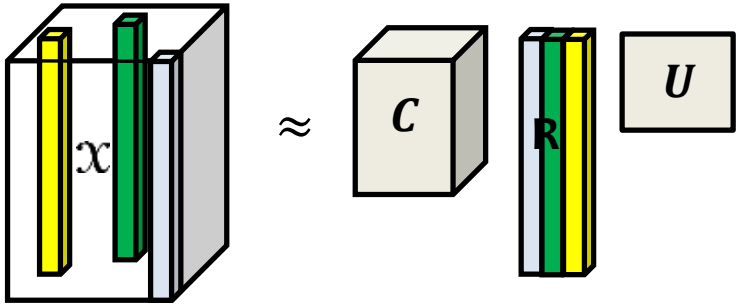
Idea 2: use sparse and interpretable input fibers as columns of a factor matrix



Interpretable Factor Matrix $\boldsymbol{R}$

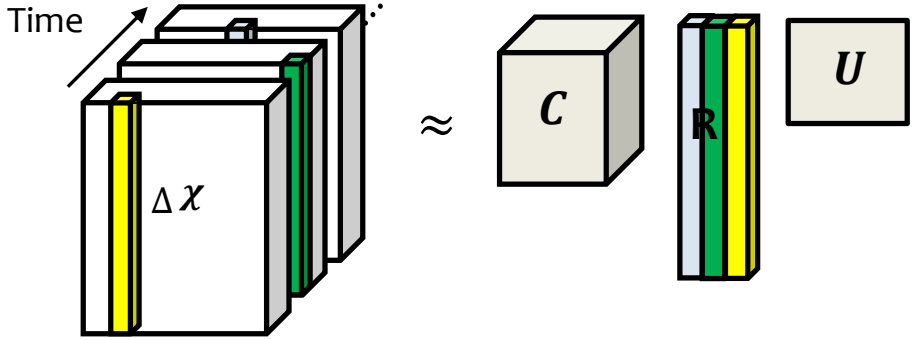
Motivation - Problem

- **Q1.** How can we design an efficient sampling-based tensor decomposition in a **static** environment?
  - Efficient = accurate, quick, and memory-efficient
- **Q2.** How can we do this in a **dynamic** environment?



Static environment

→ Offline, full data is given

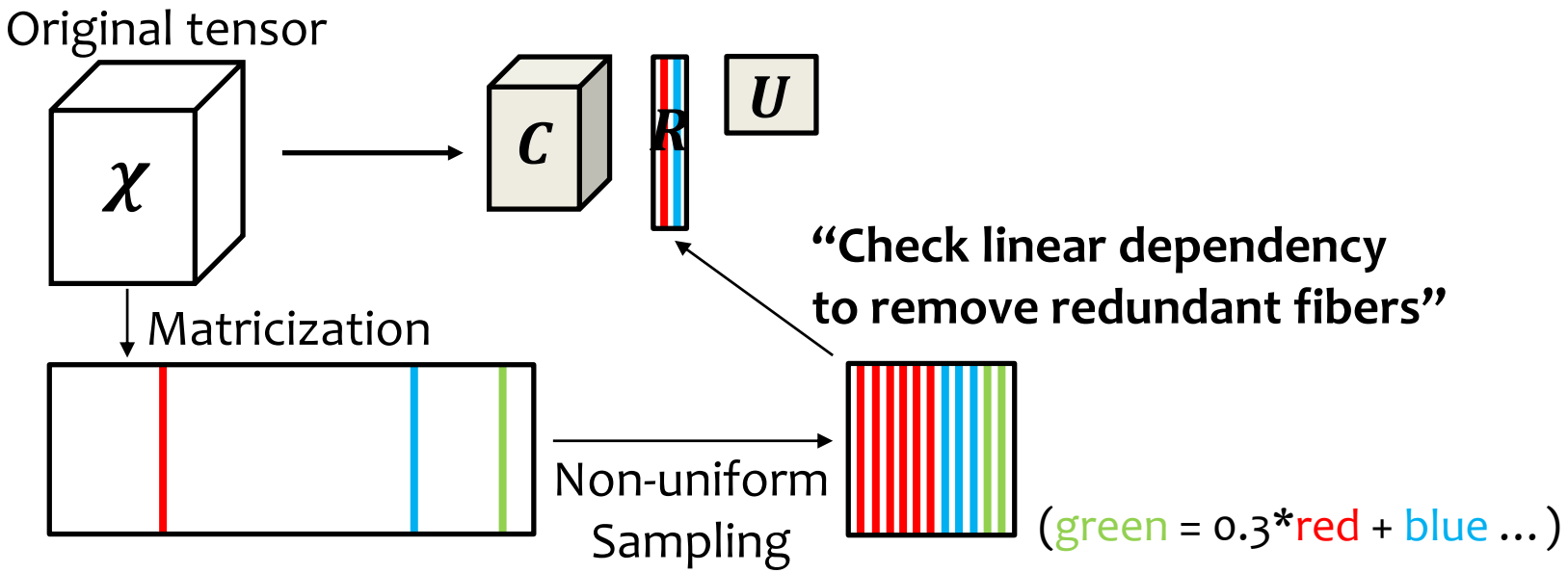


Dynamic environment

→ Online, data arrives at every time step

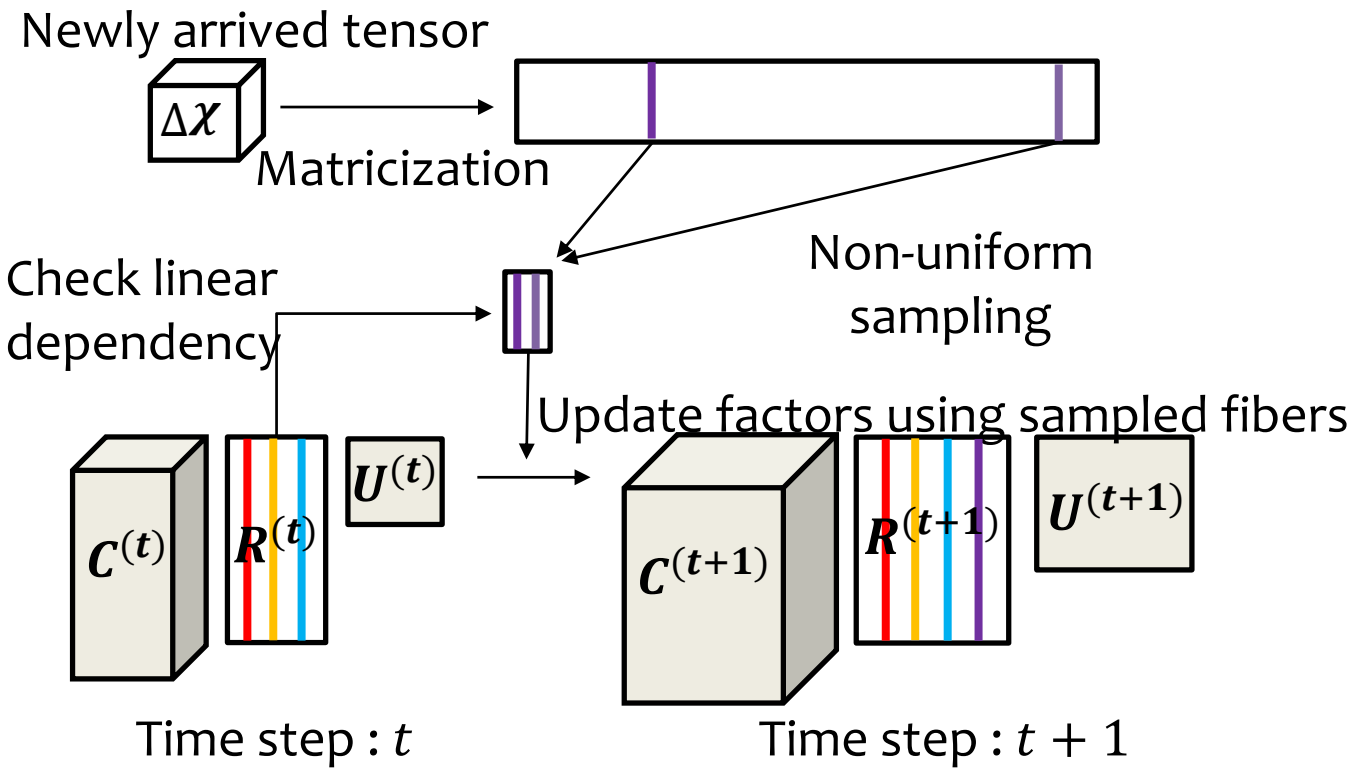
Algorithm – CTD-S

- Static version of CTD
- Only maintain linearly independent fibers to keep result compact.
- Input : tensor $\mathcal{X}$, sample size s
- Output : tensor $\mathcal{C}$, matrices U and R (consisting of fibers of $\mathcal{X}$)



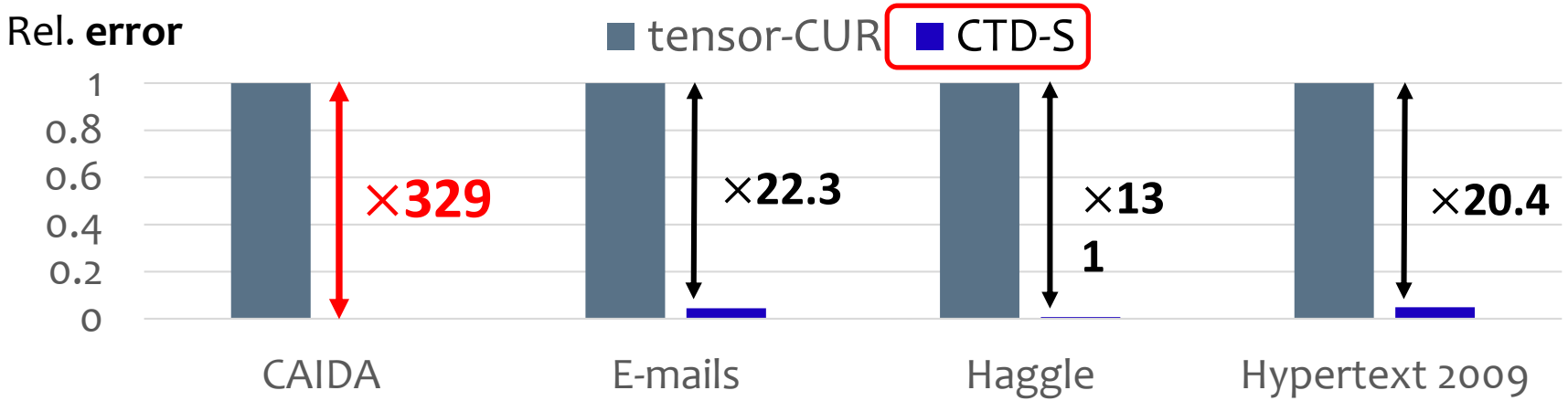
Algorithm – CTD-D

- Dynamic version of CTD
- Exploits existing factors at previous time step to update its factors quickly.
- Input : new tensor $\Delta\boldsymbol{\mathcal{X}}$, factors $\boldsymbol{C}, \boldsymbol{U}, \boldsymbol{R}$ at previous time step t
- Output : factors $\boldsymbol{C}, \boldsymbol{U}, \boldsymbol{R}$ at previous time step $t+1$

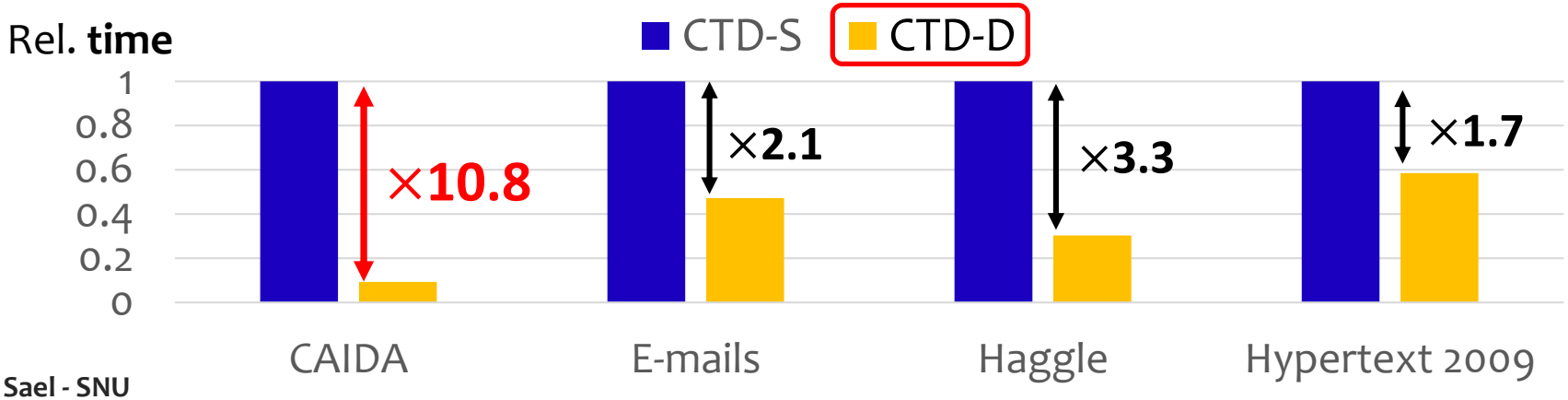


Performance

- CTD-S (static) : **2 ~ 300× more accurate** than tensor-CUR
  - Theoretically, it is proven that **CTD-S has the optimal accuracy.**



- CTD-D (dynamic) : **2 ~ 11× faster** than CTD-S



Online DDoS attack detection Example

- Directly determine destination host and occurrence time of a major activity represented in a fiber in **R** by simply tracking the indices of fibers.

n	Recall	Precision	F1 score
1	1.000	1.000	1.000
3	1.000	1.000	1.000
5	0.880	1.000	0.931
7	0.857	1.000	0.921

https://doi.org/10.1371/journal.pone.0200579.t005

Table 5. The result of online DDoS attack detection method based on CTD-D. *n* denotes the number of injected DDoS attacks

Reference

- K. L. Wagstaff, N. L. Lanza, D. R. Thompson, T. G. Dietterich, and M. S. Gilmore, “Guiding Scientific Discovery with Explanations using DEMUD,” in *AAAI 2013*, 2013, p. 7
- Lee, J., Oh S., & **Sael, L.** (2018). GIFT: Guided and Interpretable Factorization for Tensors with an Application to Large-Scale Multi-platform Cancer Analysis. Bioinformatics, bty490.
- Lee, J., Choi, D., & **Sael, L.** (2018). CTD: Fast, Accurate, and Interpretable Method for Static and Dynamic Tensor Decompositions. PLOS One 13(7):e0200579.