GB-CENT

Gradient Boosted Categorical Embedding and Numerical Trees

June 23, 2017
Chinese Academy of Sciences

Liangjie Hong
Head of Data Science, Etsy Inc.

Liangjie Hong

- **Head of Data Science**
  - **Etsy Inc.** in NYC, NY (2016 – Present)
  - Search & Discovery; Personalization and Recommendation; Computational Advertising

- **Research Scientist**
  Senior Research Scientist
  Senior Manager of Research
  - **Yahoo Research** in Sunnyvale, CA (2013 – 2016)
  Leading science efforts for personalization and search sciences

- Published papers in **SIGIR**, **WWW**, **KDD**, **CIKM**, **AAAI**, **WSDM**, **RecSys** and **ICML**

- 3 Best Paper Awards, 2000+ Citations with H-Index 18

- Program committee members in **KDD**, **WWW**, **SIGIR**, **WSDM**, **AAAI**, **EMNLP**, **ICWSM**, **ACL**, **CIKM**, **IJCAI** and various journal reviewers

- PhD in Computer Science from Lehigh University (2013)

About This Paper

- Authors
  Qian Zhao, PhD Student from **University of Minnesota**
  Yue Shi, Research Scientist at **Facebook**
  Liangjie Hong, Head of Data Science at **Etsy Inc.**

- Paper Venue
  Full Research Paper in The 26th International World Wide Web Conference, 2017 (**WWW 2017**)

High-Level Takeaways

- **A new family of models to handle categorical features and numerical features well by combining embedding models and tree-based models**
- **A simple learning algorithm that can be easily extended from existing data mining and machine learning toolkits**
- **State-of-the-art performance on major datasets**

Why we need GB-CENT

Why we need GB-CENT

Motivations

- **Real-World Data**
  Categorical features: user ids, item ids, words, document ids, ...
  Numerical features: dwell time, average purchase prices, click-through-rate,...

Why we need GB-CENT

Motivations

- **Real-World Data**
  Categorical features: user ids, item ids, words, document ids, ...
  Numerical features: dwell time, average purchase prices, click-through-rate,...

- **Ideas**
  Converting categorical features into numerical ones (e.g., statistics, embedding methods, topic models...)
  Converting numerical features into categorical ones (e.g., bucketizing, binary codes, sigmoid transformation...)

Why we need GB-CENT

Motivations

Two Families of Powerful Practical Data Mining and Machine Learning Tools

- **Tree-based Models**
  Decision Trees, Random Forest, Gradient Boosted Decision Trees…
- **Matrix-based Embedding Models**
  Matrix Factorization, Factorization Machines…

Why we need GB-CENT: Tree-based Models

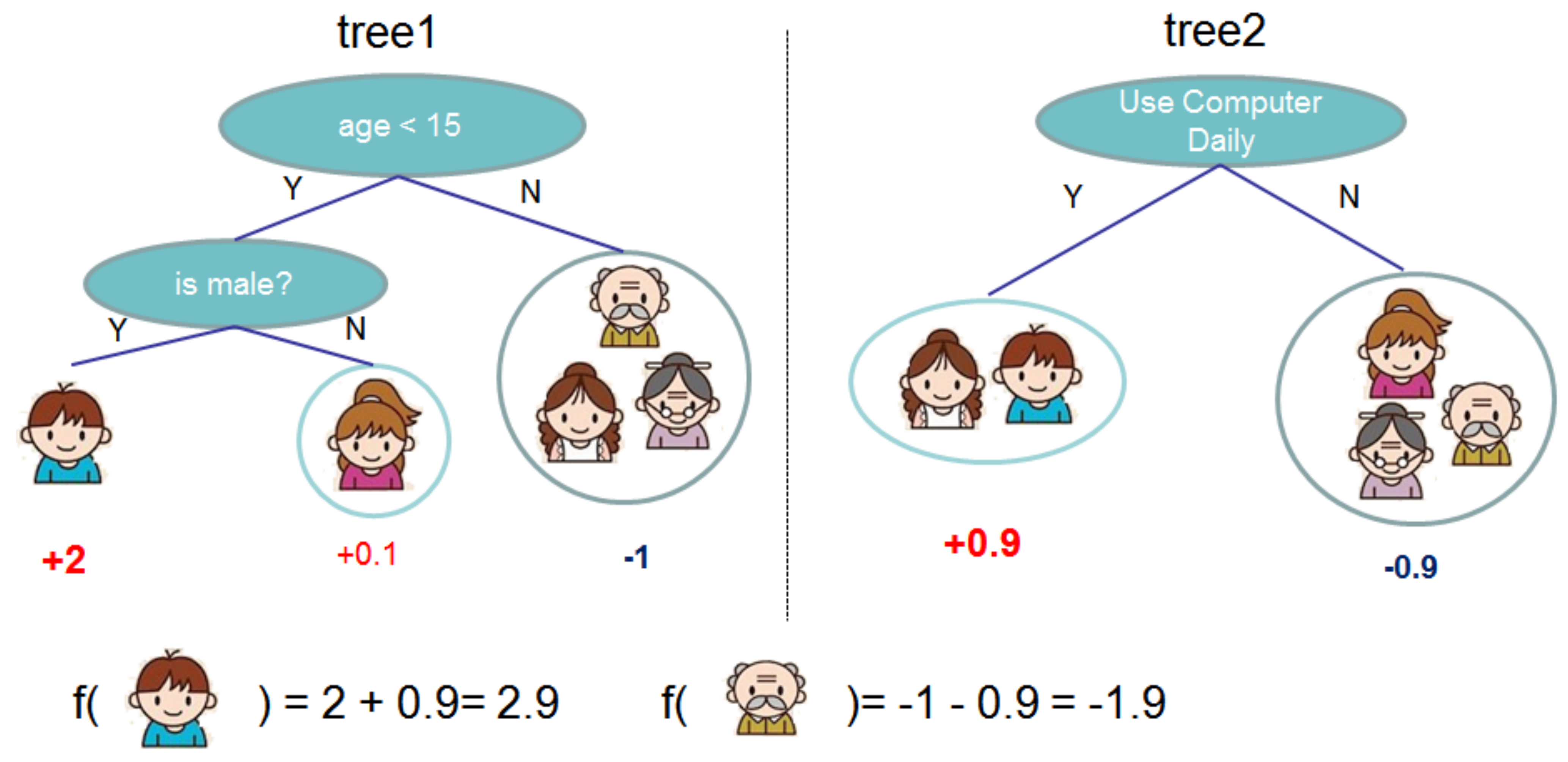
Gradient Boosted Decision Trees (GBDT)

- **Tree Ensembles**
- **Learning Trees in Additive Fashion**
- **Utilize Gradient Boosting**

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t)$$



Why we need GB-CENT: Tree-based Models

- **Pros:**

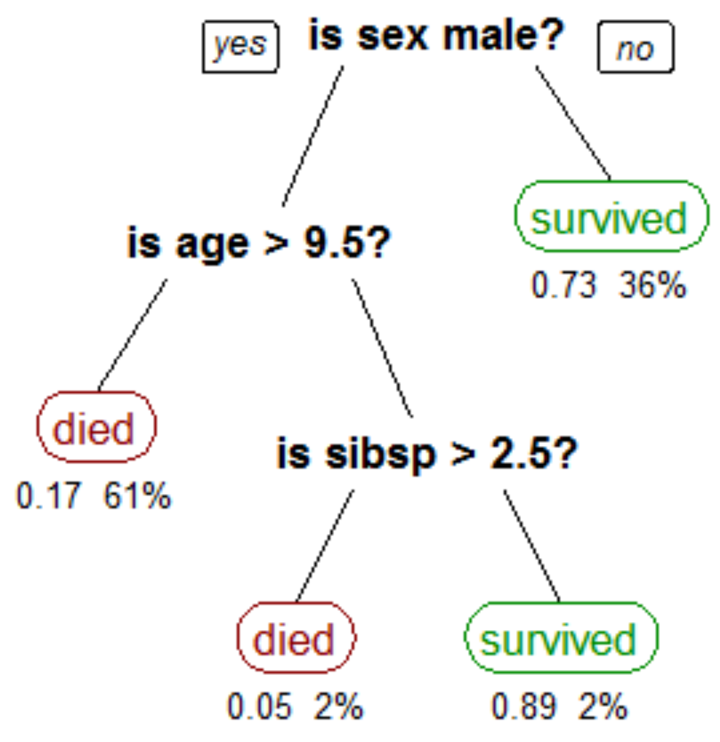
  Interpretability for simple trees

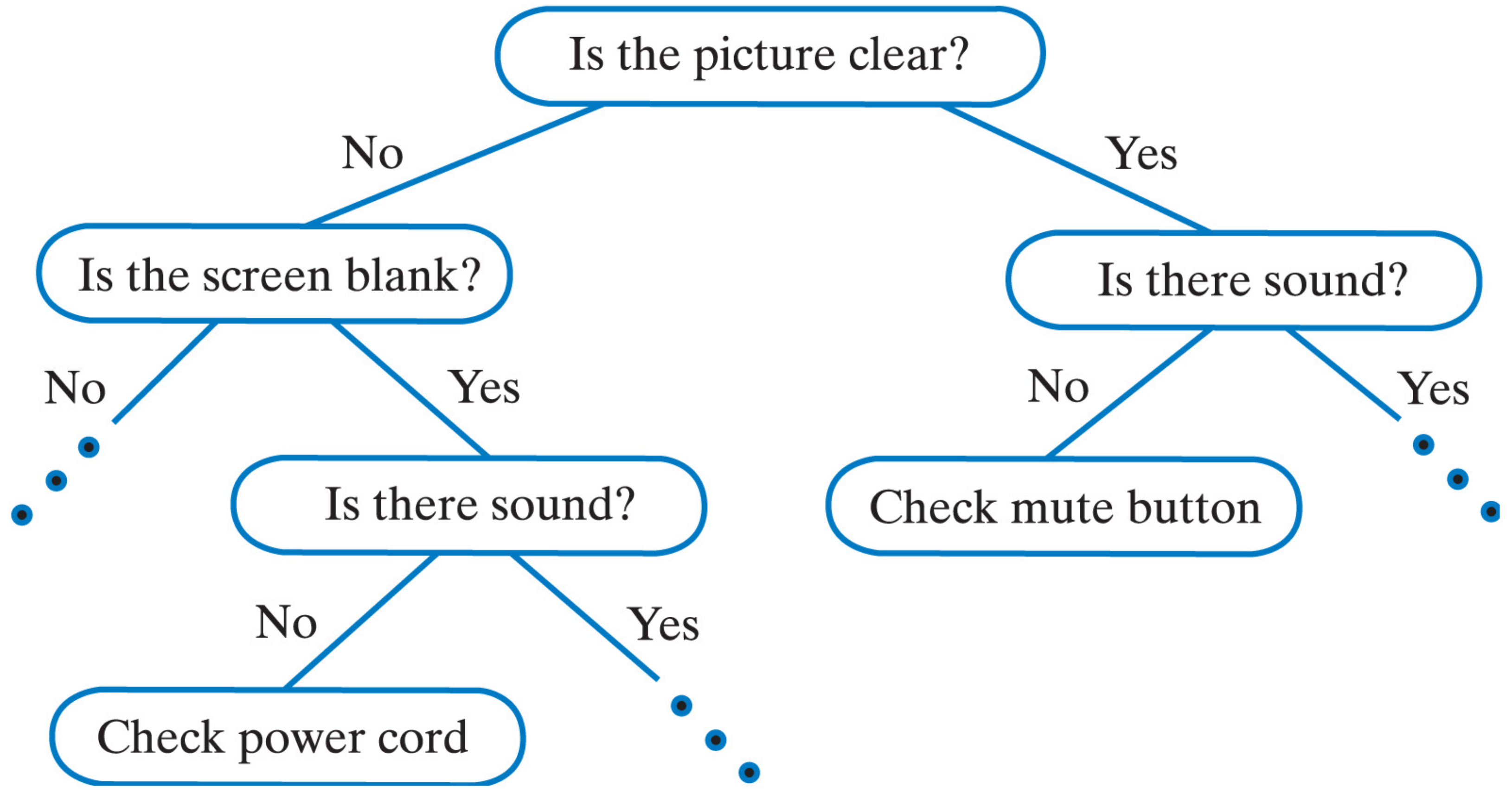
  Effectiveness in certain tasks: IR ranking models

  Simple and easy to train

  Handle numerical features well

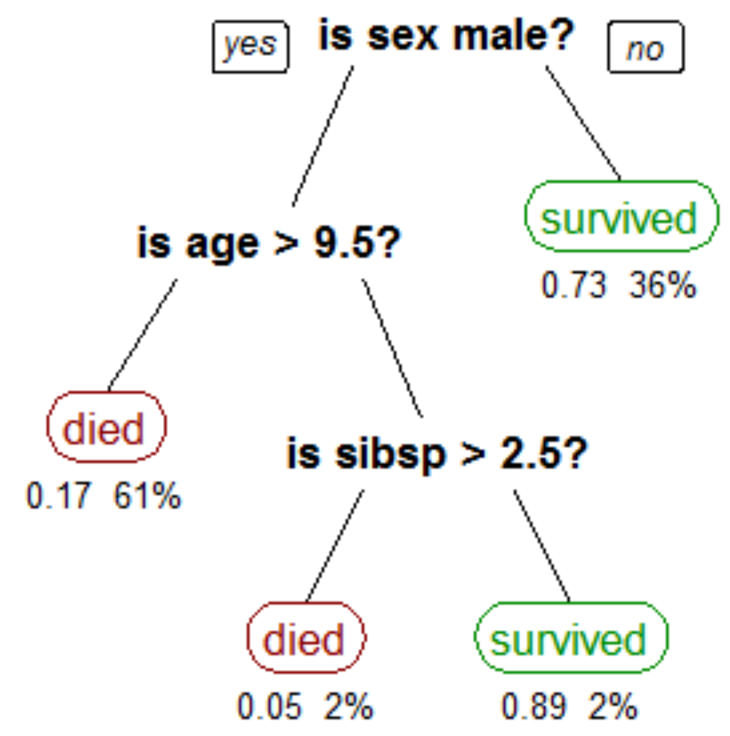
  …



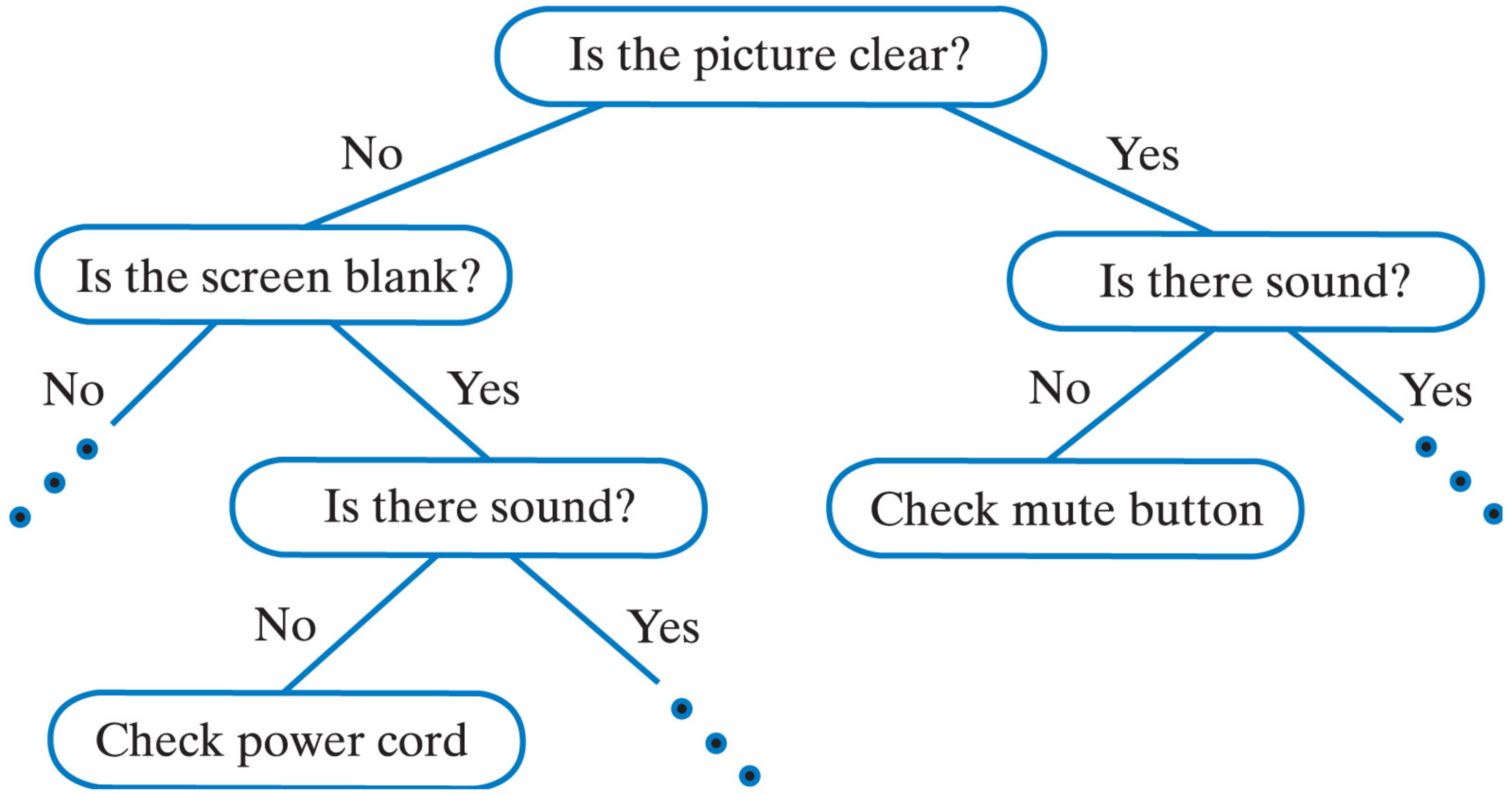


Why we need GB-CENT: Tree-based Models

- **Pros:**
  Interpretability for simple trees
  Effectiveness in certain tasks: IR ranking models
  Simple and easy to train
  Handle numerical features well
  …



- **Cons:**
  Need one-hot-encoding to handle categorical features and therefore cannot easily handle features with large cardinality*
  For complex trees, features might appear multiple times in a tree – hard to explain
  …



*TIANQI CHEN AND CARLOS GUESTRIN. **XGBOOST: A SCALABLE TREE BOOSTING SYSTEM**. KDD '16.

Why we need GB-CENT: Embedding-based Models

Factorization Machines

Feature vector x

	User				Movie					Other Movies rated					Time	Last Movie rated					Target y	
	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...	Time	TI	NH	SW	ST	...		
$x^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	13	0	0	0	0	...	5	$y^{(1)}$
$x^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...	3	$y^{(2)}$
$x^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...	1	$y^{(2)}$
$x^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...	4	$y^{(3)}$
$x^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...	5	$y^{(4)}$
$x^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...	1	$y^{(5)}$
$x^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...	5	$y^{(6)}$

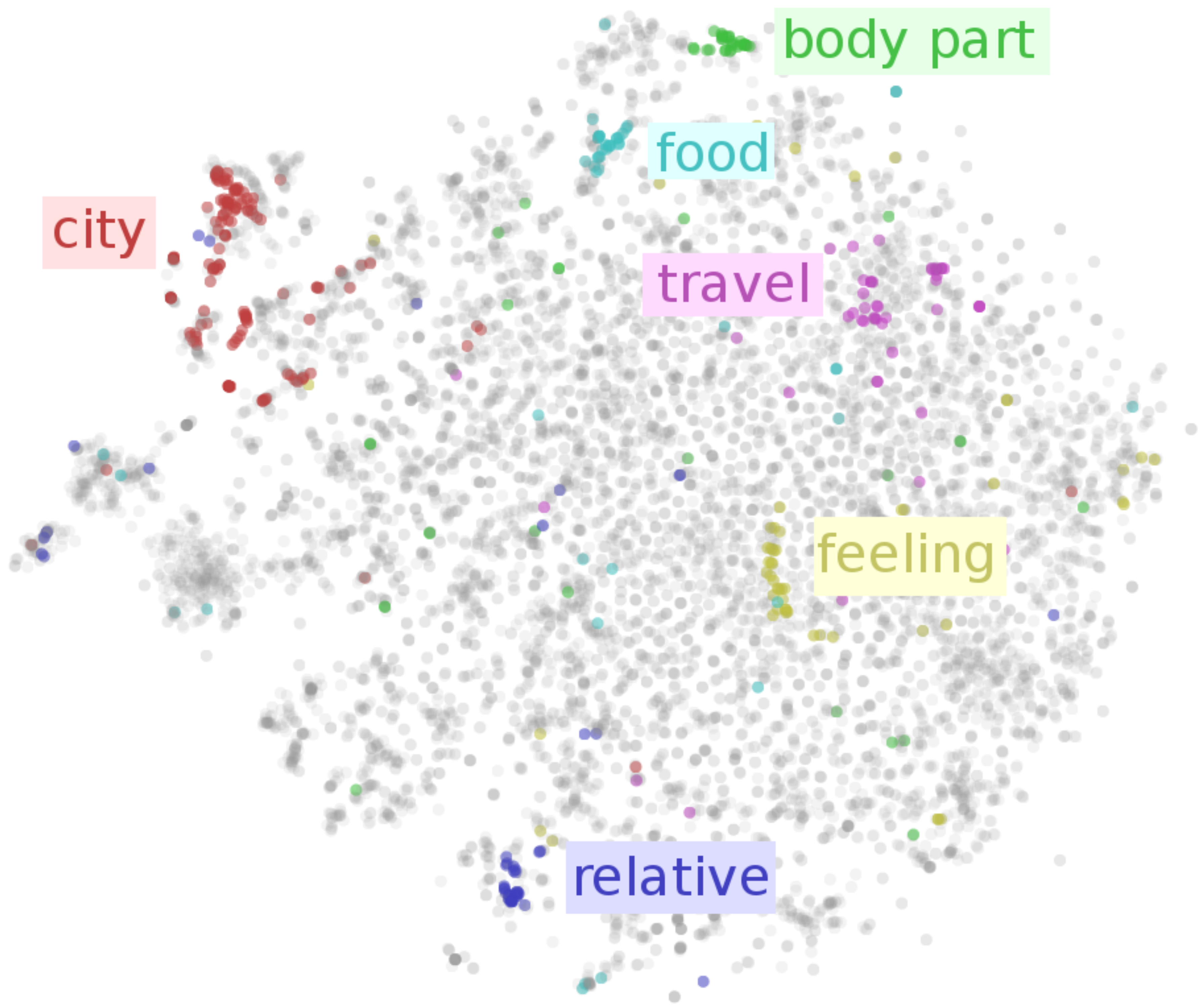
$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i \, x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j \qquad (1)$$

where the model parameters that have to be estimated are:

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^n, \quad \mathbf{V} \in \mathbb{R}^{n \times k} \qquad (2)$$

And $\langle \cdot, \cdot \rangle$ is the dot product of two vectors of size k:

$$\langle \mathbf{v}_i, \mathbf{v}_j \rangle := \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f} \qquad (3)$$



Why we need GB-CENT: Embedding-based Models

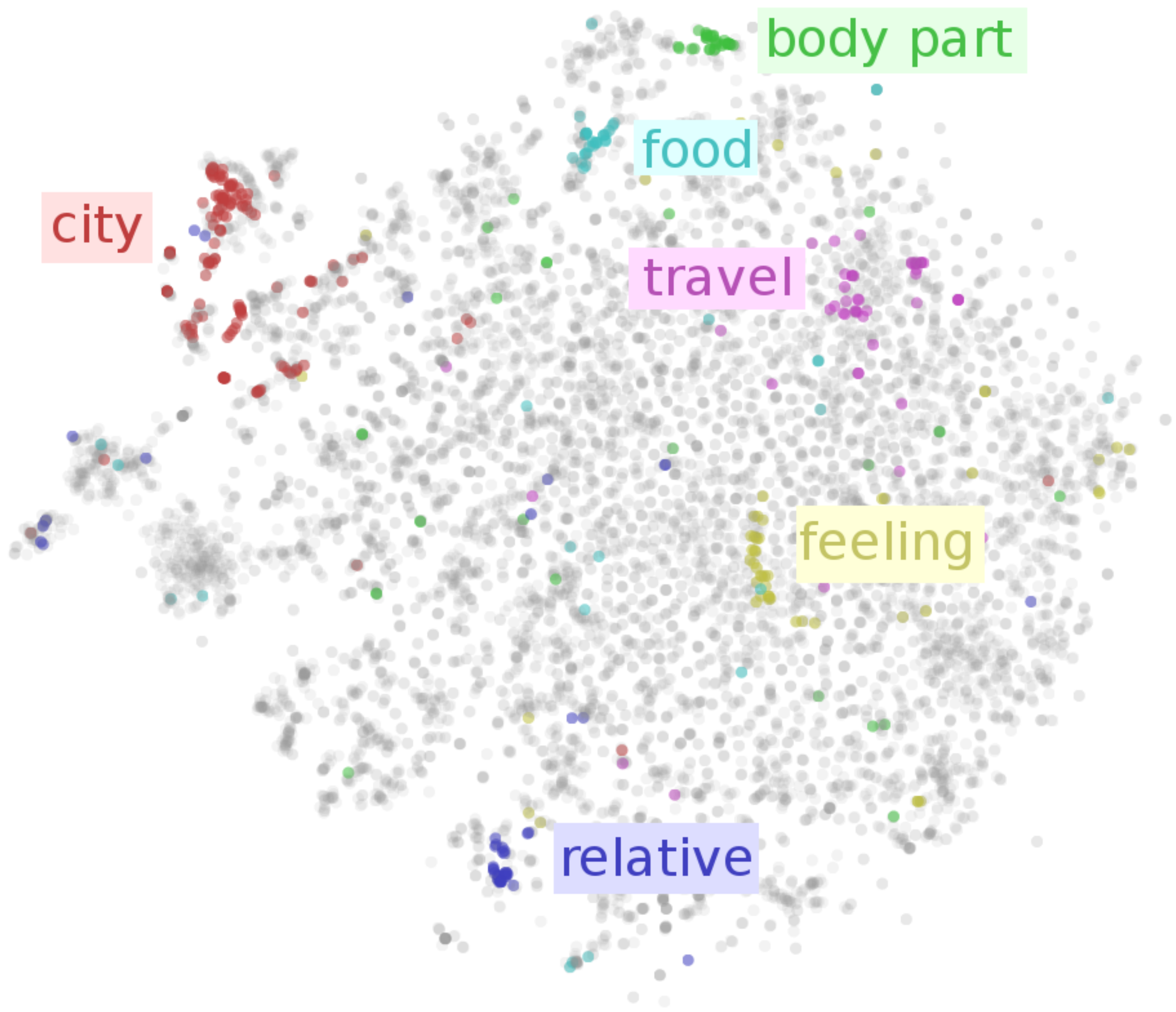
- **Pros:**

  Predictive power

  Effectiveness in certain tasks: recommender systems

  Handle categorical features well through one-hot-encoding

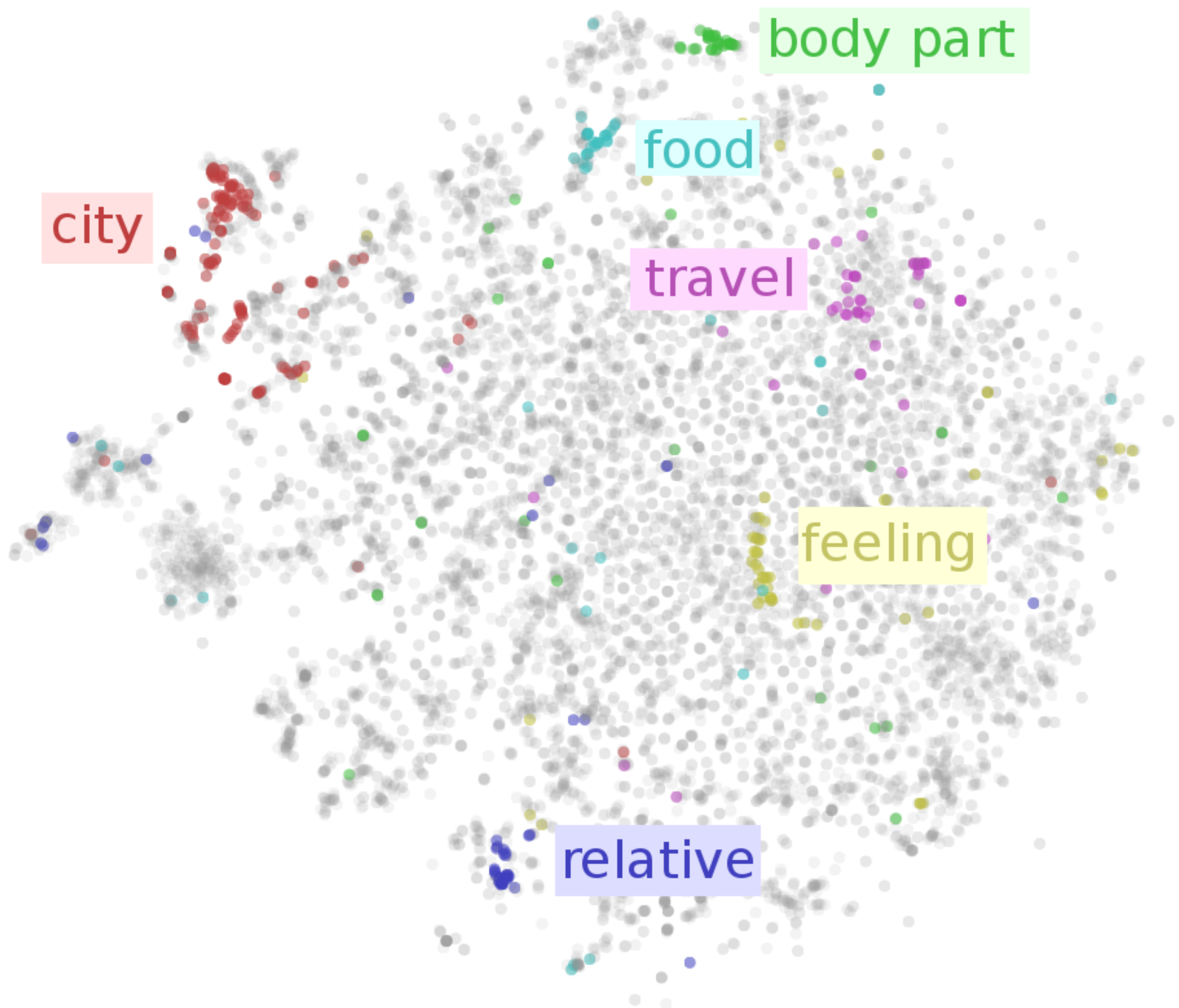
  …



Why we need GB-CENT: Embedding-based Models

- **Pros:**
  Predictive power
  Effectiveness in certain tasks: recommender systems
  Handle categorical features well through one-hot-encoding
  …

- **Cons:**
  Numerical features usually need preprocessing and hard to handle.
  Hard to interpret in general
  …



Why we need GB-CENT

Tree-based models are good at numerical features.

Embedding models are good at categorical features.

Why not combine them two?

What is GB-CENT

What is GB-CENT

In a nutshell, GB-CENT is Gradient **B**oosted **C**ategorical **E**mbedding and **N**umerical **T**rees**, which combines**

- **Matrix-based Embedding Models**
  Handle large-cardinality categorical features…

- **Tree-based Models**
  Handle numerical features…

What is GB-CENT

In a nutshell, GB-CENT is Gradient **B**oosted **C**ategorical **E**mbedding and **N**umerical **T**rees**, which combines**

- **Factorization Machines**
  Handle large-cardinality categorical features…

- **Gradient Boosted Decision Trees**
  Handle numerical features…

What is GB-CENT

$$\hat{y(x)} = \boxed{\underbrace{\underbrace{\sum_{i=0}^{k} w_{a_i}}_{bias} + \underbrace{(\sum_{a_i \in U(a)} Q_{a_i})^{\intercal}(\sum_{a_i \in I(a)} Q_{a_i})}_{factor}}_{CAT\text{-}E}} + \underbrace{\sum_{i=0}^{k} T_{a_i}(b)}_{CAT\text{-}NT}$$

CAT-E (Factorization Machines)

- Bias term for each categorical feature
- Embedding for each categorical feature
- Interactions between meaningful categorical groups
  e.g., users, items, age groups, gender...

No numerical features

What is GB-CENT

$$\hat{y(x)} = \underbrace{\underbrace{\sum_{i=0}^{k} w_{a_i}}_{bias} + \underbrace{(\sum_{a_i \in U(a)} Q_{a_i})^{\intercal}(\sum_{a_i \in I(a)} Q_{a_i})}_{factor}}_{CAT\text{-}E} + \boxed{\underbrace{\sum_{i=0}^{k} T_{a_i}(b)}_{CAT\text{-}NT}}$$

CAT-NT (Gradient Boosted Decision Trees)

- One tree per categorical feature (potentially)
- For each tree, the training data is **all data instances with numerical features containing this particular categorical feature**.

No categorical features

What is GB-CENT

$$\hat{y(x)} = \underbrace{\underbrace{\sum_{i=0}^{k} w_{a_i}}_{bias} + \underbrace{\Big(\sum_{a_i \in U(a)} Q_{a_i}\Big)^{\intercal}\Big(\sum_{a_i \in I(a)} Q_{a_i}\Big)}_{factor}}_{CAT\text{-}E} + \underbrace{\sum_{i=0}^{k} T_{a_i}(b)}_{CAT\text{-}NT}$$

CAT-E (Factorization Machines)

- *generalizes* categorical features by embedding them into low-dimensional space.

CAT-NT (Gradient Boosted Decision Trees)

- *memorizes* each categorical feature's peculiarities.

HENG-TZE CHENG, LEVENT KOC, JEREMIAH HARMSEN, TAL SHAKED, TUSHAR CHANDRA, HRISHI ARADHYE, GLEN ANDERSON, GREG CORRADO, WEI CHAI, MUSTAFA ISPIR, ROHAN ANIL, ZAKARIA HAQUE, LICHAN HONG, VIHAN JAIN, XIAOBING LIU, AND HEMAL SHAH. **WIDE & DEEP LEARNING FOR RECOMMENDER SYSTEMS**. IN *PROCEEDINGS OF THE 1ST WORKSHOP ON DEEP LEARNING FOR RECOMMENDER SYSTEMS* (DLRS 2016). ACM, NEW YORK, NY, USA, 7-10.

What is GB-CENT

$$\hat{y(x)} = \underbrace{\underbrace{\sum_{i=0}^{k} w_{a_i}}_{bias} + \underbrace{(\sum_{a_i \in U(a)} Q_{a_i})^{\intercal}(\sum_{a_i \in I(a)} Q_{a_i})}_{factor}}_{CAT\text{-}E} + \underbrace{\sum_{i=0}^{k} T_{a_i}(b)}_{CAT\text{-}NT}$$

Different from GBDT:

- The number of trees in GB-CENT depends on the cardinality of categorical features in the data set, while GBDT has a pre-specified number of trees M.
- Each tree in GB-CENT only takes numerical features as input while GBDT takes in both categorical and numerical features.
- Learning a tree for GBDT uses all N instances in the data set while the tree for a categorical feature in GB-CENT only involves its supporting instances.

What is GB-CENT

$$\hat{y(x)} = \underbrace{\underbrace{\sum_{i=0}^{k} w_{a_i}}_{bias} + \underbrace{(\sum_{a_i \in U(a)} Q_{a_i})^{\intercal}(\sum_{a_i \in I(a)} Q_{a_i})}_{factor}}_{CAT\text{-}E} + \underbrace{\sum_{i=0}^{k} T_{a_i}(b)}_{CAT\text{-}NT}$$

Training GB-CENT:

- Train CAT-E part firstly using Stochastic Gradient Descent (SGD)
- Train CAT-NT part secondly

What is GB-CENT

$$\hat{y(x)} = \underbrace{\underbrace{\sum_{i=0}^{k} w_{a_i}}_{bias} + \underbrace{(\sum_{a_i \in U(a)} Q_{a_i})^{\mathsf{T}} (\sum_{a_i \in I(a)} Q_{a_i})}_{factor}}_{CAT\text{-}E} + \underbrace{\sum_{i=0}^{k} T_{a_i}(b)}_{CAT\text{-}NT}$$

Training GB-CENT:

- Train CAT-E part firstly using Stochastic Gradient Descent (SGD)

- Train CAT-NT part secondly
  -- 1) Sort categorical features by their support (how many data instances)
  -- 2) Check whether we meet *minTreeSupport*
  -- 3) Use *maxTreeDepth* and *minNodeSplit* to fit a tree
  -- 4) Use *minTreeGain* to decide whether keeping a tree

How does GB-CENT perform

How does GB-CENT perform

- **Datasets**
  MovieLens
  Statistics: 240K users, 33K movies, 22M instances, 5 ratings
  Categorical features: user_id, item_id, genre, language, country, grade
  Numerical features: year, runTime, imdbVotes, imdbRating, metaScore

  RedHat
  Statistics: 151K customers, 7 categories, 2M instances, binary response
  Categorical features: people_id, activity_category
  Numerical features: activity characteristics

How does GB-CENT perform

- **Datasets**

  MovieLens

  Evaluation Metric: Root Mean Squared Error (RMSE)

  RedHat

  Evaluation Metric: Area Under the Curve (AUC)

  80% of train, 10% of validation and 10% of testing

  We also compare empirical training time.

How does GB-CENT perform

- **Baselines**

  GB-CENT variants:

  1) CAT-E

  2) CAT-NT

  3) GB-CENT

  GBDT variants:

  1) GBDT-OH: GBDT + One-hot-encoding for categorical features

  2) GBDT-CE: Fit CAT-E firstly and then feed into GBDT

  FM variants:

  1) FM-S: Transform numerical features by sigmoid and feed into FM

  2) FM-D: Transform numerical features by discretizing them and feed into FM

  SVDFeature variants:

  1) SVDFeature-S: Transform numerical features by sigmoid and feed into SVDFeature

  2) SVDFeature-D: Transform numerical features by discretizing them and feed into SVDFeature

  All latent dimensionality is 20. For GB-CENT, *minTreeSupport* = 50, *minTreeGain* = 0.0, *minNodeSplit* = 50 and *maxTreeDepth* = 3.

How does GB-CENT perform

Data Set	Metric	GBDT-OH	GBDT-CE	SVDFeature-S	SVDFeature-D	FM-S	FM-D	CAT-E	CAT-NT	GB-CENT
MovieLens	RMSE	0.883 (0.007) -%1.8	0.863 (0.006) +%0.4	0.877 (0.009) -%1.1	0.867 (0.006) +%0.0	0.913 (0.024) -%5.3	0.888 (0.005) -%2.4	0.886 (0.011) -%2.1	0.900 (0.006) -%3.8	0.867 (0.006)
	Time (s)	282 +1.08	1034 +6.65	68 -%49.6	66 -%51.1	73 -%45.9	60 -$55.5	77 -%42.9	54 -%60.0	135
RedHat	AUC	0.955 (0.0005) -%3.6	0.981 (0.0003) -%1.0	0.975 (0.0002) -%1.6	0.976 (0.0003) -%1.5	0.986 (0.0009) -%0.5	0.987 (0.0003) -%0.4	0.967 (0.0002) -%2.4	0.942 (0.0006) -%4.9	0.991 (0.00006)
	Time (s)	857 +%35.8	3140 +3.97	130 -%79.3	241 -%61.8	204 -%67.6	181 -%71.3	561 -%11.0	98 -%84.4	631

How does GB-CENT perform

Data Set	Metric	GBDT-OH	GBDT-CE	SVDFeature-S	SVDFeature-D	FM-S	FM-D	CAT-E	CAT-NT	GB-CENT
MovieLens	RMSE	0.883 (0.007) -%1.8	0.863 (0.006) +%0.4	0.877 (0.009) -%1.1	0.867 (0.006) +%0.0	0.913 (0.024) -%5.3	0.888 (0.005) -%2.4	0.886 (0.011) -%2.1	0.900 (0.006) -%3.8	0.867 (0.006)
	Time (s)	282 +1.08	1034 +6.65	68 -%49.6	66 -%51.1	73 -%45.9	60 -$55.5	77 -%42.9	54 -%60.0	135
RedHat	AUC	0.955 (0.0005) -%3.6	0.981 (0.0003) -%1.0	0.975 (0.0002) -%1.6	0.976 (0.0003) -%1.5	0.986 (0.0009) -%0.5	0.987 (0.0003) -%0.4	0.967 (0.0002) -%2.4	0.942 (0.0006) -%4.9	0.991 (0.00006)
	Time (s)	857 +%35.8	3140 +3.97	130 -%79.3	241 -%61.8	204 -%67.6	181 -%71.3	561 -%11.0	98 -%84.4	631

How does GB-CENT perform

Table 3: The effect of minTreeSupport and maxTreeDepth on MovieLens data set. minTreeSupport is held to be 50 when varying maxTreeDepth; maxTreeDepth is held to be 3 when varying minTreeSupport.

minTree-Support	RMSE	maxTree-Depth	RMSE
10	0.902	2	0.901
50	0.906	3	0.906
100	0.917	5	0.918
200	0.925	8	0.924
300	0.936	10	0.929
400	0.943	15	0.950

How does GB-CENT perform

Table 3: The effect of minTreeSupport and maxTreeDepth on MovieLens data set. minTreeSupport is held to be 50 when varying maxTreeDepth; maxTreeDepth is held to be 3 when varying minTreeSupport.

minTree-Support	RMSE	maxTree-Depth	RMSE
10	0.902	2	0.901
50	0.906	3	0.906
100	0.917	5	0.918
200	0.925	8	0.924
300	0.936	10	0.929
400	0.943	15	0.950

Main takeaway: Learn many shallow small trees

How does GB-CENT perform

Table 4: The effect of tree regularization on MovieLens data set. minTreeSupport=50, maxTreeDepth=3.

Regularization	minTree-Gain	Number of Accepted Trees	RMSE
AAT	N.A.	7926	0.905
VSLR	0	7606	0.906
	1	7559	0.913
	3	7441	0.921
	5	6737	0.928
	8	6375	0.945

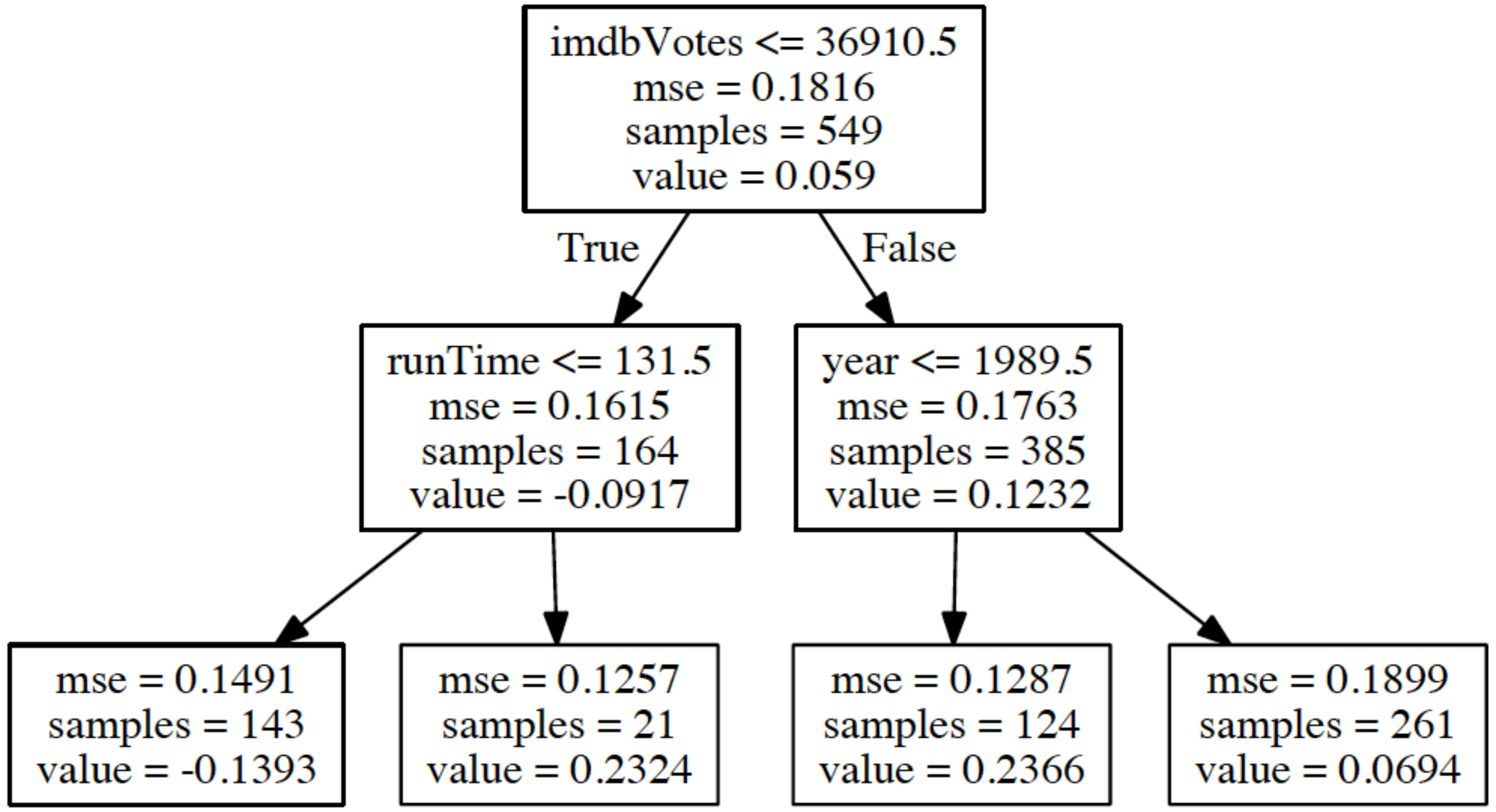
How does GB-CENT perform

Table 4: The effect of tree regularization on MovieLens data set. minTreeSupport=50, maxTreeDepth=3.

Regularization	minTree-Gain	Number of Accepted Trees	RMSE
AAT	N.A.	7926	0.905
VSLR	0	7606	0.906
	1	7559	0.913
	3	7441	0.921
	5	6737	0.928
	8	6375	0.945

Main takeaway: Learn many shallow small trees

How does GB-CENT perform



Summary

GB-CENT

- Combine Factorization Machines (handle categorical features) and GBDT (handle numerical features) together
- Combine interpretable results and high predictive power
- Achieve high performance in real-world datasets

Questions