UNIVERSITY of VIRGINIA

DSW 2019

IEEE Data Science Workshop

Statistical Learning using Hierarchical Modeling of Probability Tensors

Magda Amiridi (joint work with Nikos Kargas, Nikos Sidiropoulos)

June 3, 2019

Motivation

Real data is complex (**high dimensional **+ seemingly unstructured**)**

Main goal : → Probability Mass Function (**PMF**) **estimation**

Given discrete variables $X_1, \ldots, X_N$, construct $\widehat{P}_{X_1,\ldots,X_N}(i_1, \ldots, i_N)$

based on realizations sampled from the true PMF.

- **Why?**
  - Derive optimal estimators
  - Impute missing data
  - Anomaly detection
  - …

Our Contribution

Challenge : → Curse of Dimensionality

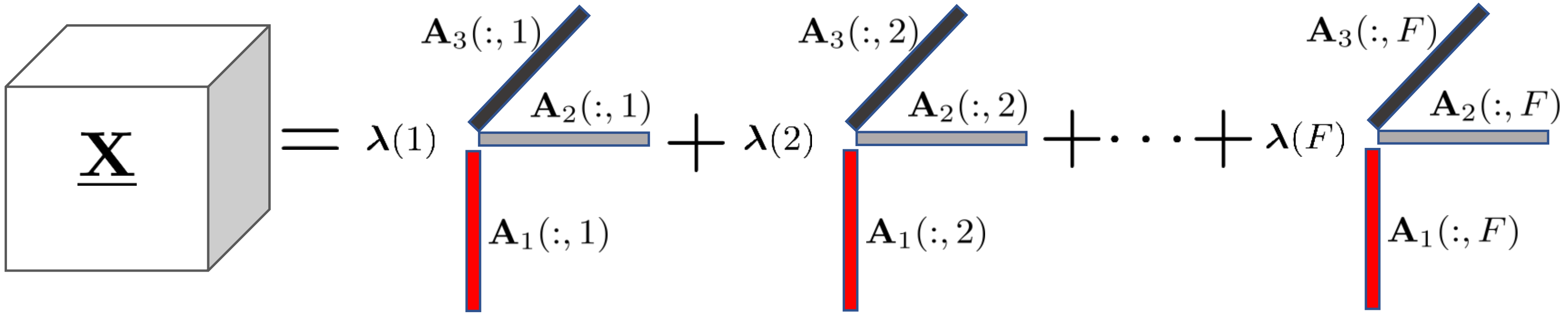
Joint PMF estimation is often considered impossible

(10 variables, 10 values each → 10^{10} parameters)

We will present:

- Effective modeling of a joint PMF using **hierarchical tensor decomposition**
- Leveraging the mere definition of **conditional probability**
  - **Parallelization** - fast computations (no data sharing, smaller subproblems)
  - **Better modeling capabilities** (regional low-rank structures)
  - **Flexibility** (e.g., control resolution level)

Canonical Polyadic Decomposition



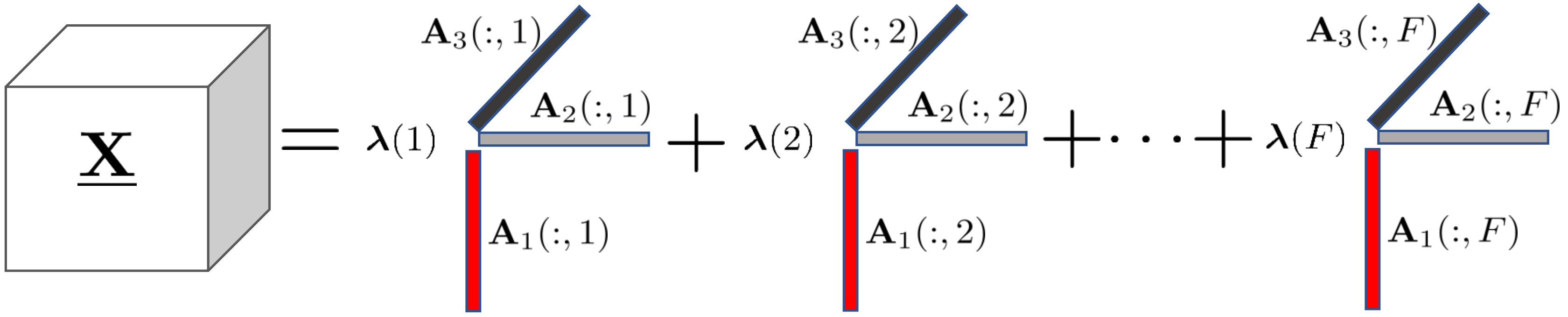
- An N-way tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is a multidimensional array whose elements are indexed by N indices.
- Any tensor can be decomposed as a sum of F rank-1 tensors.

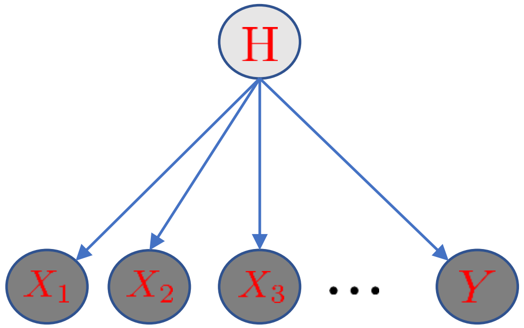
$$\underline{\mathbf{X}} = \sum_{f=1}^{F} \boldsymbol{\lambda}(f) \mathbf{A}_1(:, f) \circ \mathbf{A}_2(:, f) \circ \cdots \circ \mathbf{A}_N(:, f)$$

$$\underline{\mathbf{X}}(i_1, i_2, \ldots, i_N) = \sum_{f=1}^{F} \boldsymbol{\lambda}(f) \prod_{n=1}^{N} \mathbf{A}_n(i_n, f)$$

$$\underline{\mathbf{X}} = [\![\boldsymbol{\lambda}, \mathbf{A}_1, \ldots, \mathbf{A}_N]\!]$$

Probability Tensors – Naive Bayes model





- A joint PMF of $X_1, \ldots, X_N$ can be modeled as a **probability tensor** $\underline{\mathbf{X}}$, where:
  - Size of each dimension ⟶ the alphabet size $I_1, \ldots, I_N$
  - The indexed elements ⟶ $\underline{\mathbf{X}}(i_1, \ldots, i_N) = P_{X_1,\ldots,X_N}(i_1, \ldots, i_N)$

- *Every joint PMF admits a naive Bayes representation via the CPD* [Kargas et al. 2018]

$$P_{X_1,\ldots,X_N}(i_1, \ldots, i_N) = \sum_{f=1}^{F} P_H(f) \prod_{n=1}^{N} P_{X_n|H}(i_n|f)$$

Model PMF with a single large tensor?

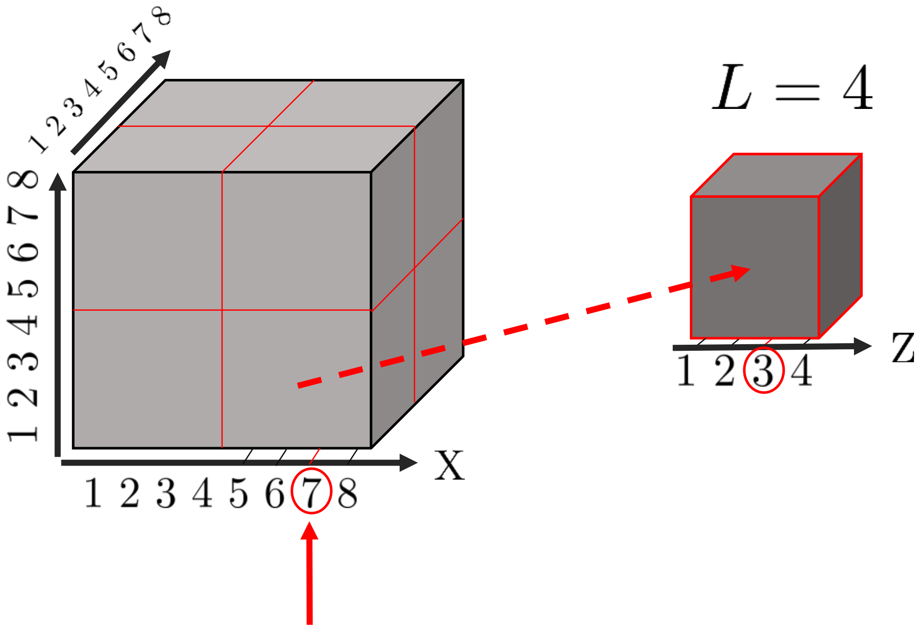
Advantage

- The number of free parameters in the tensor $\underline{\mathbf{X}} \longrightarrow \mathcal{O}(I^N)$
  Approximating by a low-rank CPD , number of free parameters $\longrightarrow \mathcal{O}(NIF)$

Considerations

- Large alphabet (discrete, finely-quantized continuous random variables)
  $\longrightarrow$ **high memory / computational complexity**
- PMF tensor usually contains local all-zero regions
- Not flexible – only 1 parameter to control

Decomposition into coarse / fine PMF



$\Pr(X = 7) = \Pr(4 < X \leq 8)\Pr(Z = 3)$
$\ell(7) = 2, r(7) = 3$

We define the following two mappings:

$$\ell(i) := \lceil \tfrac{i}{L} \rceil \in \{1, \ldots, \lceil \tfrac{I}{L} \rceil\},$$
$$r(i) := i - L(\ell(i) - 1) \in \{1, \ldots, L\},$$

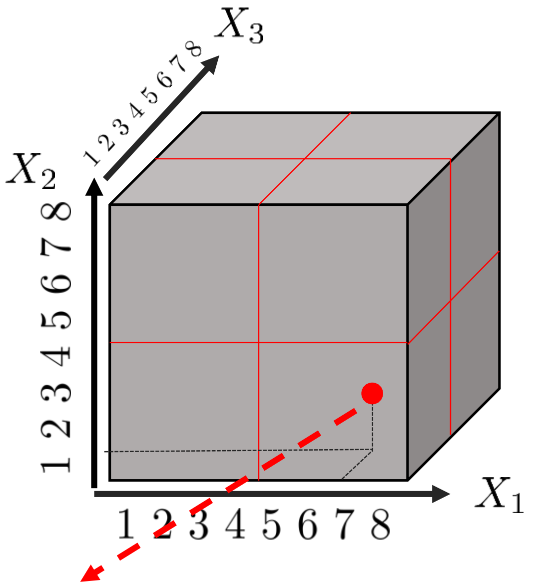
then $i = L(\ell(i) - 1) + r(i)$

$$\begin{aligned}\Pr(X = i) &= \Pr(\ell(X) = \ell(i), r(X) = r(i)) \\ &= \Pr(\ell(X) = \ell(i))\Pr(r(X) = r(i) \mid \ell(X) = \ell(i)).\end{aligned}$$

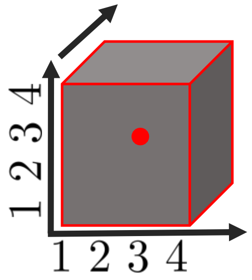
Defining, $Y := \ell(X)$ and $Z := r(X)$

$$P_X(i) = P_Y(\ell(i))P_{Z|Y}(r(i) \mid \ell(i)).$$

Decomposition into coarse / fine PMF



Pr$(X_1 = 7, X_2 = 3, X_3 = 3)$



Extending to three random variables X_1, X_2, X_3

$$P_{X_1,X_2,X_3}(i_1,i_2,i_3) = P_{Y_1,Y_2,Y_3}(\ell(i_1),\ell(i_2),\ell(i_3))$$
$$P_{Z_1,Z_2,Z_3|Y_1,Y_2,Y_3}(r(i_1),r(i_2),r(i_3) \mid \ell(i_1),\ell(i_2),\ell(i_3)).$$

$$P(i_1,i_2,i_3) = Q(\ell(i_1),\ell(i_2),\ell(i_3)) S_{\ell(i_1),\ell(i_2),\ell(i_3)}(r(i_1),r(i_2),r(i_3)).$$

Coarse PMF

Fine PMF

- **Restrict to smaller universe**
- **Conditional distributions resolve a finer level of detail**

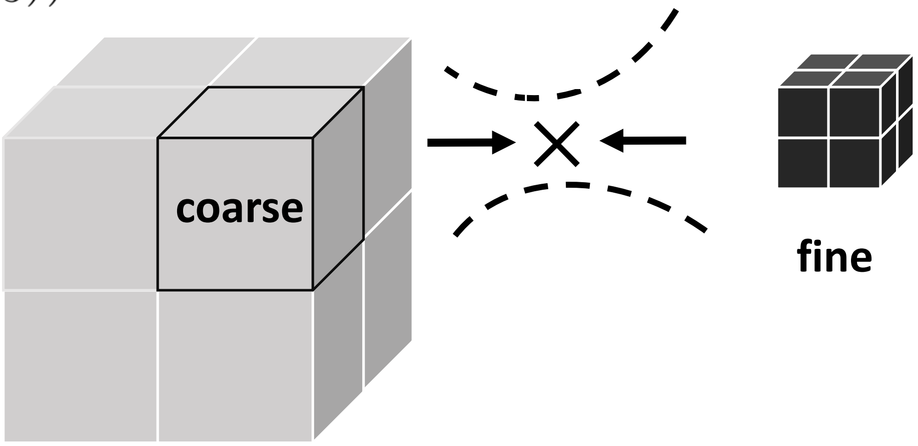
Decomposition into coarse / fine PMF

Extending to three random variables X_1, X_2, X_3

$$P(i_1, i_2, i_3) = Q(\ell(i_1), \ell(i_2), \ell(i_3)) S_{\ell(i_1),\ell(i_2),\ell(i_3)}(r(i_1), r(i_2), r(i_3)).$$

$\lceil \frac{I}{L} \rceil^3$ L^3

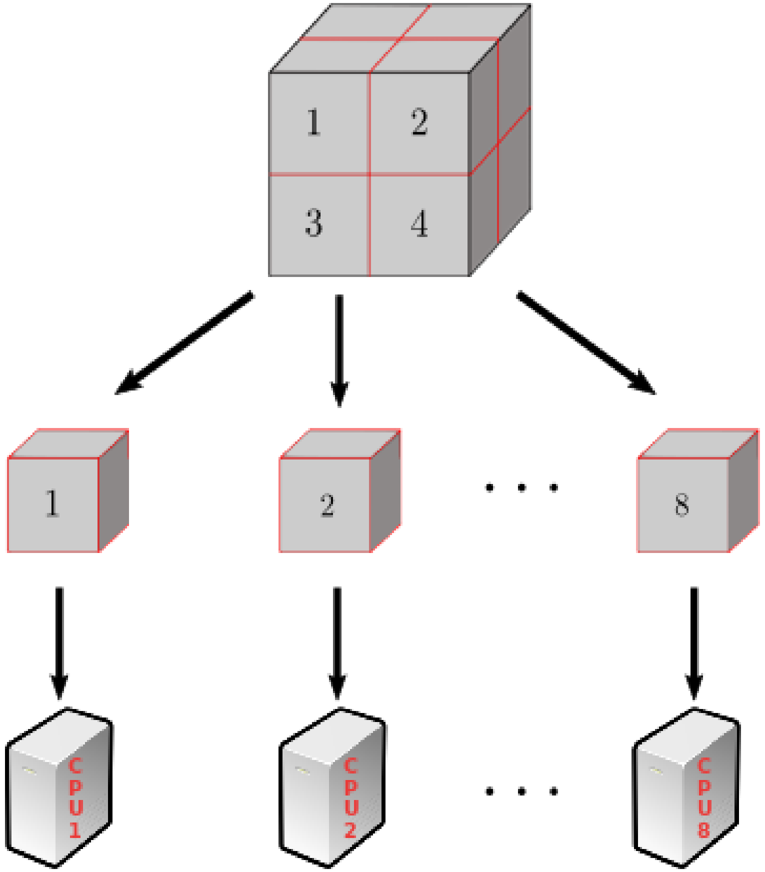
low-resolution tensor **refinement tensor**



$$Q(\ell(i_1), \ell(i_2), \ell(i_3)) = \sum_{f=1}^{F} \boldsymbol{\lambda}(f) \mathbf{A}_1(\ell(i_1), f) \mathbf{A}_2(\ell(i_2), f) \mathbf{A}_3(\ell(i_3), f)$$

$$S_{\ell(i_1),\ell(i_2),\ell(i_3)}(r(i_1), r(i_2), r(i_3)) = \sum_{f=1}^{F^{(g)}} \boldsymbol{\lambda}^{(g)}(f) \mathbf{A}_1^{(g)}(r(i_1), f) \mathbf{A}_2^{(g)}(r(i_2), f) \mathbf{A}_3^{(g)}(r(i_3), f)$$

What have we accomplished?

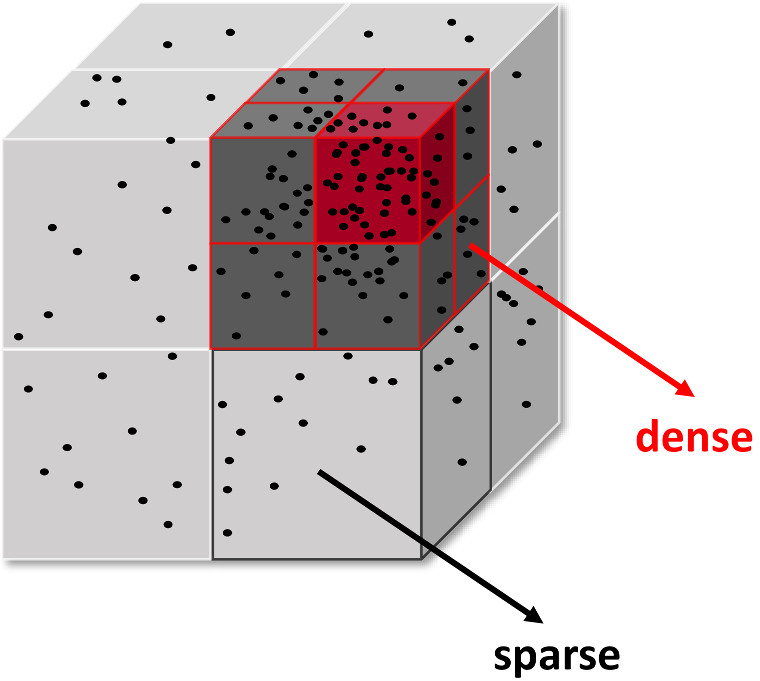


Split data in blocks → each processor decomposes a different subtensor, has access to points falling in its own block.

- All **decompositions** are computed completely in **parallel**
- Each tensor is **much smaller** (speeds up computations)
- Sub-tensors of **lower rank**

Recursive splits – a Hierarchical approach

Consider **more than two** decomposition layers :

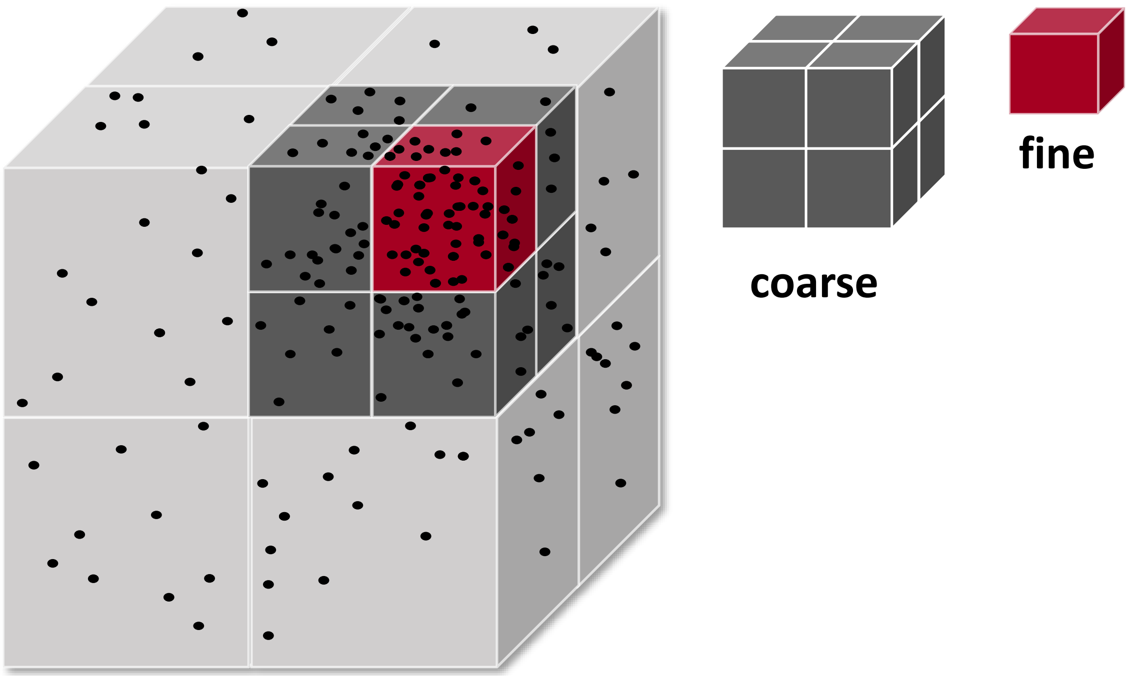


- Split each dimension $I_1, \ldots, I_n$ in half
- Assign a binary label (**dense** or **sparse**) to each block
- On the next layer, split **only** the the dense blocks

Subtensor rank assignment

- Parameters equal to single CPD → $R_{\text{fine}} = \frac{L^2 F}{I^2}$
- Higher rank to **dense** subtensors

Algorithmic Approach



$$D_{KL}(\underline{\mathbf{X}}\|\underline{\mathbf{Y}}) := \sum_{i_1,\dots,i_N} \underline{\mathbf{X}}(i_1,\dots,i_N)\log\frac{\underline{\mathbf{X}}(i_1,\dots,i_N)}{\underline{\mathbf{Y}}(i_1,\dots,i_N)}$$

$$\begin{aligned}
\min_{\boldsymbol{\lambda},\mathbf{A}_1,\dots,\mathbf{A}_N} \quad & D_{KL}\left(\underline{\mathbf{X}}\|[\![\boldsymbol{\lambda},\mathbf{A}_1,\dots,\mathbf{A}_N]\!]\right) \\
\text{subject to} \quad & \boldsymbol{\lambda} \geq \mathbf{0}, \\
& \mathbf{1}^T\boldsymbol{\lambda} = 1, \\
& \mathbf{A}_n \geq \mathbf{0},\ \ n = 1\dots N, \\
& \mathbf{1}^T\mathbf{A}_n = \mathbf{1}^T,\ \ n = 1,\dots,N,
\end{aligned}$$

- Approximate each **refinement tensor** + **low resolution tensor** by a CPD model

Experiments

Method	Skin	Bank notes	Activity	Shuttle	Older people
H-CPD Test	2.560	10.716	3.356	0.612	1.171
F-CPD Test	2.427	12.126	3.442	0.676	1.341

KL divergence of test set

Datasets	N	M
Skin	4	245057
Bank notes	5	1372
Activity	9	75128
Shuttle	9	58000
Older people	6	100000

- More **reliable** joint distribution
- ↑M ⟶ PMF learning is refined both in Hierarchical and Full CPD

Experiments

Method	Skin	Bank notes	Activity	Shuttle	Older people
H-CPD Test	2.560	10.716	3.356	0.612	1.171
F-CPD Test	2.427	12.126	3.442	0.676	1.341

KL divergence of test set

Datasets	N	M
Skin	4	245057
Bank notes	5	1372
Activity	9	75128
Shuttle	9	58000
Older people	6	100000

	Binary			Multiclass	
Method	Skin	Bank notes	Activity	Shuttle	Older people
SVM	92.879	98.909	65.983	90.732	90.282
Naive Bayes	92.434	87.636	70.146	90.767	91.453
Decision tree	99.934	97.818	96.664	98.008	96.152
Full model	99.634	87.818	96.701	97.767	94.725
Hierarchical	99.593	98.916	96.762	97.861	94.799

Prediction accuracy

Experiments

- Our method is always either **comparable** or **superior** to the baselines

- **Keep in mind:** **H-CPD** is a ***general tool*** for joint PMF estimation
  - Models *any* desired optimal estimator without any additional retraining
  - Handles randomly missing variables

	Binary			Multiclass	
Method	**Skin**	**Bank notes**	**Activity**	**Shuttle**	**Older people**
SVM	92.879	98.909	65.983	90.732	90.282
Naive Bayes	92.434	87.636	70.146	90.767	91.453
Decision tree	99.934	97.818	96.664	98.008	96.152
Full model	99.634	87.818	96.701	97.767	94.725
Hierarchical	99.593	98.916	96.762	97.861	94.799

Prediction accuracy

Experiments

- Our method is always either **comparable** or **superior** to the baselines

- **Keep in mind: H-CPD** is a ***general tool*** for joint PMF estimation
  - Models *any* desired optimal estimator without any additional retraining
  - Handles randomly missing variables

Missing data %	Decision tree	H-CPD
20	87.539	94.740
30	83.151	92.467
40	81.630	90.473
50	80.918	89.622

Accuracy with missing data on skin dataset.

Experiments

- Our method is always either **comparable** or **superior** to the baselines

- **Keep in mind:** **H-CPD** is a ***general tool*** for joint PMF estimation
  - Models *any* desired optimal estimator without any additional retraining
  - Handles randomly missing variables

Missing data %	DT with H-CPD	DT with mean
20	95.798	87.539
30	95.214	83.151
40	92.455	81.630
50	91.398	80.918

Accuracy with imputed data on skin dataset.

Take - home points

Novel method for joint PMF estimation

<u>**Competitive advantages**</u>

- Enables **accurate** distribution estimate
- **Faster** and at **lower complexity** due to the “divide and conquer” approach
- **Swiss knife** for multiple ML problems (classification, missing value imputation…)

Thank you!

Questions