Recent and Emerging Trends in Tabular Data Generation

Nayyar Zaidi, Yishuo Zhang, Gang Li

School of Information Technology, Deakin University, VIC 3220, Australia

Acknowledgement of the Country

Deakin would like to acknowledge the Traditional Custodians of the lands on which our University campuses are based.

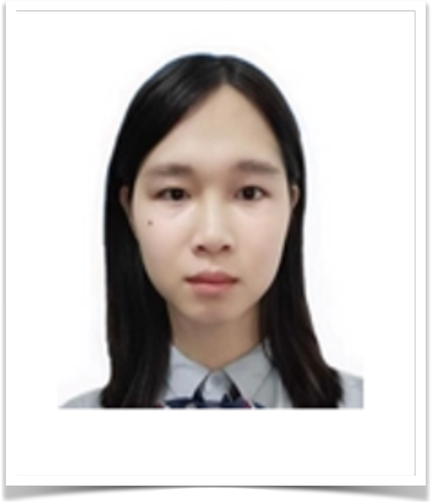
*The **Wadawurrung people** of the **Kulin Nation** on whose Country our Geelong campuses are located, the **Wurundjeri people** of the **Kulin Nation** on whose Country our Burwood campus is located and the **Peek Whurrong** people of the **Maar Nation** on whose Country our Warrnambool campus is located.*

Team

Nayyar Zaidi

Gang Li

Yishuo Zhang



Jiahui Zhou

Haiyang Hou

About the Tutorial

- Check Tutorial Website for:
  - Slides, Code, Datasets
  - Link to various relevant publications

https://www.tulip.academy/tutorials/ganblr/ICDM2022/

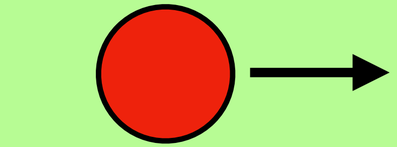
- Take notes

- Be proactive:
  - Interact and discuss
  - Ask questions and time
  - Build collaboration

- **Session I**
  - Introduction, background, preliminaries
- **Session II**
  - Historical, recent and emerging trends in tabular data generation
- **Session III** + **Accompanied Lab Session IV**
  - GANBLR
- **Session V**
  - Evaluation methods on synthetic data and interpretation
  - Applications

About this Tutorial

- Tabular Data Generation
  - Why is this important/relevant?
  - What are popular techniques?
  - What are the new trends?
  - What are the applications?



Data Generation Model

ID	Age	Salary	Location
1	23	29k	VIC
2	49	102k	NSW
...			
1000	35	91k	VIC

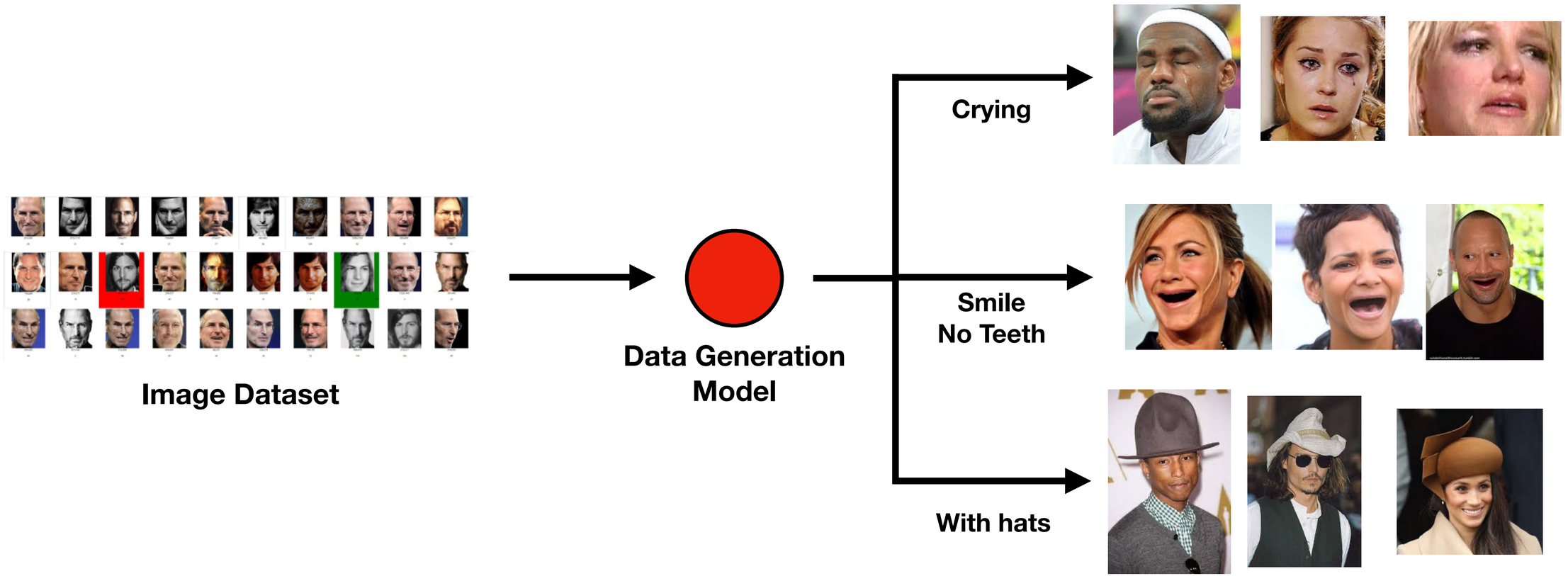
- Accompanied iPython Notebook requires following packages:
- Code for state-of-the-art tabular data generation models
- Offers a generic framework that offers to build and evaluate your own data generation algorithm
  - Plug and Play

Session I

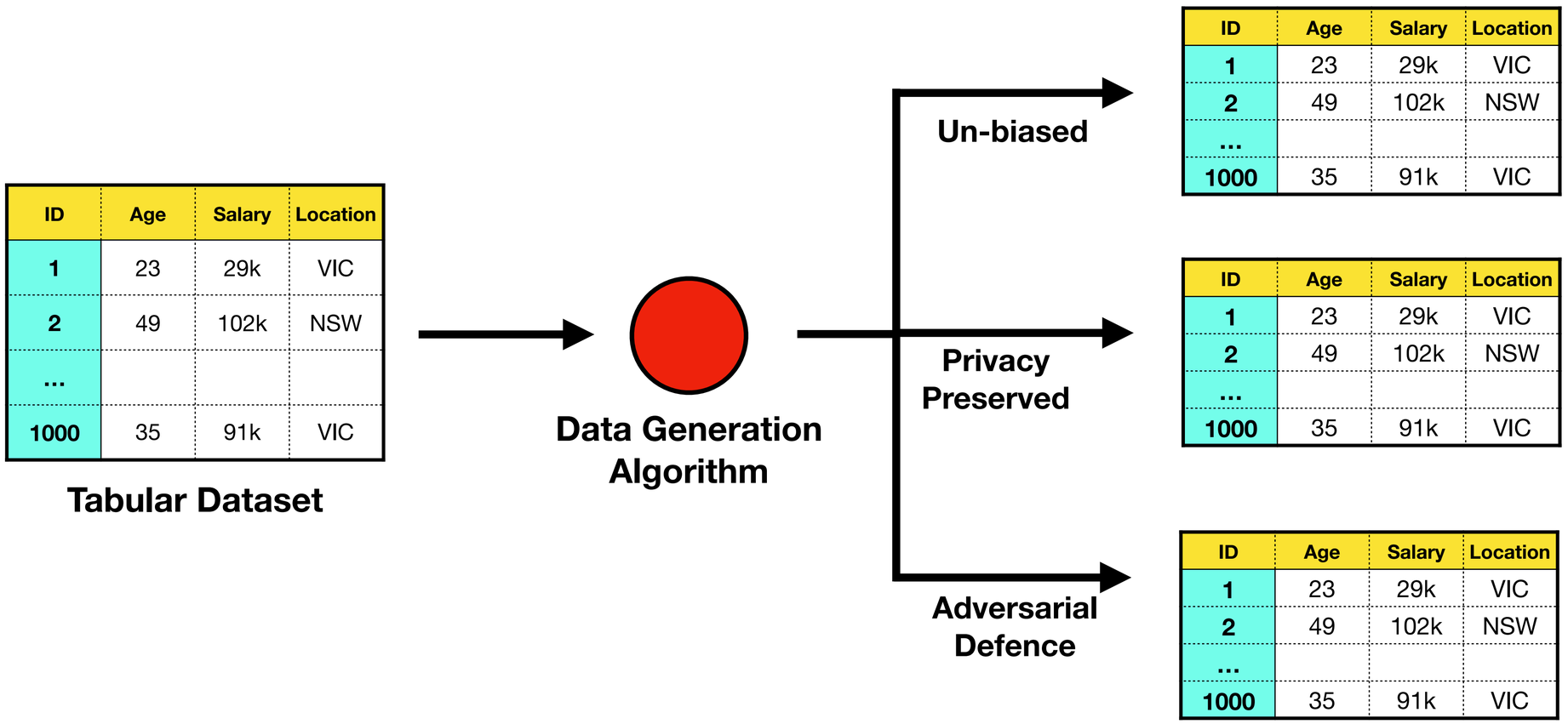
Introduction

1. Motivations (with an example)
2. Generative vs. Discriminative Models
   - Why study generative models?
3. Problem Formulation
4. Generative models taxonomy
5. Tabular Data Generation
6. Structured Data vs. Tabular Data

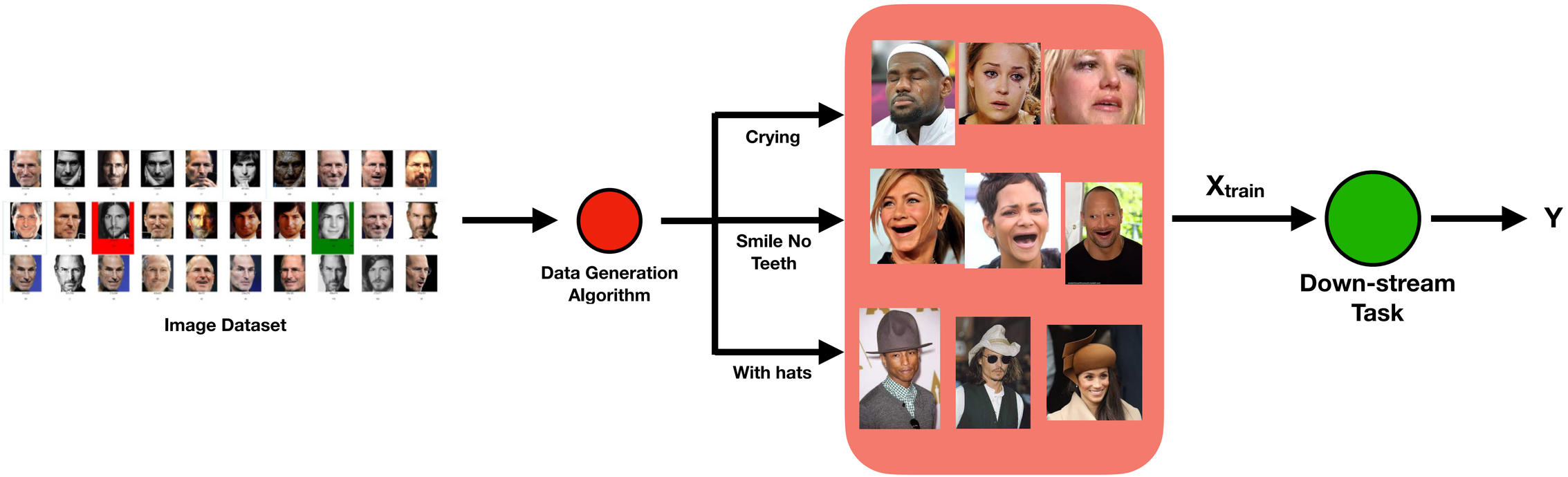
Motivations



Motivations



Motivations

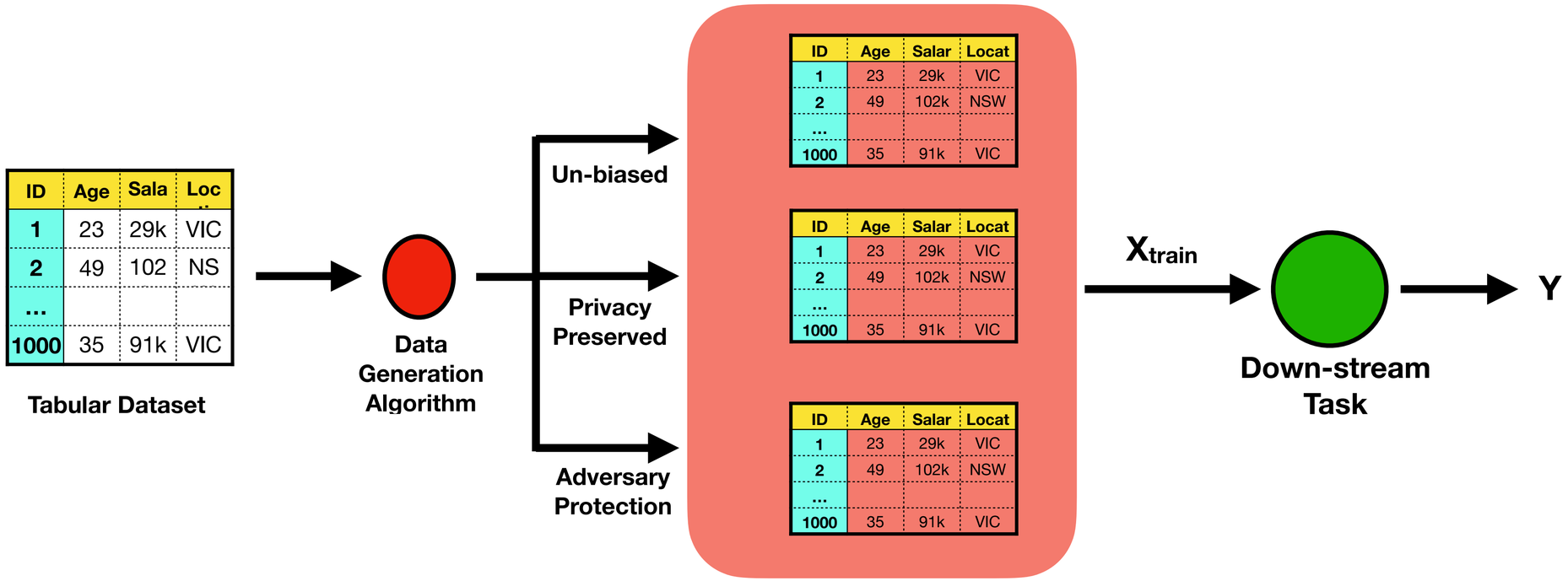


Stage 1

Stage 2

End-to-end Learning

Motivations

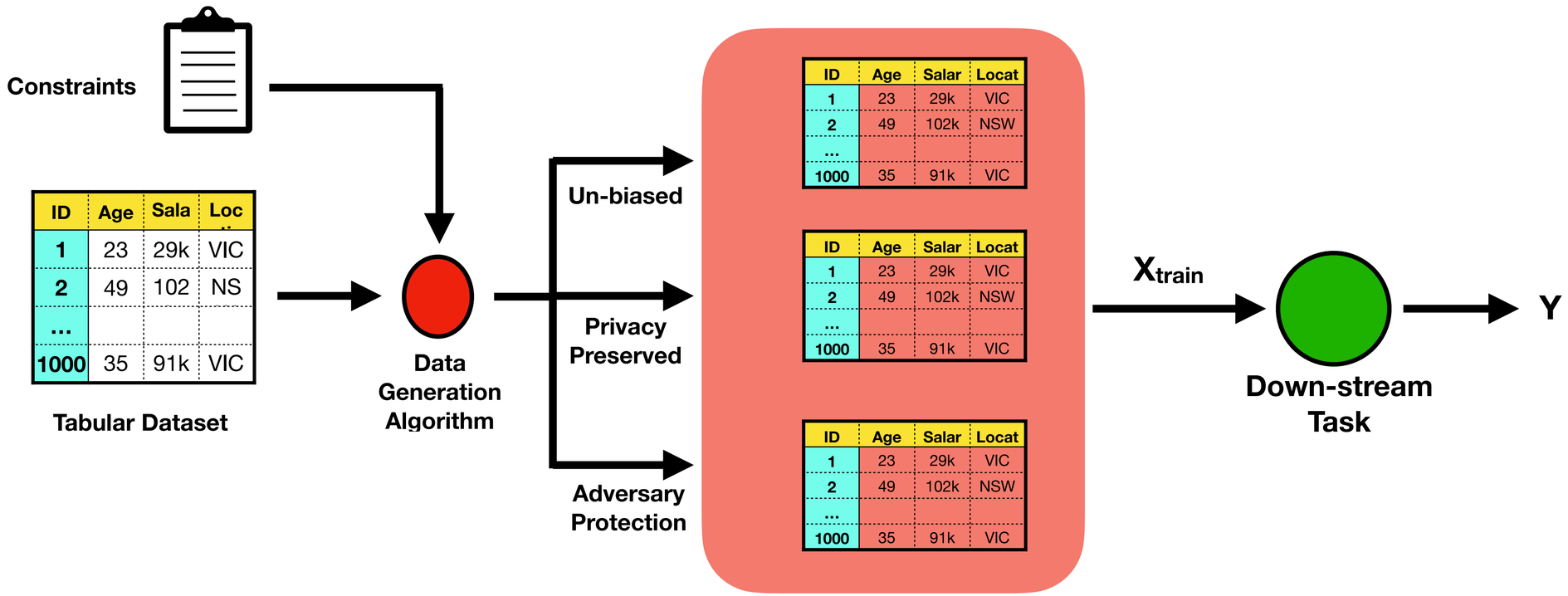


Stage 1

Stage 2

End-to-end Learning

Controlled Learning

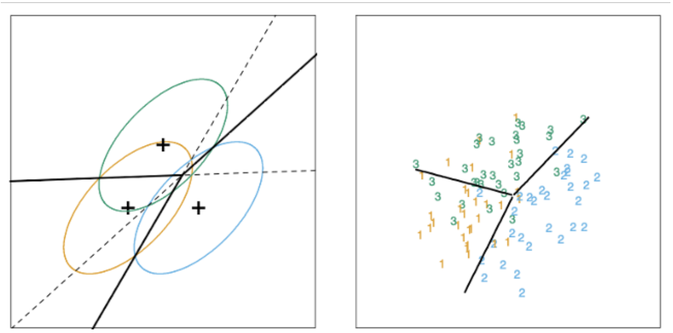


End-to-end Learning

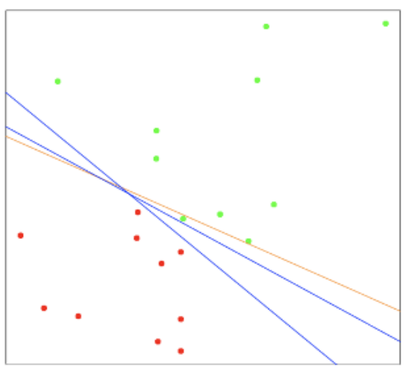
Generative Models

- Generative and Discriminative models have a long history in Machine Learning **[1]**
- Generative Models estimates: **P(x,y)**
  - Use Bayes Rule to estimate P(y|x)
  - Example model: Naive Bayes, TAN, KDB
- Whereas Discriminative Models directly estimates: **P(y|x)**
  - Example model: Logistic Regression, ANN
  - Leads to better classification performance

- Generative Models solves much deeper problem of learning a distribution
  - Fundamentally, learning is not only about distinguishing between two classes
  - But about learning distinct patterns that makes each class unique
  - Much useful in cases when no class labels are present
    - E.g., what about estimating only **P(x)**?



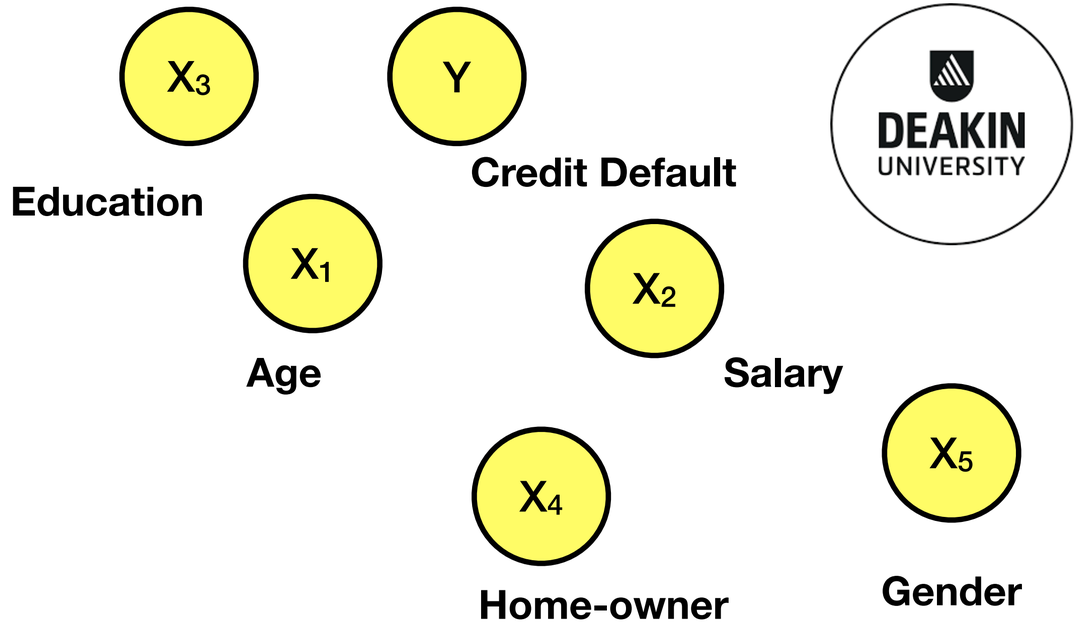
Generative Classifier [2]

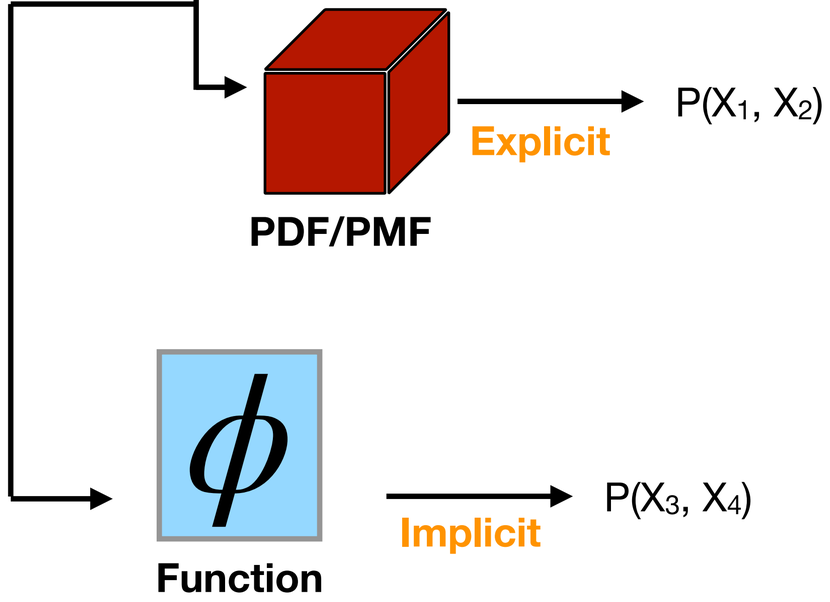


Discriminative Classifier [2]

Why P(X)?

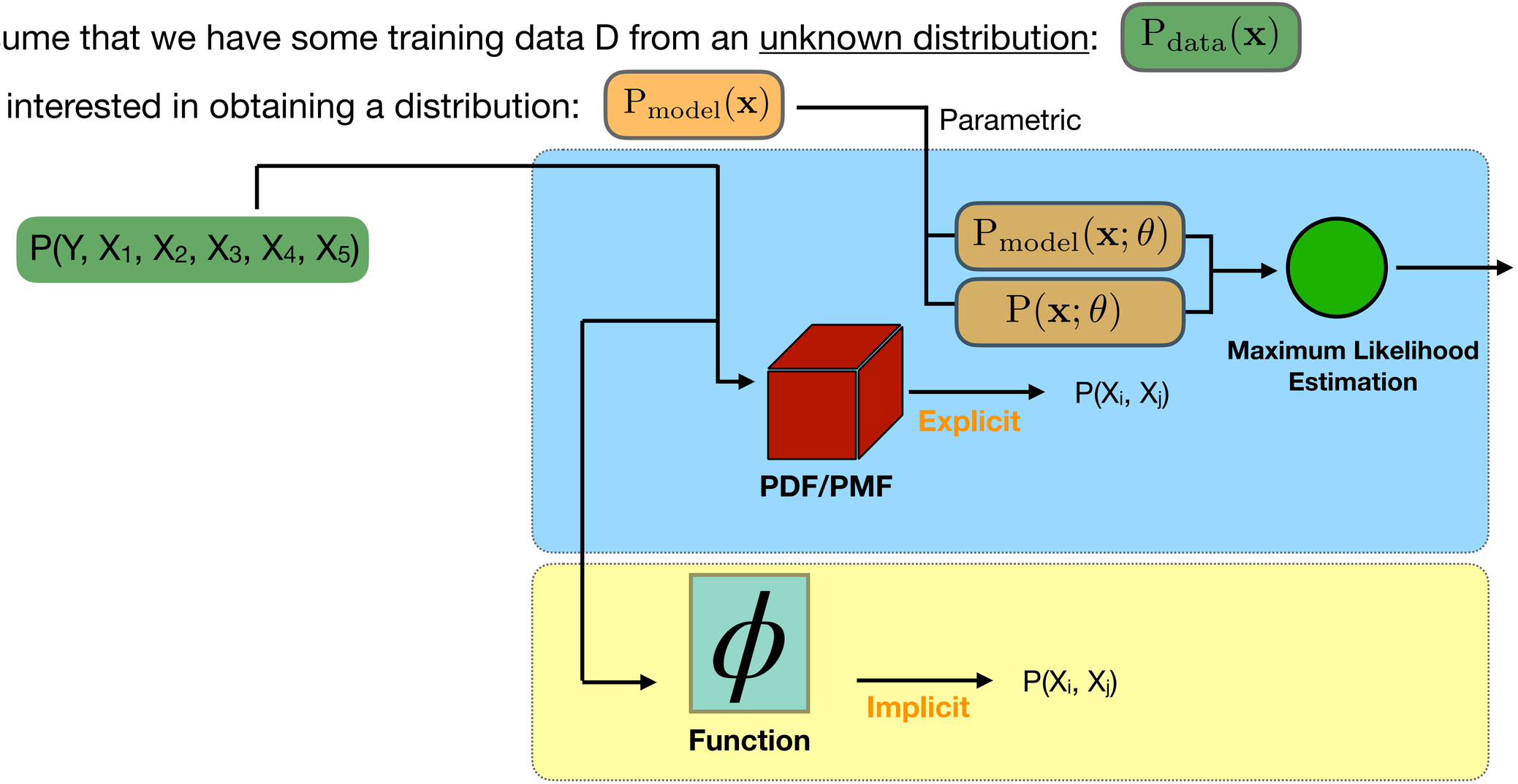
- Controlled Learning
- Much powerful analysis
  - From $P(Y, X_1, X_2, X_3, X_4, X_5)$, we can compute:
    - $P(Y \mid X_1, X_2, X_3, X_4, X_5)$
    - $P(X_1 \mid Y, X_2, X_3, X_4, X_5)$
    - ⋮
    - $P(X_5 \mid Y, X_2, X_3, X_4, X_1)$





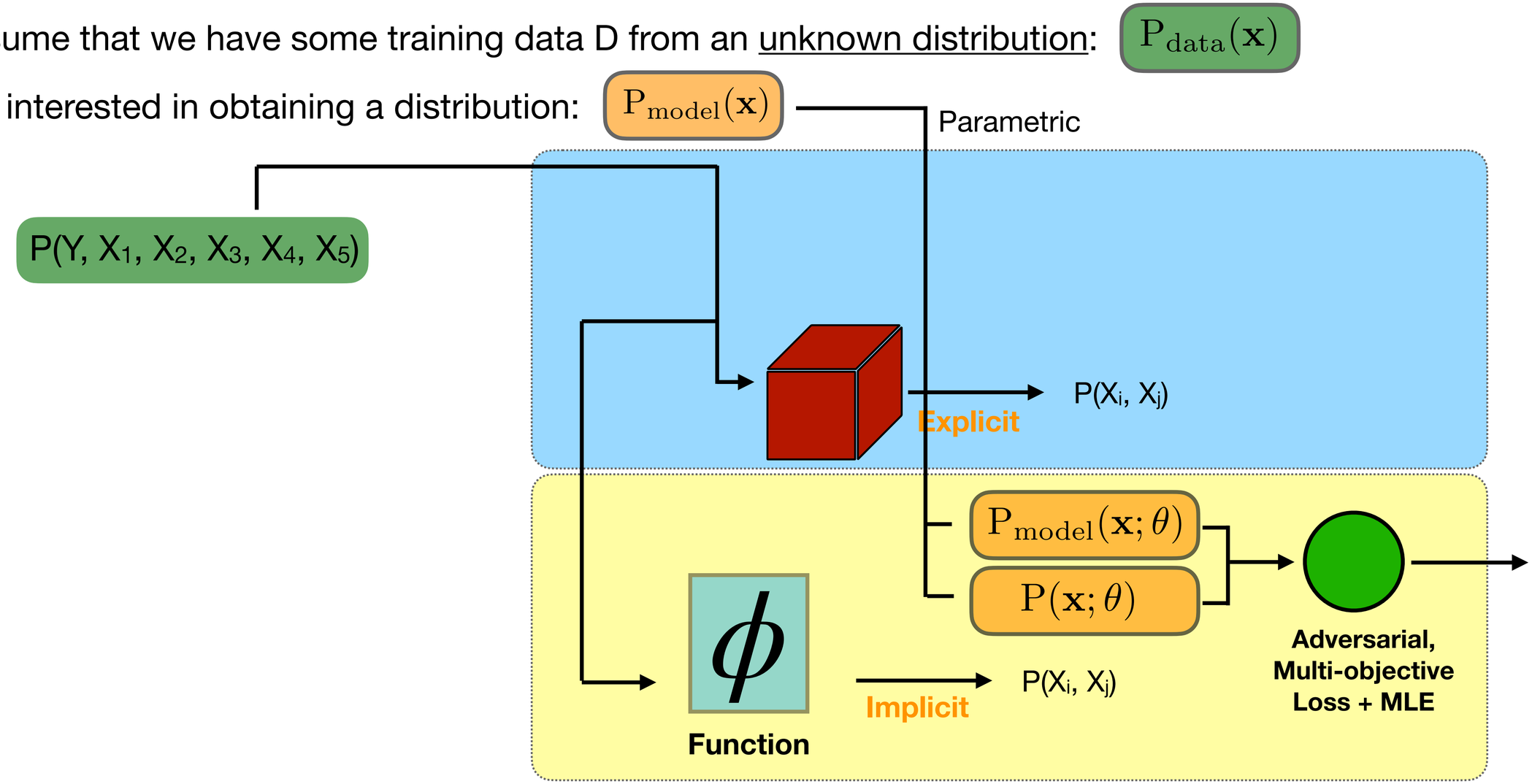
Problem Formulation

- We assume that we have some training data D from an unknown distribution: $\mathrm{P}_{\mathrm{data}}(\mathbf{x})$
- We are interested in obtaining a distribution: $\mathrm{P}_{\mathrm{model}}(\mathbf{x})$



Problem Formulation

- We assume that we have some training data D from an unknown distribution: $\mathrm{P}_{\mathrm{data}}(\mathbf{x})$
- We are interested in obtaining a distribution: $\mathrm{P}_{\mathrm{model}}(\mathbf{x})$



Maximum Likelihood Estimation (MLE)

- Principle of MLE states that choose a set of parameters — θ for the model that maximises the likelihood of the training data

- Minimizing the KL divergence between $P_{data}(\mathbf{x})$ and $P_{model}(\mathbf{x})$ is exactly equivalent to maximizing the log-likelihood of the training set

- If $P_{data}(\mathbf{x})$ lies within the same family of distributions as $P_{model}(\mathbf{x})$, the model would recover $P_{data}(\mathbf{x})$ exactly

$$\mathrm{P}(\mathbf{x};\theta) \rightarrow \prod_{i=1}^{N} \mathrm{P}(x^{(i)};\theta)$$

$$\sum_{i=1}^{N} \log \mathrm{P}(x^{(i)};\theta)$$

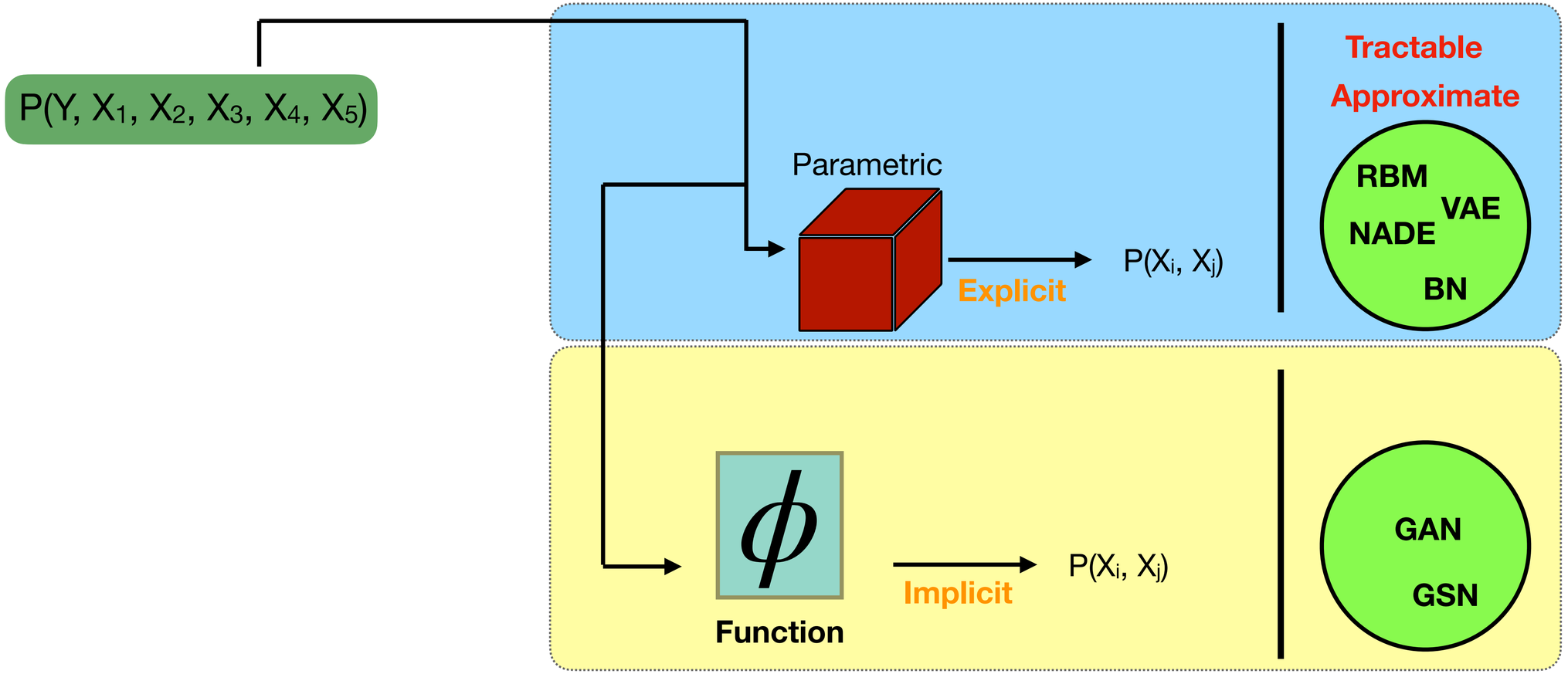
$$\theta^* = \arg\max_{\theta} \sum_{i=1}^{N} \log \mathrm{P}(x^{(i)};\theta)$$

Equivalent to

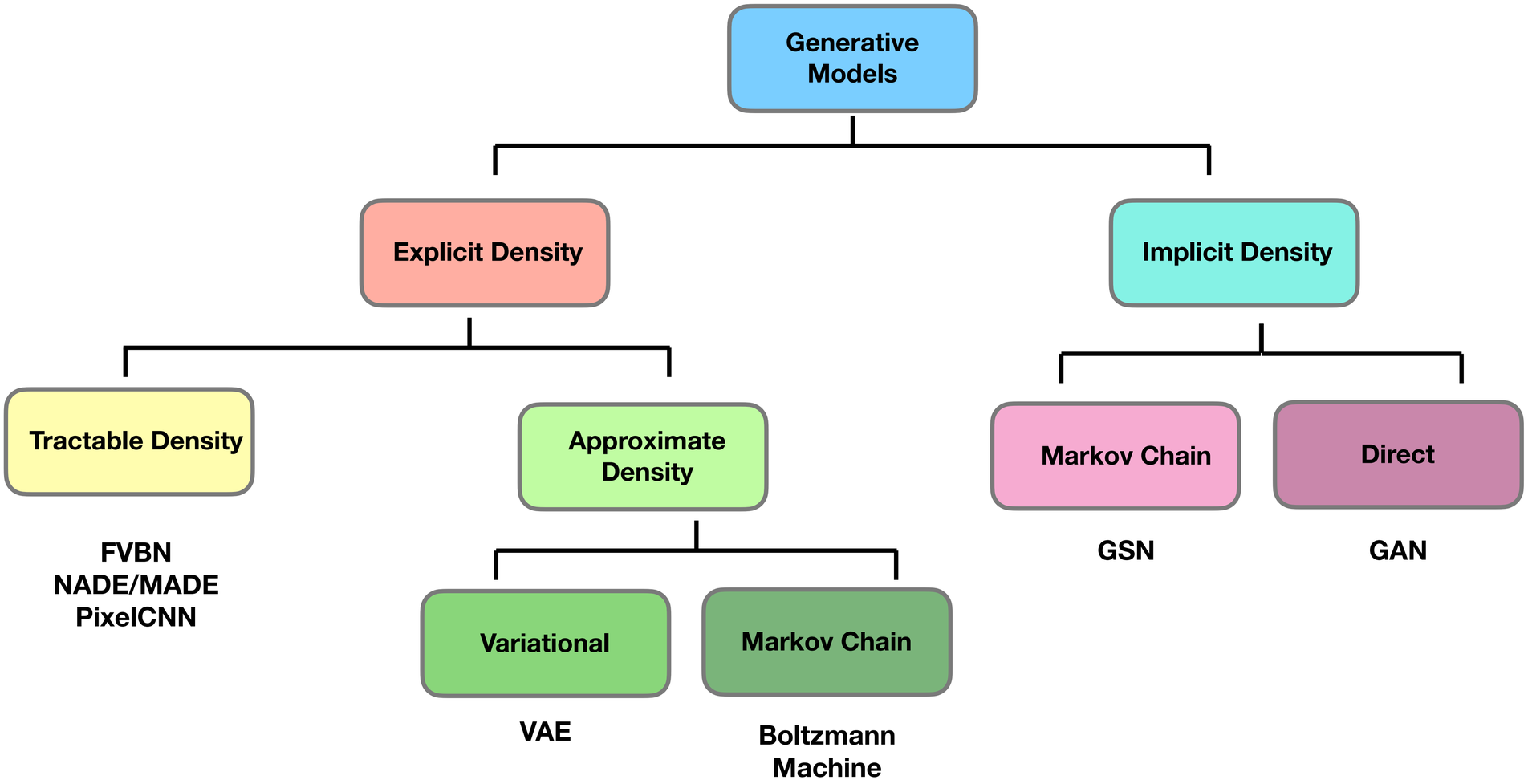
$$\theta^* = \arg\max_{\theta} \mathrm{D}_{\mathrm{KL}}(\mathrm{P}_{\mathrm{data}}(\mathbf{x})||\mathrm{P}_{\mathrm{model}}(\mathbf{x};\theta))$$

Problem Formulation

- We assume that we have some training data D from an unknown distribution: $\mathrm{P}_{\mathrm{data}}(\mathbf{x})$
- We are interested in obtaining a distribution: $\mathrm{P}_{\mathrm{model}}(\mathbf{x})$



Taxonomy



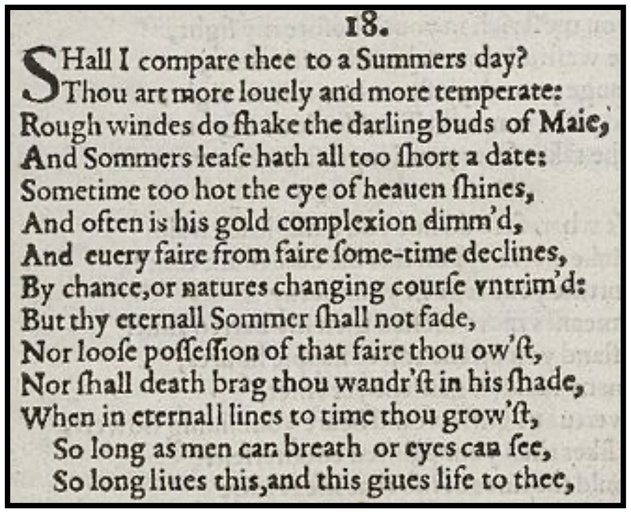
[1] Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. arXiv preprint arXiv:1701.00160.

Structured Data

- Explicit interactions exist among the features
  - E.g., pixels next to each other affect each other values

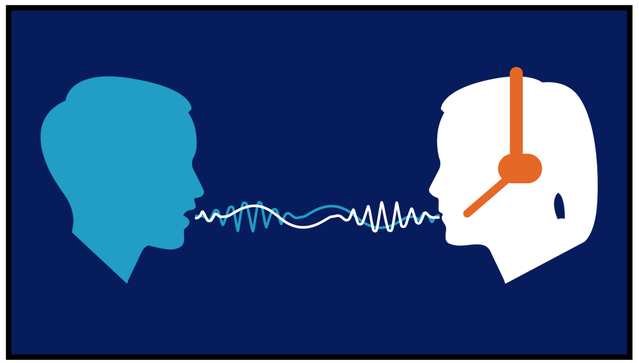
- Example:
  - Images, Text, Speech
  - Time Series

- Operations:
  - Operations exists for feature engineering and data representation, e.g., convolution



18.

SHall I compare thee to a Summers day?
Thou art more louely and more temperate:
Rough windes do ſhake the darling buds of Maie,
And Sommers leaſe hath all too ſhort a date:
Sometime too hot the eye of heauen ſhines,
And often is his gold complexion dimm'd,
And euery faire from faire ſome-time declines,
By chance,or natures changing courſe vntrim'd:
But thy eternall Sommer ſhall not fade,
Nor looſe poſſeſſion of that faire thou ow'ſt,
Nor ſhall death brag thou wandr'ſt in his ſhade,
When in eternall lines to time thou grow'ſt,
So long as men can breath or eyes can ſee,
So long liues this,and this giues life to thee,



Structured Data

Data = 

Transforming n x n matrix into 1 x n^2 vector

	Attribute 1	Attribute 2	Attribute 3	...	Attribute l	Attribute l + 1	Attribute l + 2	...	Attribute m	Attribute m + 1	Attribute m + 2	...	Attribute n - 2	Attribute n - 1	Location n
Datum 1	0	0	0	...	0	0	0	...	1	1	0	...	0	0	0
Datum 2	0	0	1	...	0	0	0	...	0	0	0	...	0	0	0
Datum 3	0	0	0	...	0	0	0	...	0	0	0	...	0	0	0
Datum 4	0	0	0	...	0	0	0	...	0	0	0	...	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Datum N	0	0	0	...	0	0	0	...	0	0	0	...	0	0	0

Correlated Features

Data =

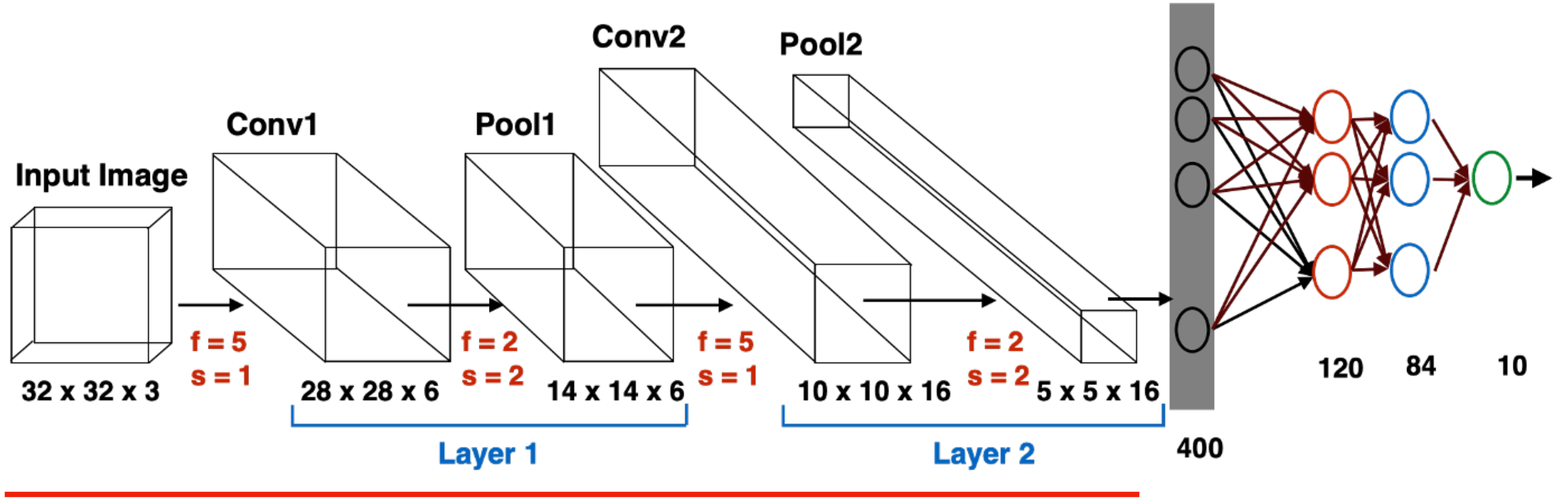
Machine Learning is super cool, we all should build awesome models.
I really appreciate your help.
Thanks so much... hey?
....
How is it going? We should have coffee sometime.

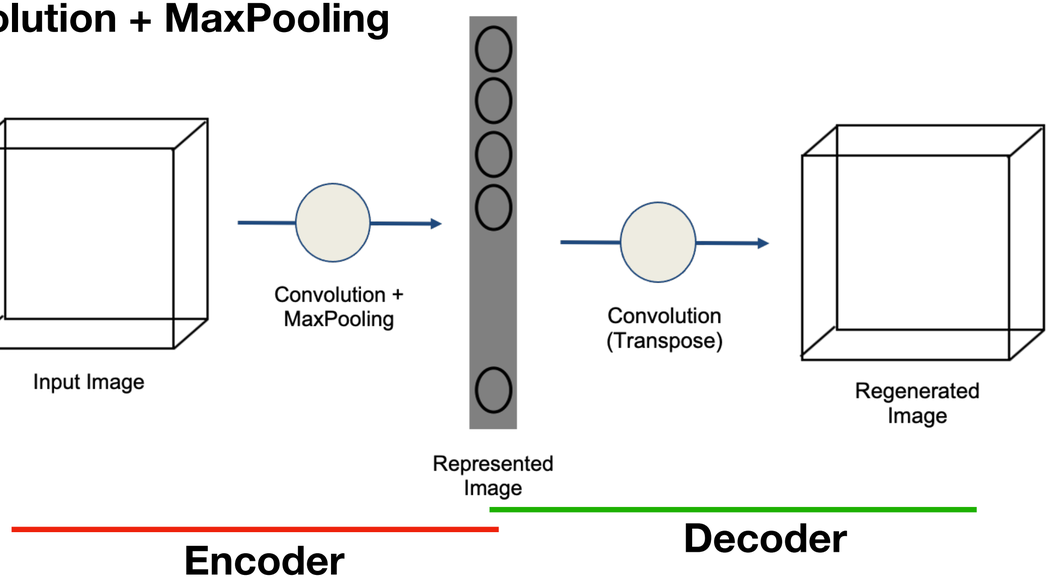
Transforming sentences into tabular form — representing as vector

		A		Abacus		...	Costume		...	Drastic		Machine		...	Zyzzyva	
		Attribute 1	Location 1	Attribute 2	Location 2	...	Attribute k	Location k	...	Attribute m	Location m	Attribute m+1	Location m+1	...	Attribute n	Location n
Sentence1	Datum 1	0	0	0	0	...	0	0	...	1	1	0	0	...	0	0
Sentence2	Datum 2	0	0	1	0	...	0	0	...	0	0	0	0	...	0	0
Sentence3	Datum 3	0	0	0	0	...	0	0	...	0	0	0	0	...	0	0
Sentence4	Datum 4	0	0	0	0	...	0	0	...	0	0	0	0	...	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
SentenceN	Datum N	0	0	0	0	...	0	0	...	0	0	0	0	...	0	0

Positional information

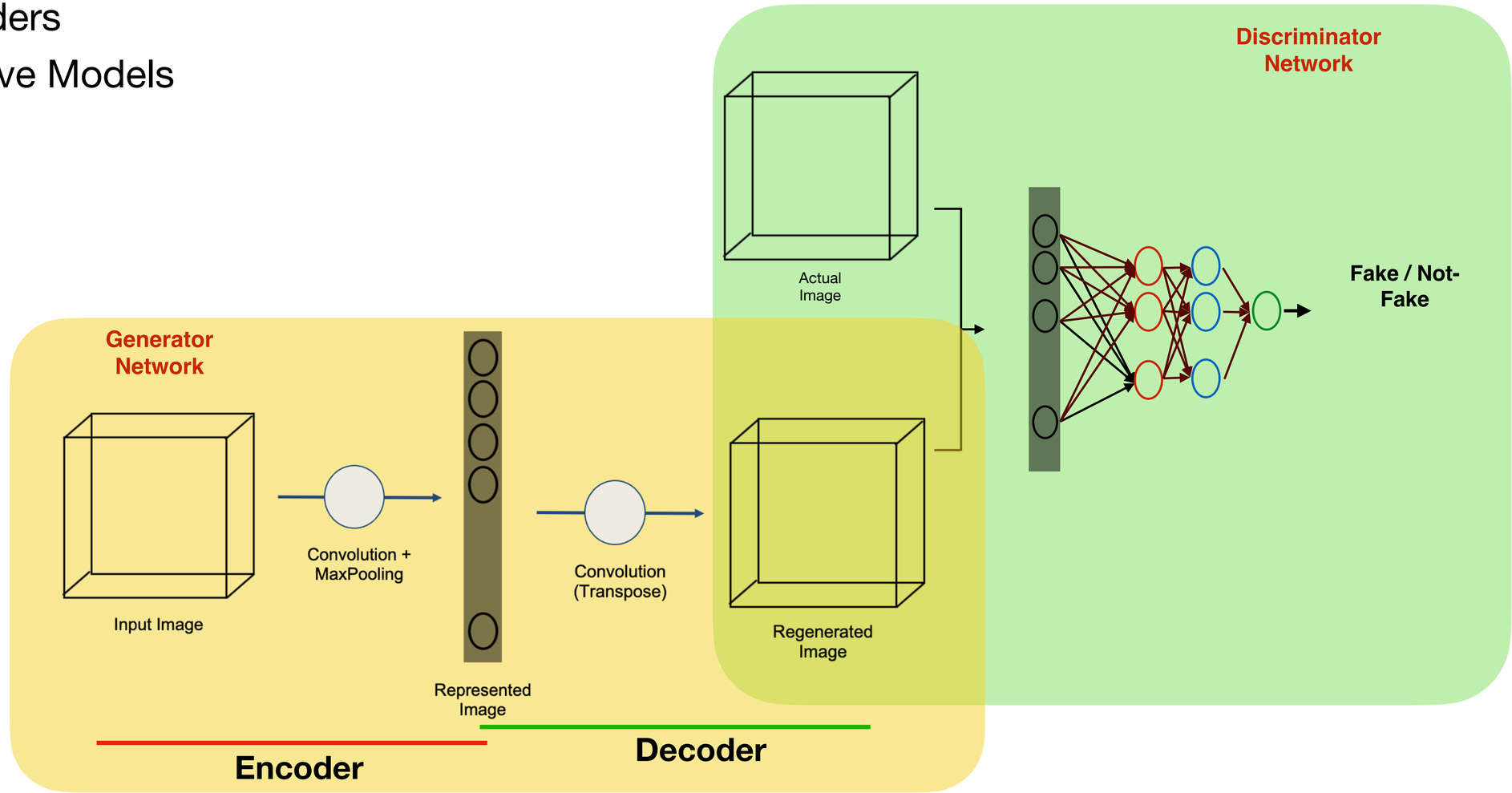
Models for Structured Data





Models for Structured Data

- Generative Adversarial Networks
- Variational AutoEncoders
- Neural Auto-Regressive Models
- Etc



Tabular Data

- No explicit interactions exist among the features
  - E.g., age has no present correlation with salary or credit-score, unless specified by the domain expert

- Example:
  - Customer's Data (Retail, Banking)
  - Census Data
  - Etc.

- Operations:
  - Unlike Convolutions, no explicit operations exists

		ID	Age	Gender	Salary	Suburb	Education	Home Owner		voteLabor
		Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5	Attribute 6	Attribute 7	...	Attribute n
Sam	Datum 1	1	19	M	Low	SE	Grad	Y	...	0
Adam	Datum 2	2	25	F	High	W	Grad	Y	...	0
Alan	Datum 3	3	67	?	Medium	IC	PhD	N	...	1
Bart	Datum 4	4	20	F	High	City	School	N	...	1
Gru	Datum 5	5	56	M	Medium	W	High-School	Y	...	0
Brett	Datum 6	6	39	?	Low	NE	Grad	N	...	1
Lucy	Datum 7	7	10	?	Low	SE	?	N	...	0
	...	...	...	...	...	...	...	...	...	...
Sara	Datum N	N	20	M	Low	E	Grad	N	...	0

Tabular Data

	Attribute 1	Attribute 2	Attribute 3	...	Attribute n
Datum 1	1	19	12	...	0
Datum 2	2	25	3	...	0
Datum 3	3	67	5	...	1
Datum 4	4	20	6	...	1
Datum 5	5	56	0	...	0
Datum 6	6	39	1	...	1
Datum 7	7	10	10	...	0
...	...	...	...	...	...
Datum N	100	20	3	...	0

	Items
1	{Bread, Soda, Eggs, Muffins}
2	{Bread, Butter, Soda}
3	{Soda, Onions, Chips, Potatoes, Shampoo}
4	{Diaper, Milk, Soda}

	Team	Soccer	Barcelona	League	Champion	Lost	Winning	...	Goals
Document 1	1	1	3	4	4	0	0	...	0
Document 2	0	0	1	0	2	0	0	...	0
Document 3	0	1	0	1	1	1	1	...	0
Document 4	0	1	0	0	0	0	1	...	1

		ID	Age	Gender	Salary	Suburb	Education	Home Owner		voteLabor
		Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5	Attribute 6	Attribute 7	...	Attribute n
Sam	Datum 1	1	19	M	Low	SE	Grad	Y	...	0
Adam	Datum 2	2	25	F	High	W	Grad	Y	...	0
Alan	Datum 3	3	67	?	Medium	IC	PhD	N	...	1
Bart	Datum 4	4	20	F	High	City	School	N	...	1
Gru	Datum 5	5	56	M	Medium	W	High-School	Y	...	0
Brett	Datum 6	6	39	?	Low	NE	Grad	N	...	1
Lucy	Datum 7	7	10	?	Low	SE	?	N	...	0
	...	...	...	...	...	...	...	...	...	...
Sara	Datum N	N	20	M	Low	E	Grad	N	...	0

Question

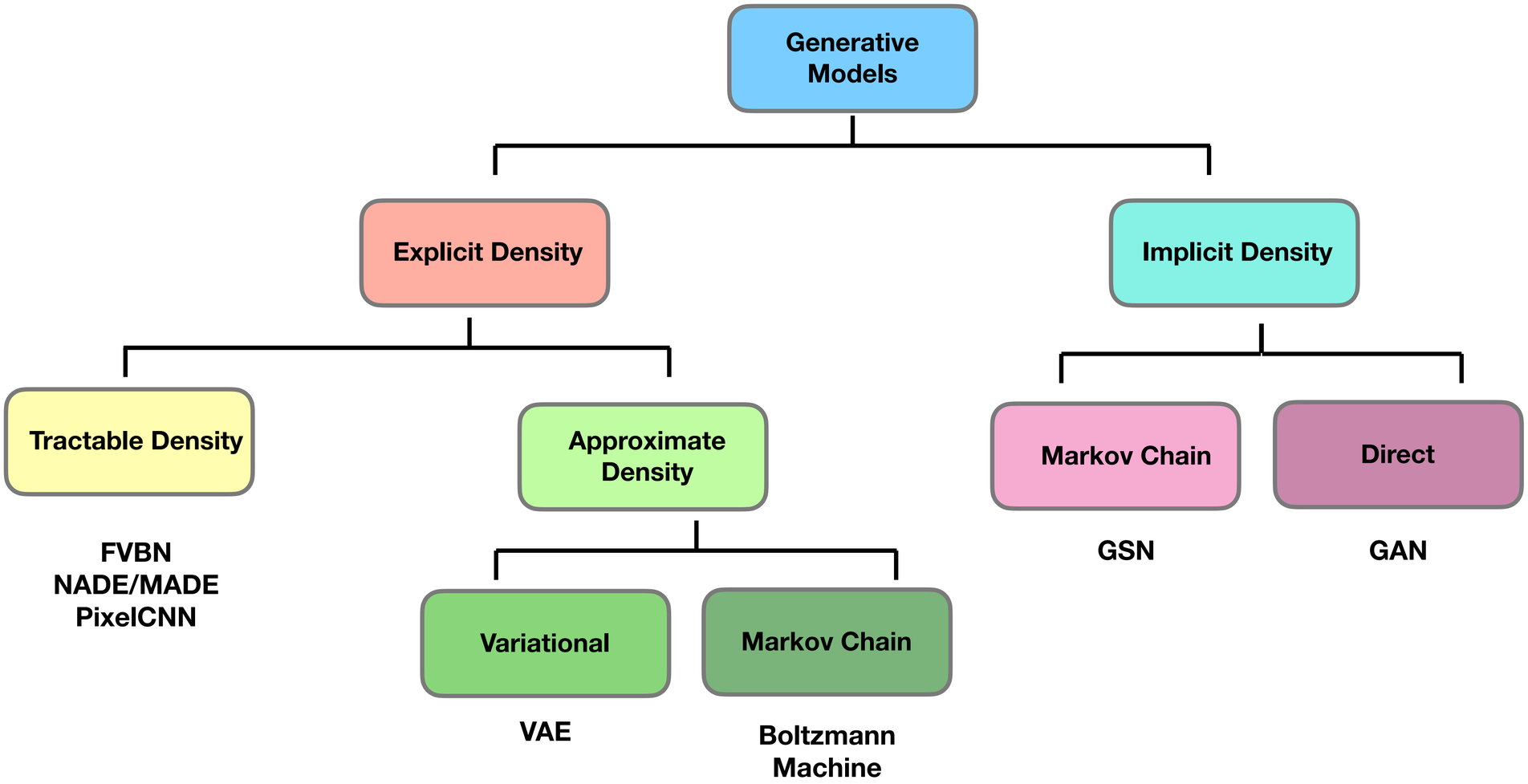
- Can recent advancements (especially deep and adversarial learning) in structured data generation be used for tabular data generation?

- **Yes**

  - In fact there has already been several papers that target tabular data generation based on techniques that are inspired from GANs
  - However, plain application of GANs to tabular datasets is challenging as operation of convolution is not defined for tabular data
  - In the next section, we will visit various recent GAN-inspired methods

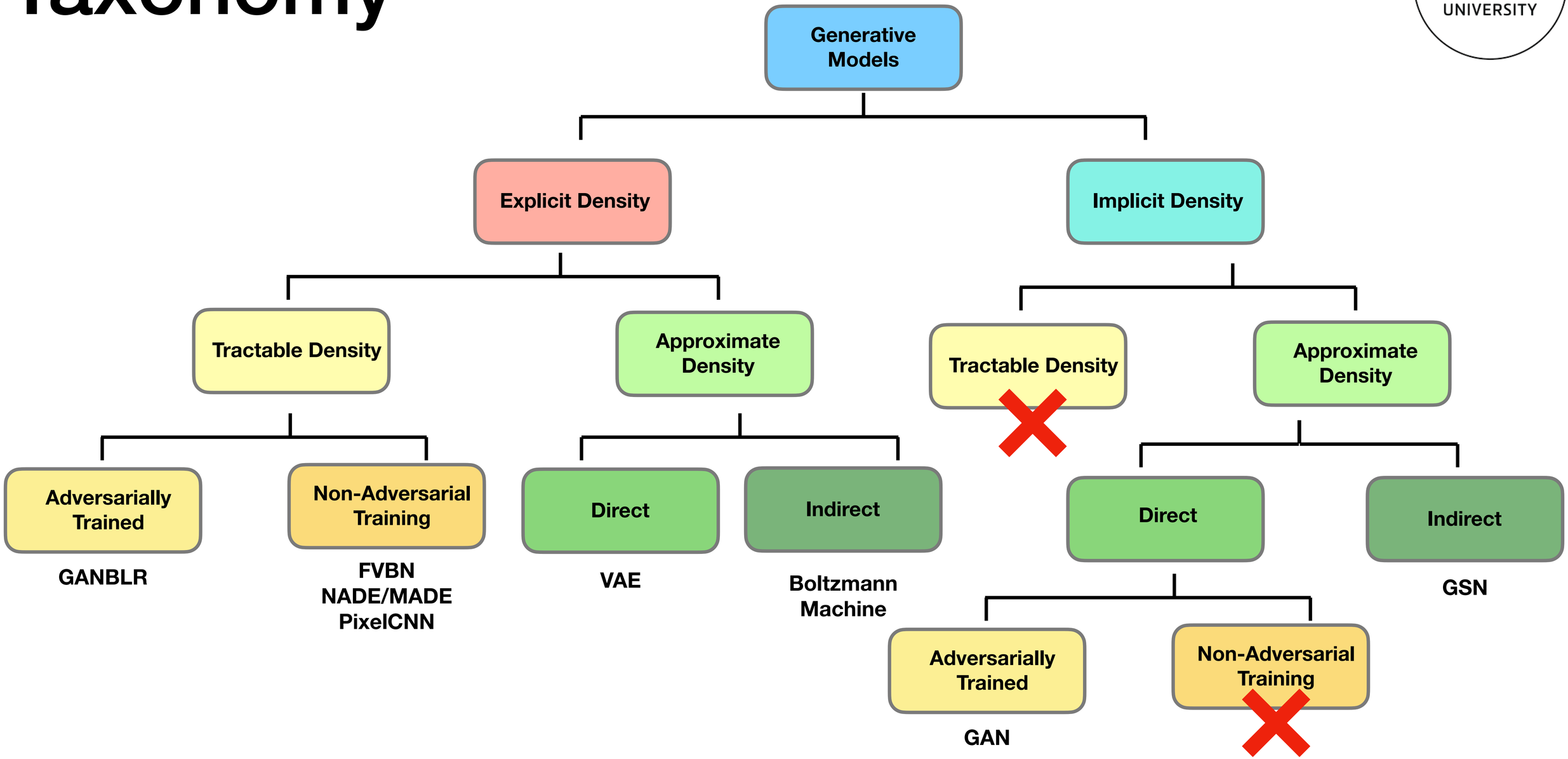
Session II

Taxonomy

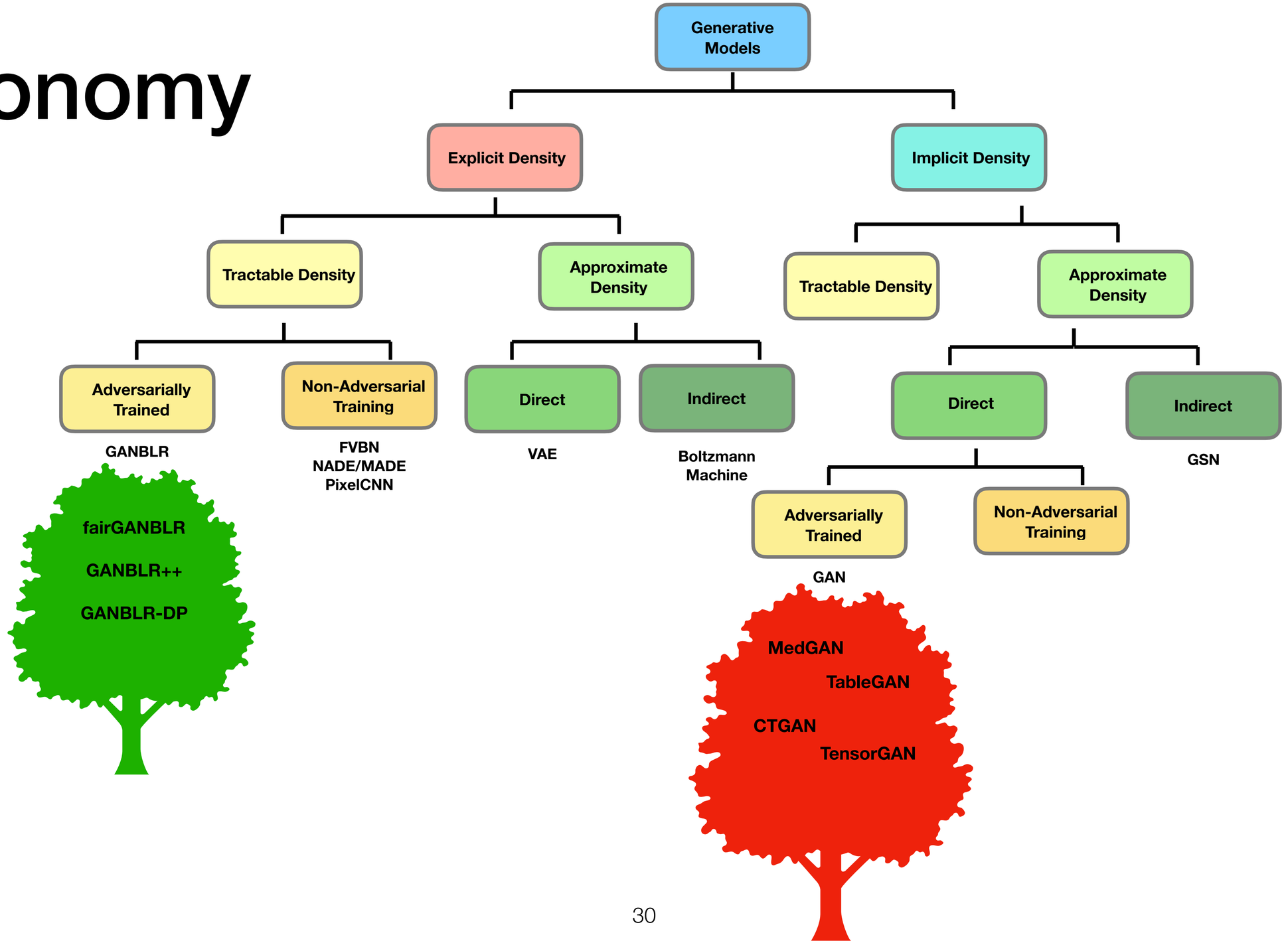


[1] Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. arXiv preprint arXiv:1701.00160.

Taxonomy


Generative Models
Explicit Density
Implicit Density
Tractable Density
Approximate Density
Tractable Density
Approximate Density
Adversarially Trained
Non-Adversarial Training
Direct
Indirect
Direct
Indirect
GANBLR
FVBN
NADE/MADE
PixelCNN
VAE
Boltzmann Machine
Adversarially Trained
Non-Adversarial Training
GSN
GAN

Taxonomy


Generative Models
Explicit Density
Implicit Density
Tractable Density
Approximate Density
Tractable Density
Approximate Density
Adversarially Trained
Non-Adversarial Training
Direct
Indirect
Direct
Indirect
GANBLR
FVBN
NADE/MADE
PixelCNN
VAE
Boltzmann Machine
GSN
Adversarially Trained
Non-Adversarial Training
GAN
fairGANBLR
GANBLR++
GANBLR-DP
MedGAN
TableGAN
CTGAN
TensorGAN

Generative Models

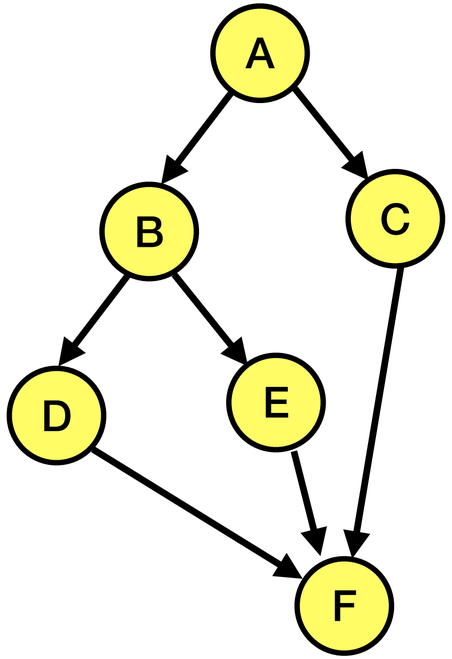
- Older Techniques
  - Bayesian Networks **[3]**
  - Restricted Boltzmann Machines **[4]**
  - Variational Auto-Encoders **[5]**
  - NADE **[6]**

- Existing Trends
  - Generative Adversarial Networks (GAN) **[7]**
    - MedGAN **[8]**
    - TableGAN **[9]**
    - CTGAN **[10]**
    - TensorGAN **[11]**

- Existing Trends
  - GANBLR **[12]**

- Related Trends
  - xDeepFM **[13]**
  - TabNet **[14]**

Bayesian Networks



- A BN B is characterized by the structure G (a directed acyclic graph), where each vertex is a variable and a set of parameters
- A BN computes the joint probability distribution as:

$$\mathrm{P}(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} \mathrm{P}(X_i | \Pi(X_i))$$

- Two step learning:
  - Structure Learning
  - Probability Learning
- Special consideration to the class variable (Y):

$$\mathrm{P}(Y, X_1, X_2, \ldots, X_n) = \mathrm{P}(Y) \prod_{i=1}^{n} \mathrm{P}(X_i | Y, \Pi(X_i))$$

$$\mathrm{P}(Y = y, \mathbf{X} = \mathbf{x}) = \theta_y \prod_{i=1}^{n} \theta_{X_i = x_i | Y = y, \Pi(X_i) = \Pi(x_i)}$$

$$\mathrm{P}(y, \mathbf{x}) = \theta_y \prod_{i=1}^{n} \theta_{x_i | y, \Pi(x_i)}$$

How about we maximise this?

Parameter Learning [20]

- Maximize Log-likelihood:

$$\begin{aligned}\mathrm{LL}(\mathcal{G}, \boldsymbol{\theta}) &= \sum_{j=1}^{N} \log \mathrm{P}(y^{(j)}, \mathbf{x}^{(j)}) \\ &= \sum_{j=1}^{N} \left(\log \theta_{y^{(j)}} + \sum_{i=1}^{n} \log \theta_{x_i^{(j)} | y^{(j)}, \Pi(x_i^{(j)})} \right)\end{aligned}$$

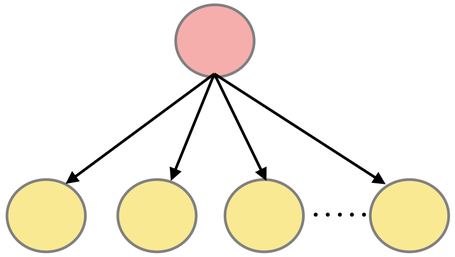
- With constraints:

$$\sum_{y \in \mathcal{Y}} \theta_y = 1 \text{ and } \sum_{x_i \in \mathcal{X}_i} \theta_{x_i | y, \Pi(x_i)} = 1$$

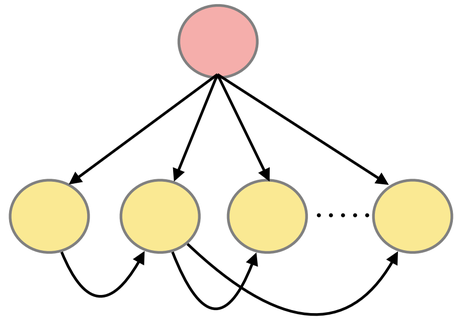
- **Theorem:** *Log-likelihood of Equation 1 is maximized given the constraints in Equation 2, when parameters corresponds to empirical probabilities computed from the sample data, i.e.,* $\theta_y = \mathrm{P}_{\mathrm{data}}(y)$ *and* $\theta_{x_i | y, \Pi(x_i)} = \mathrm{P}_{\mathrm{data}}(x_i | y, \Pi(x_i))$.

- Why maximize Log-likelihood? Why not Conditional Log-Likelihood?

Structure Learning

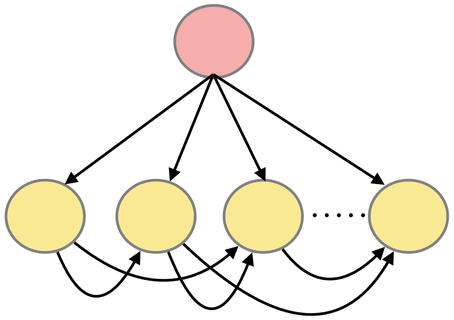


Naive Bayes

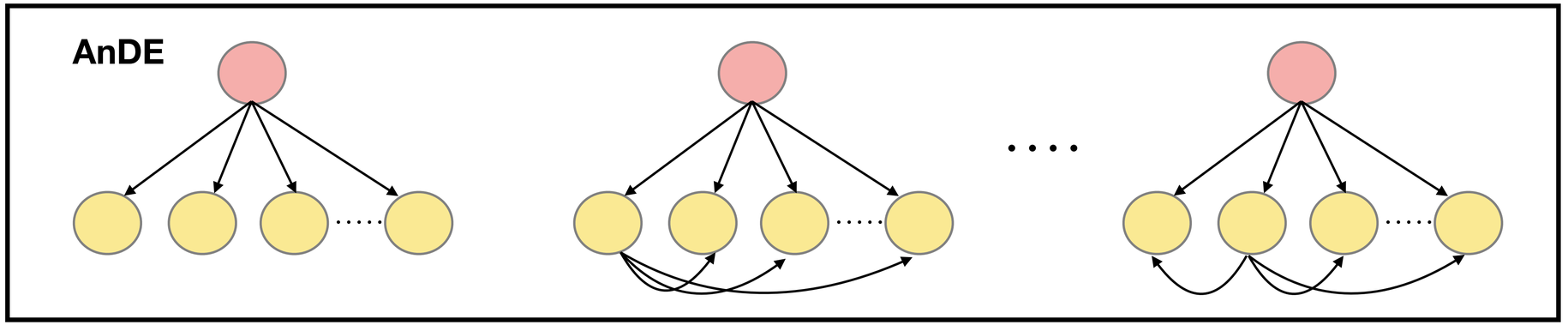


TAN

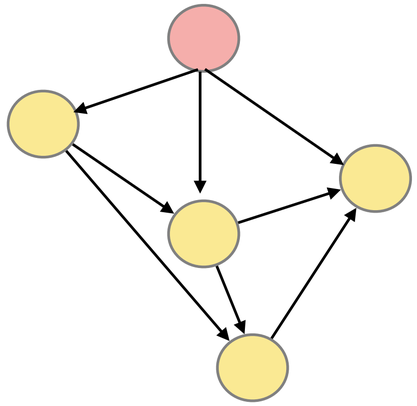
- Restricted:



KDB

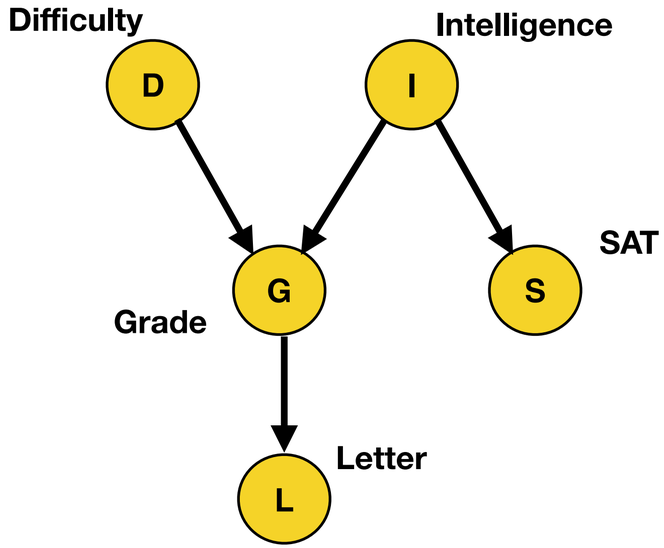


- Un-restricted:



  - Hill Climbing Search (K2)

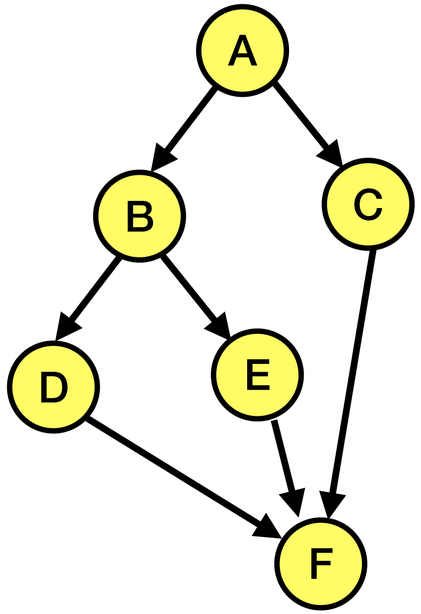
- Domain Experts



More about Bayesian Networks

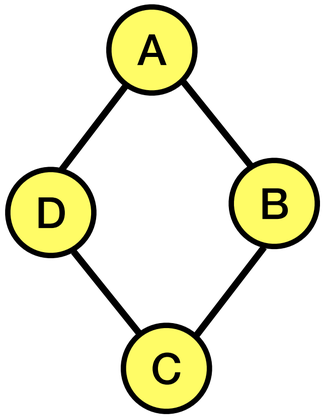
- Pros:
  - Structural guarantees
  - Easy sampling
  - Tractable

- Cons:
  - Not all distributions can be captured by Bayesian Networks



Bayesian Networks provides a descent framework for data generation. They are tractable, intuitive and if the assumptions are held, they can provide excellent results. In any study, they should be used as a baseline to bench-mark the performance of new models.

Markov Networks



- A Markov Network is an undirected graph, and factorized joint distribution as:

$$\mathrm{P}(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{m} \Phi(D_i)$$

- Here D_i is the maximal clique in our network
- Issue in Markov Networks is the tractability, as we must normalise to formulate a probability distribution:

$$\mathrm{P}(X_1, X_2, \ldots, X_n) = \frac{1}{Z} \prod_{i=1}^{m} \Phi(D_i)$$

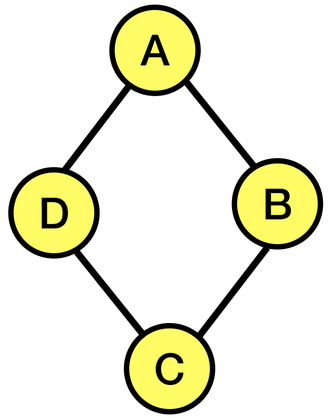
$$\mathrm{P}(A, B, C, D) = \Phi(A, B)\Phi(B, C)\Phi(C, D)\Phi(D, A)$$

$\Phi(A,B)$			$\Phi(B,C)$			$\Phi(C,D)$			$\Phi(D,A)$		
a^0	b^0	30	b^0	c^0	100	c^0	d^0	1	d^0	a^0	1000
a^0	b^1	5	b^0	c^1	1	c^0	d^1	30	d^0	a^1	1
a^1	b^0	1	b^1	c^0	1	c^1	d^0	30	d^1	a^0	1
a^1	b^1	10	b^1	c^1	100	c^1	d^1	1	d^1	a^1	100

$$\mathrm{P}(A, B, C, D) = \frac{1}{Z}\Phi(A, B)\Phi(B, C)\Phi(C, D)\Phi(D, A)$$

More about Markov Networks

- Pros:
  - Structural guarantees
  - Moderate ease of sampling
  - Can represent wide range of distributions



- Cons:
  - Intractable

Markov Networks such as Restricted Boltzmann Machines (RBM) provides a systematic way to approximate the distribution. RBMs can be impractical for image datasets, but still can provide some hope for tabular data especially in the context of Controlled Learning.

Learning in RBMs

$$\mathrm{P}(V,H) = \frac{1}{Z}\prod_i\prod_j \phi_{ij}(v_i,v_j)\prod_i \psi_i(v_i)\prod_j \xi_j(h_j)$$

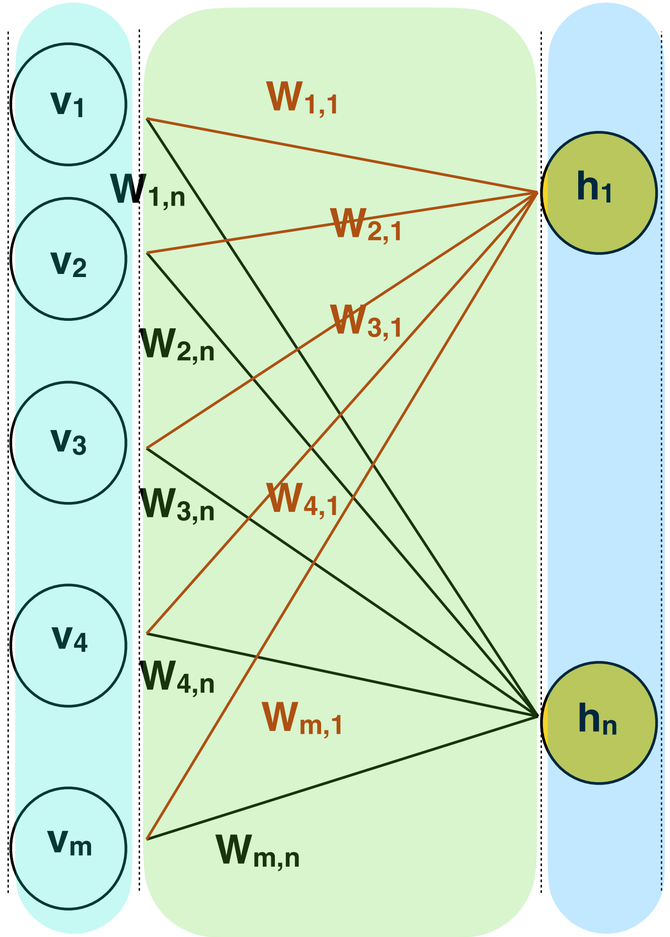
- Introducing parameters in RBM to learn:
  - A parameter form for clique potentials is introduced and the parameters are learned

$$\phi_{ij}(v_i,h_j) = \mathrm{e}^{W_{ij}v_ih_j}$$
$$\psi_i(v_i) = \mathrm{e}^{b_iv_i}$$
$$\xi_j(h_j) = \mathrm{e}^{c_jh_j}$$

$$\mathrm{P}(V,H) = \frac{1}{Z}\prod_i\prod_j \mathrm{e}^{W_{ij}v_ih_j}\prod_i \mathrm{e}^{b_iv_i}\prod_j \mathrm{e}^{c_jh_j}$$

$$\mathrm{P}(V,H) = \frac{1}{Z}\mathrm{e}^{\sum_i\sum_j W_{ij}v_ih_j+\sum_i b_iv_i+\sum_j c_jh_j}$$

$$\mathrm{P}(V,H) = \frac{1}{Z}\mathrm{e}^{-\mathrm{E}(V,H)}$$



$$\mathrm{E}(V,H) = -\sum_i\sum_j W_{ij}v_ih_j - \sum_i b_iv_i - \sum_j c_jh_j$$

RBM — A Stochastic ANN

$$\mathrm{P}(v_l = 1|H) = \sigma(\sum_{j=1}^{n} W_{lj} h_j + b_l)$$

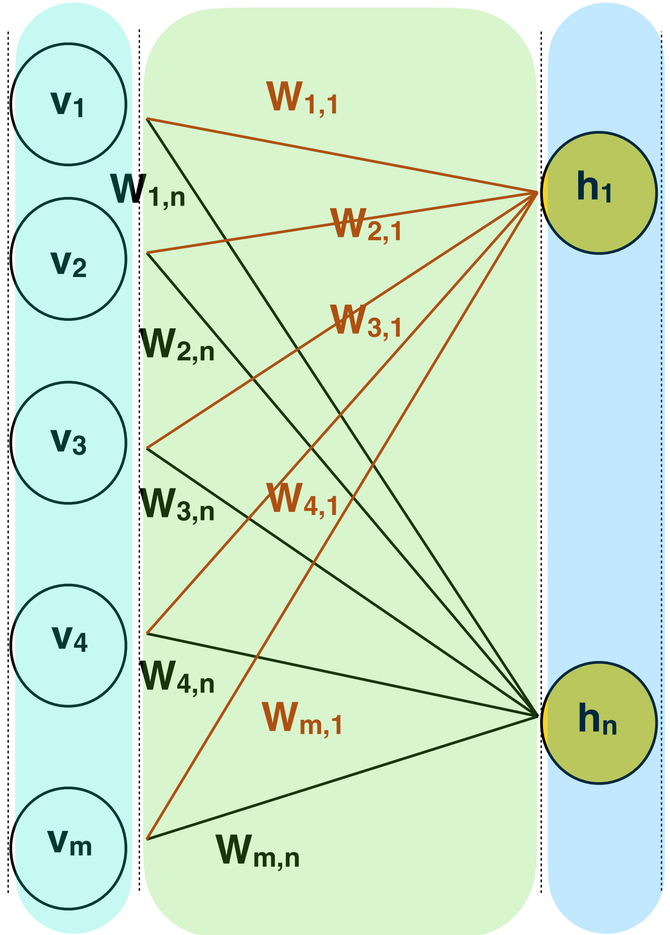
$$\mathrm{P}(h_l = 1|V) = \sigma(\sum_{i=1}^{m} W_{il} v_j + c_l)$$

- What objective function should we use to optimise the weights?

$$\theta^* = \arg\max_{\theta} \sum_{i=1}^{N} \log \mathrm{P}(\mathbf{x}^{(i)}; \theta)$$

$$= \log \frac{1}{Z} \sum_{H} e^{-\mathrm{E}(V,H)}$$

$$= \log \sum_{H} e^{-\mathrm{E}(V,H)} - \log \sum_{V,H} e^{-\mathrm{E}(V,H)}$$



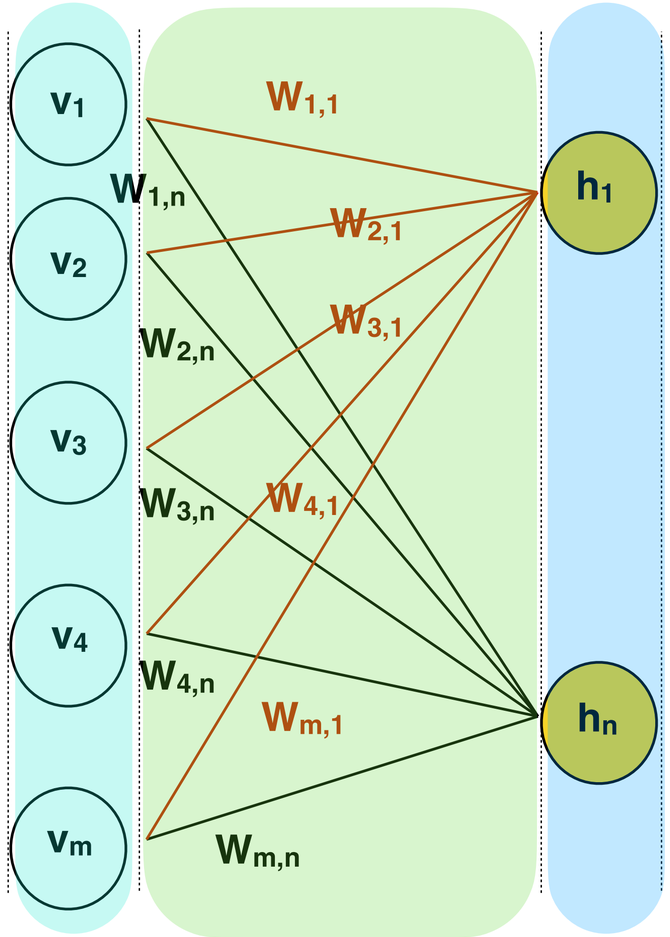
$$\mathrm{E}(V, H) = -\sum_i \sum_j W_{ij} v_i h_j - \sum_i b_i v_i - \sum_j c_j h_j$$

RBM — A Stochastic ANN

$$\mathrm{L}(\theta) = \log \sum_{H} e^{-\mathrm{E}(V,H)} - \log \sum_{V,H} e^{-\mathrm{E}(V,H)}$$

$$\frac{\partial \mathrm{L}(\theta)}{\partial w_{ij}} = \mathrm{E}_{\mathrm{P}(H|V)}\ h_i v_j - \mathrm{E}_{\mathrm{P}(V,H)}\ h_i v_j$$

- Second summation has exponential number of terms and hence is intractable in practice
- What should we do?
  - Sample from another distribution Q, making sure that if you keep drawing from distribution Q, your samples will start to look as if they are drawn from P
  - Gibbs Sampling
    - Contrastive Divergence



$$\mathrm{E}(V,H) = -\sum_i \sum_j W_{ij} v_i h_j - \sum_i b_i v_i - \sum_j c_j h_j$$

RBM — A Stochastic ANN

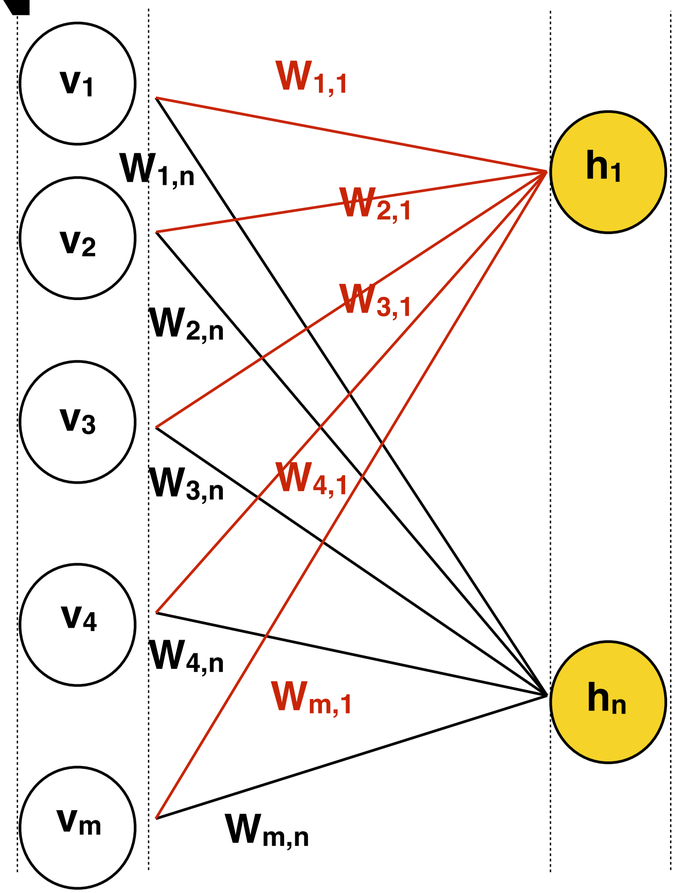
$$\frac{\partial \mathrm{L}(\theta)}{\partial w_{ij}} = \sigma(\sum_{i=1}^{m} W_{ij} v_j + c_i) v_j - \sum_{V} \mathrm{P}(V) \sigma(\sum_{i=1}^{m} W_{ij} v_j + c_i) v_j$$

Gibbs Sampling

$$\approx \frac{1}{K} \sum_{k=1}^{K} \sigma(\sum_{i=1}^{m} W_{ij} v_j^{(k)} + c_i) v_j^{(k)}$$

Contrastive Divergence

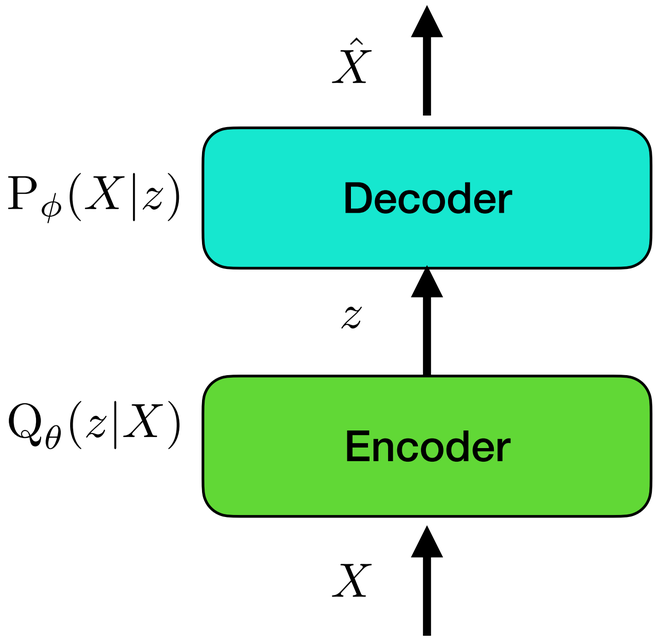
$$\approx \sigma(\sum_{i=1}^{m} W_{ij} v_j^{(k)} + c_i) v_j^{(k)}$$

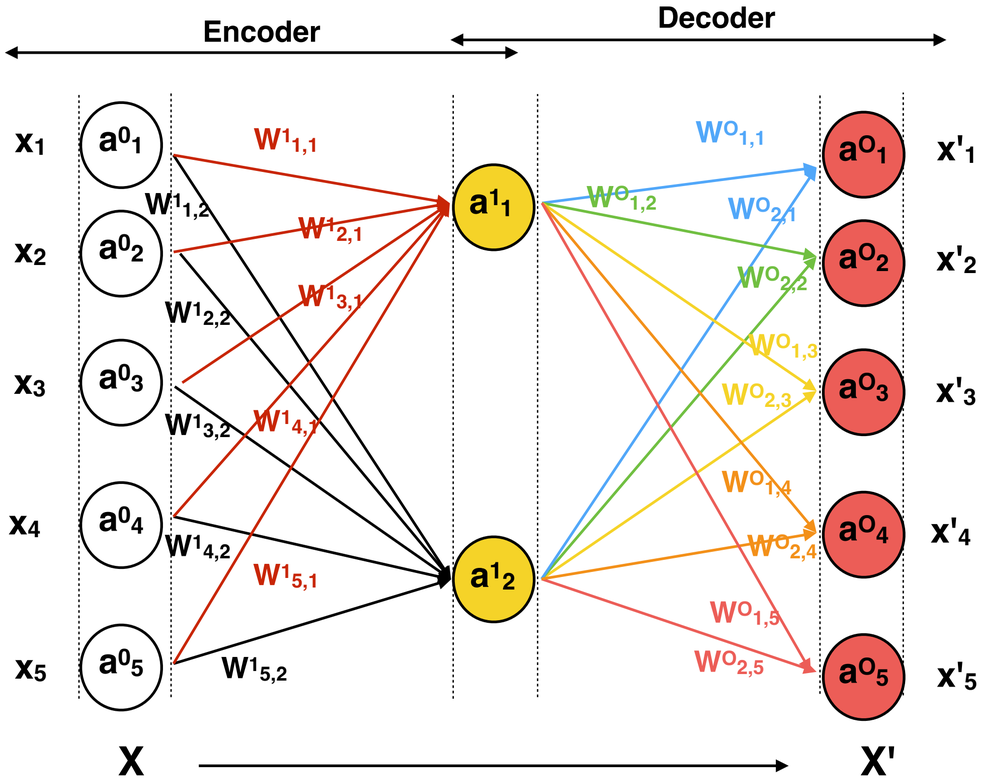


Restricted Boltzmann Machines (RBM) through Gibbs Sampling provides an excellent opportunity for performing Controlled Learning.

Variational Auto-Encoder [5]

- Driven from typical Auto-Encoders
- **Key idea:** make both the encoder and the decoder probabilistic
  - That is, the latent variables z, are drawn from a probability distribution depending on the input, X, and the reconstruction is chosen probabilistically from z





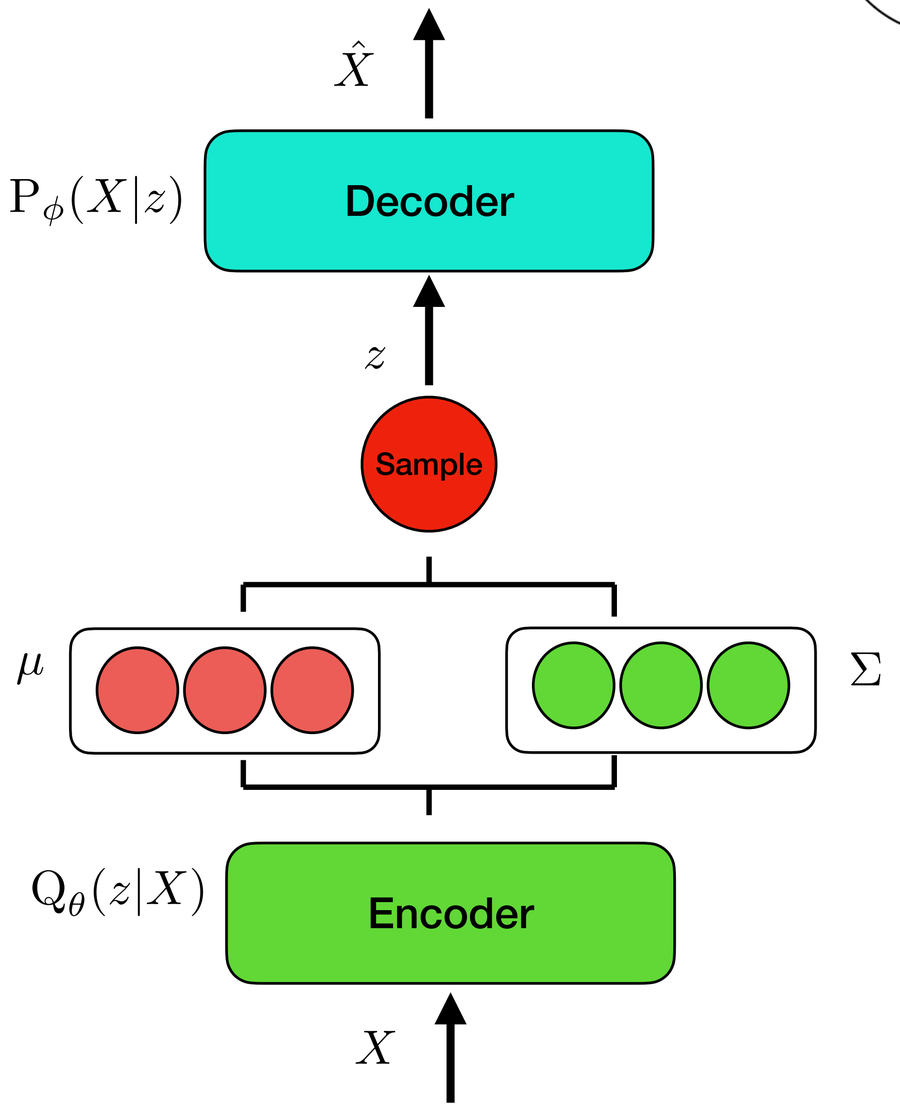
$$\mathbf{h} = g(W\mathbf{x} + b)$$
$$\hat{X} = f(\tilde{W}\mathbf{h} + c)$$

Variational Auto-Encoder

- Goal 1:
  - Learn a distribution over the latent variables
- Goal 2:
  - Learn a distribution over the visible variables

- Use ANN as encoders and decoders
- P(z) are assumed to come from standard normal distribution — zero mean, unit variance

- Encoder:
  - Job of encoder is to learn the parameters of this distribution (mean, variance)
- Decoder
  - P(X|z) is a Gaussian distribution with unit variance and the job of the decoder is to learn the parameters of this distribution

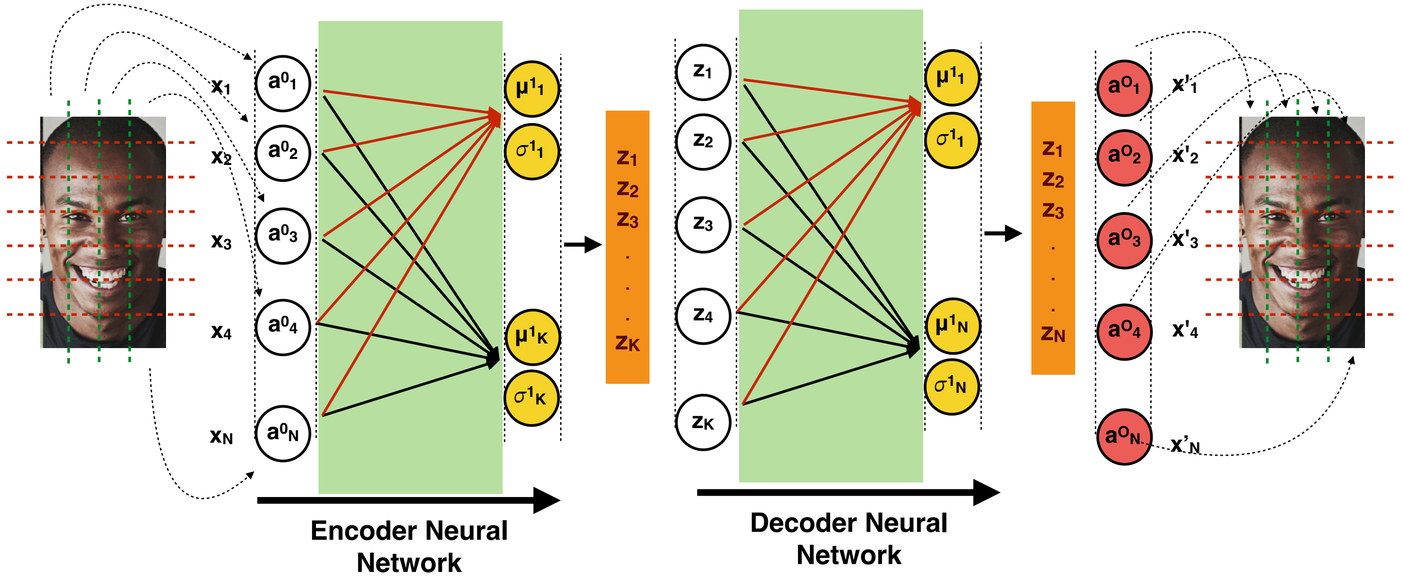


VAE — Objective Function

- VAE maximizes following objective function:

$$\mathrm{P}(\mathbf{x}) = \int \mathrm{P}(z)\mathrm{P}(\mathbf{x}|z)dz$$

$$= -\mathrm{E}_{z\sim \mathrm{Q}_\theta(z|\mathbf{x})}[\log \mathrm{P}_\phi(\mathbf{x}|z)]$$



- Since P(z) is assumed to be Gaussian, and we want Q(z|x) to be as close to P(z) as possible, we modify the loss function slightly as:

$$\mathrm{L}(\theta,\phi) = -\mathrm{E}_{z\sim \mathrm{Q}_\theta(z|\mathbf{x})}[\log \mathrm{P}_\phi(\mathbf{x}|z)] + \mathrm{KL}(\mathrm{Q}_\theta(z|\mathbf{x})||\mathrm{P}(z))$$

Encoder/Decoder

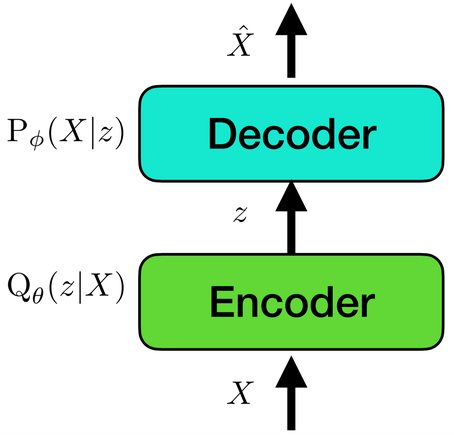
Closed-form Solution

View 1

VAE — Objective Function

$$\mathrm{L}(\theta, \phi) = -\mathrm{E}_{z \sim \mathrm{Q}_\theta(z|\mathbf{x})}[\log \mathrm{P}_\phi(\mathbf{x}|z)] + \mathrm{KL}(\mathrm{Q}_\theta(z|\mathbf{x})||\mathrm{P}(z))$$

View 1



View 2

$$\begin{aligned}
\mathrm{KL}(\mathrm{Q}_\theta(z|\mathbf{x})||\mathrm{P}(z|\mathbf{x})) &= \int \mathrm{Q}_\theta(z|\mathbf{x}) \log \mathrm{Q}_\theta(z|\mathbf{x}) dz - \int \mathrm{Q}_\theta(z|\mathbf{x}) \log \mathrm{P}(z|\mathbf{x}) dz \\
&= \mathrm{E}_{\mathrm{Q}_\theta(z|\mathbf{x})}[\log \mathrm{Q}_\theta(z|\mathbf{x}) - \log \mathrm{P}(z|\mathbf{x})] \\
&= \mathrm{E}_{\mathrm{Q}_\theta(z|\mathbf{x})}[\log \mathrm{Q}_\theta(z|\mathbf{x}) - \log \mathrm{P}(\mathbf{x}|z) - \log \mathrm{P}(z) + \log \mathrm{P}(\mathbf{x})] \\
&= \mathrm{E}_{\mathrm{Q}_\theta(z|\mathbf{x})}[\log \mathrm{Q}_\theta(z|\mathbf{x}) - \log \mathrm{P}(z)] - \mathrm{E}_{\mathrm{Q}_\theta(z|\mathbf{x})}[\log \mathrm{P}(\mathbf{x}|z)] + \log \mathrm{P}(\mathbf{x}) \\
&= \mathrm{KL}(\mathrm{Q}_\theta(z|\mathbf{x})||\mathrm{P}(z)) - \mathrm{E}_{\mathrm{Q}_\theta(z|\mathbf{x})}[\log \mathrm{P}(\mathbf{x}|z)] + \log \mathrm{P}(\mathbf{x})
\end{aligned}$$

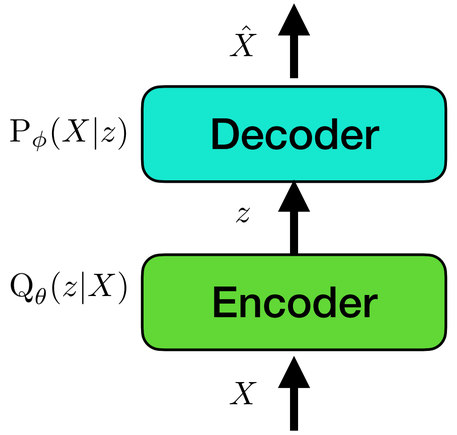
$$\log \mathrm{P}(\mathbf{x}) = \mathrm{E}_{\mathrm{Q}_\theta(z|\mathbf{x})}[\log \mathrm{P}(\mathbf{x}|z)] - \mathrm{KL}[\mathrm{Q}_\theta(z|\mathbf{x})||\mathrm{P}(z)] + \mathrm{KL}(\mathrm{Q}_\theta(z|\mathbf{x})||\mathrm{P}(z|\mathbf{x}))$$

Encoder/Decoder Network

Nice closed form solution

Intractable

VAE — Objective Function



$$\mathrm{L}(\theta,\phi) = -\mathrm{E}_{z\sim \mathrm{Q}_\theta(z|\mathbf{x})}[\log \mathrm{P}_\phi(\mathbf{x}|z)] + \mathrm{KL}(\mathrm{Q}_\theta(z|\mathbf{x})||\mathrm{P}(z))$$

$$\log \mathrm{P}(\mathbf{x}|z) = C - \frac{1}{2}\|\mathbf{x} - \mu_{\mathbf{x}}(z)\|^2$$

$$\log \mathrm{P}(\mathbf{x}|z) = C - \frac{1}{2}\|\mathbf{x} - f_\phi(z)\|^2$$

$$\mathcal{N}(\mu_z(\mathbf{x}), \Sigma_z(\mathbf{x}))$$

$$\mathcal{N}(\mathbf{0}, I)$$

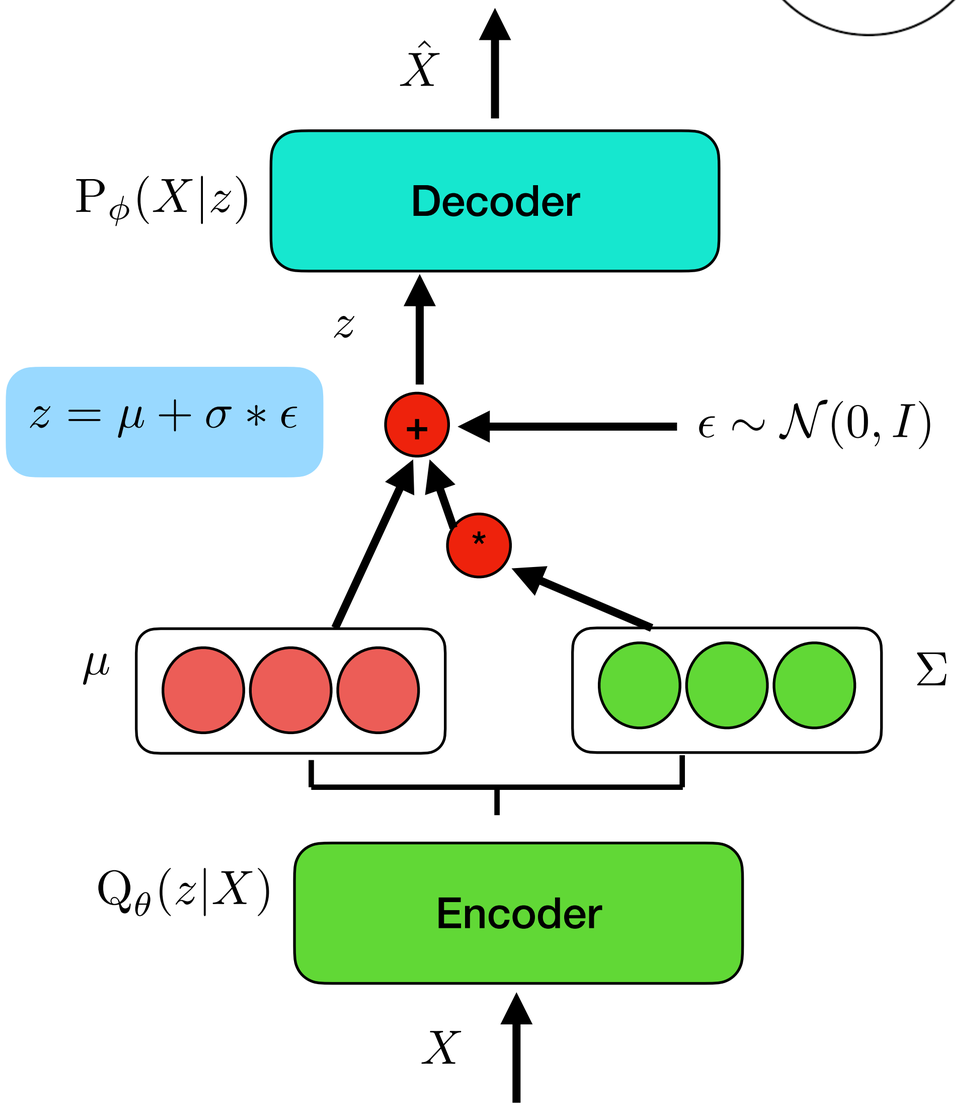
$$\mathrm{KL}(\mathcal{N}(\mu_z(\mathbf{x}), \Sigma_z(\mathbf{x}))||\mathcal{N}(\mathbf{0}, I))$$

$$\frac{1}{2}\left(\mathrm{tr}(\Sigma_z(\mathbf{x})) + (\mu_z(\mathbf{x}))^T[\mu_z(\mathbf{x}) - k - \log\det(\Sigma_z(\mathbf{x}))]\right)$$

$$\mathrm{L}(\theta,\phi) = \left(\mathrm{tr}(\Sigma_z(\mathbf{x})) + (\mu_z(\mathbf{x}))^T[\mu_z(\mathbf{x}) - k - \log\det(\Sigma_z(\mathbf{x}))]\right) + \frac{1}{2}\|\mathbf{x} - f_\phi(z)\|^2$$

Reparametrizatoin Trick

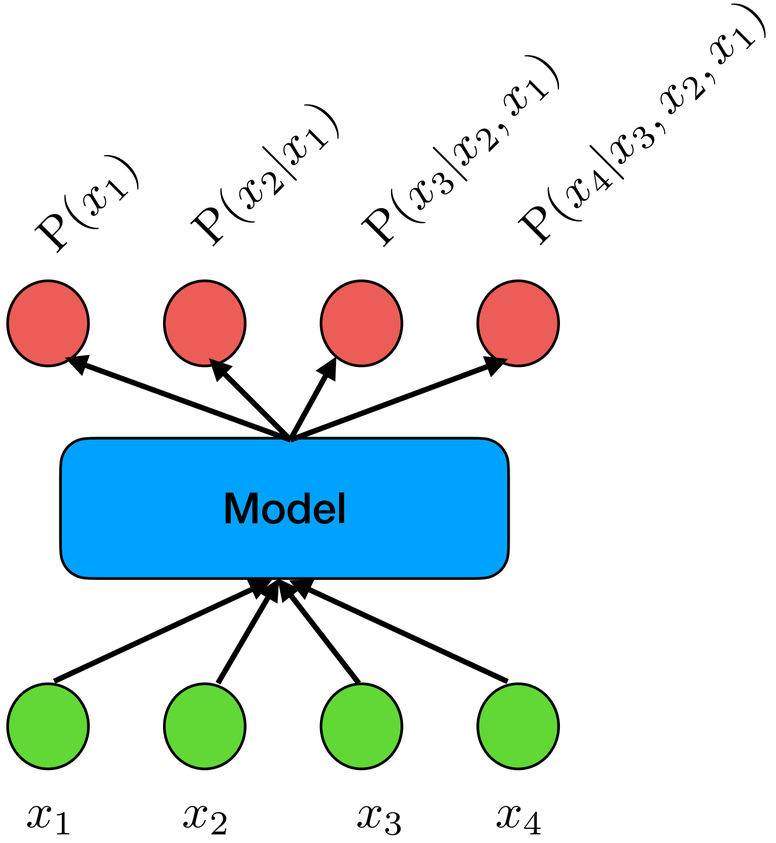
$$\mathrm{L}(\theta, \phi) = -\mathrm{E}_{z \sim \mathrm{Q}_\theta(z|\mathbf{x})}[\log \mathrm{P}_\phi(\mathbf{x}|z)] + \mathrm{KL}(\mathrm{Q}_\theta(z|\mathbf{x})||\mathrm{P}(z))$$



VAE models have sound theoretical foundations, and can be a potential tool for Controlled Learning.

NADE [6]

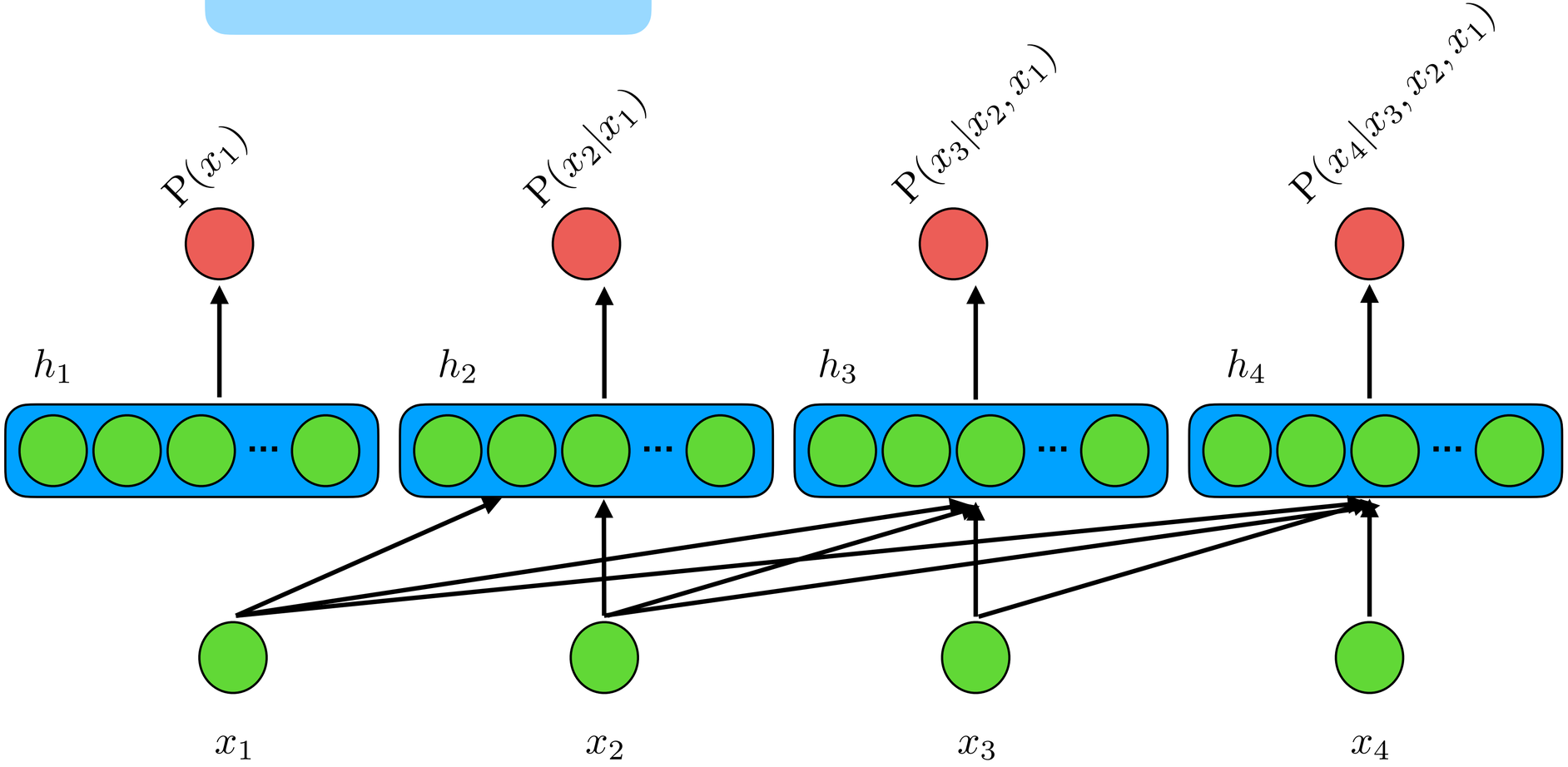
- What we want:
  - We have n number of inputs
  - We have n number of outputs
    - As n increases, the number of parameters of our model should not grow exponentially
- NADE
  - Neural Autoregressive Density Estimator
  - For every output unit, a hidden representation using only the relevant input units is used



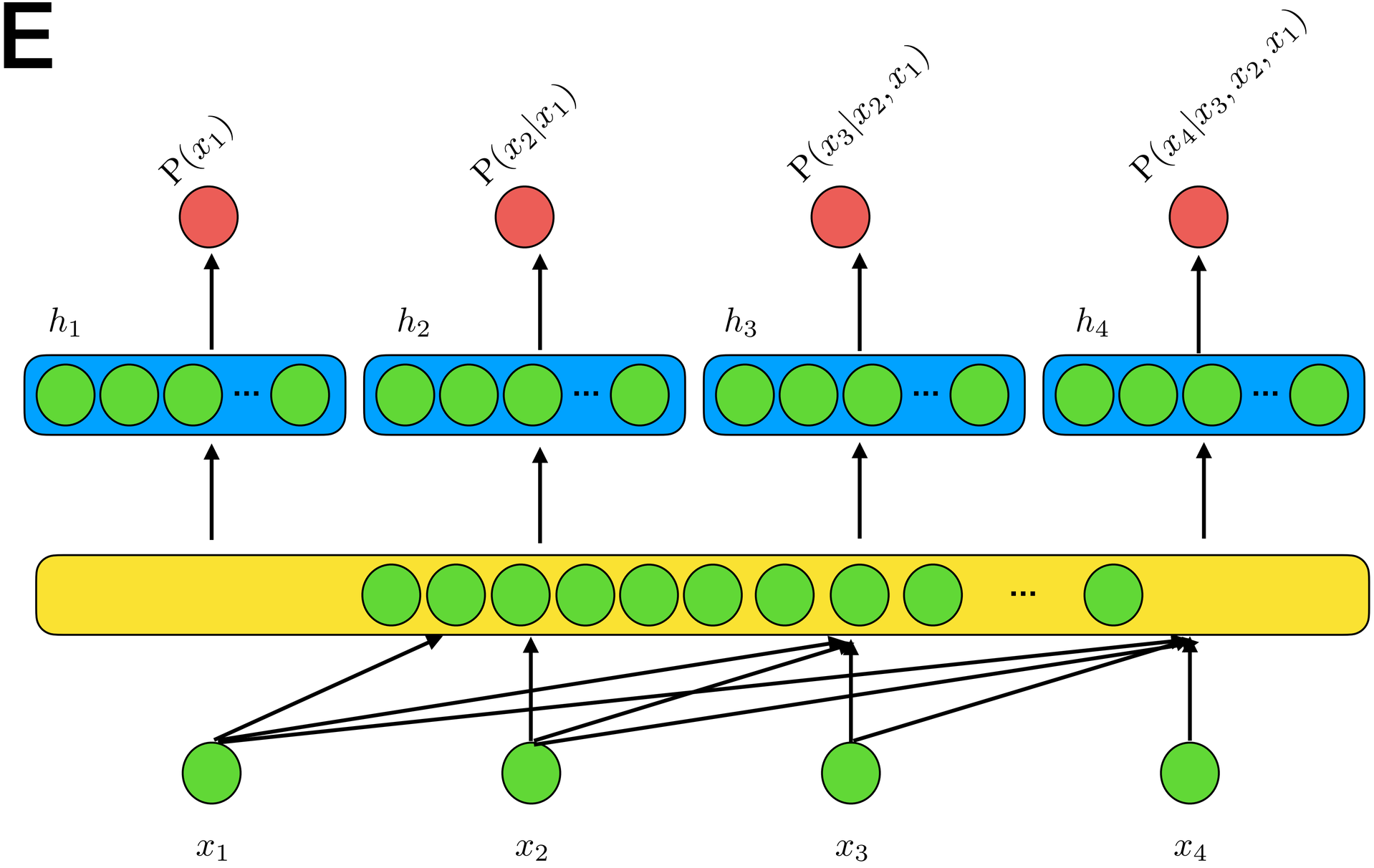
NADE

$$\mathrm{P}(x_k|x_{k-1}, x_{k-2}, \ldots, x_1) = \sigma(V_k h_k + c_k)$$

$$h_k = \sigma(W_{.,<k}\mathbf{x}_{<k} + b)$$



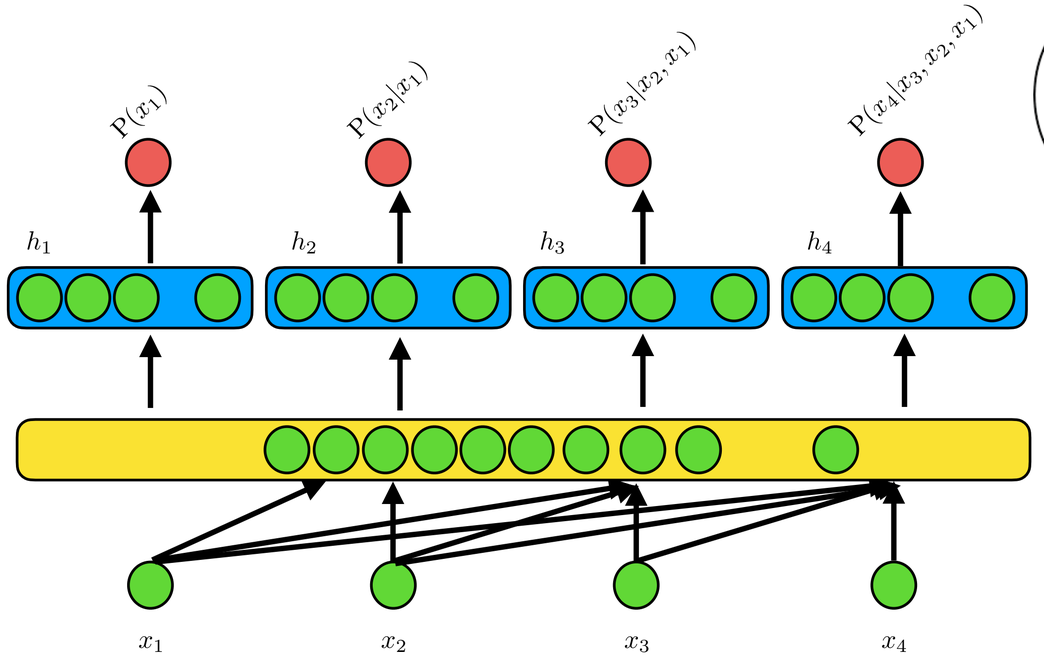
NADE


$\mathrm{P}(x_1)$
$\mathrm{P}(x_2|x_1)$
$\mathrm{P}(x_3|x_2,x_1)$
$\mathrm{P}(x_4|x_3,x_2,x_1)$
h_1
h_2
h_3
h_4
x_1
x_2
x_3
x_4

Training NADE

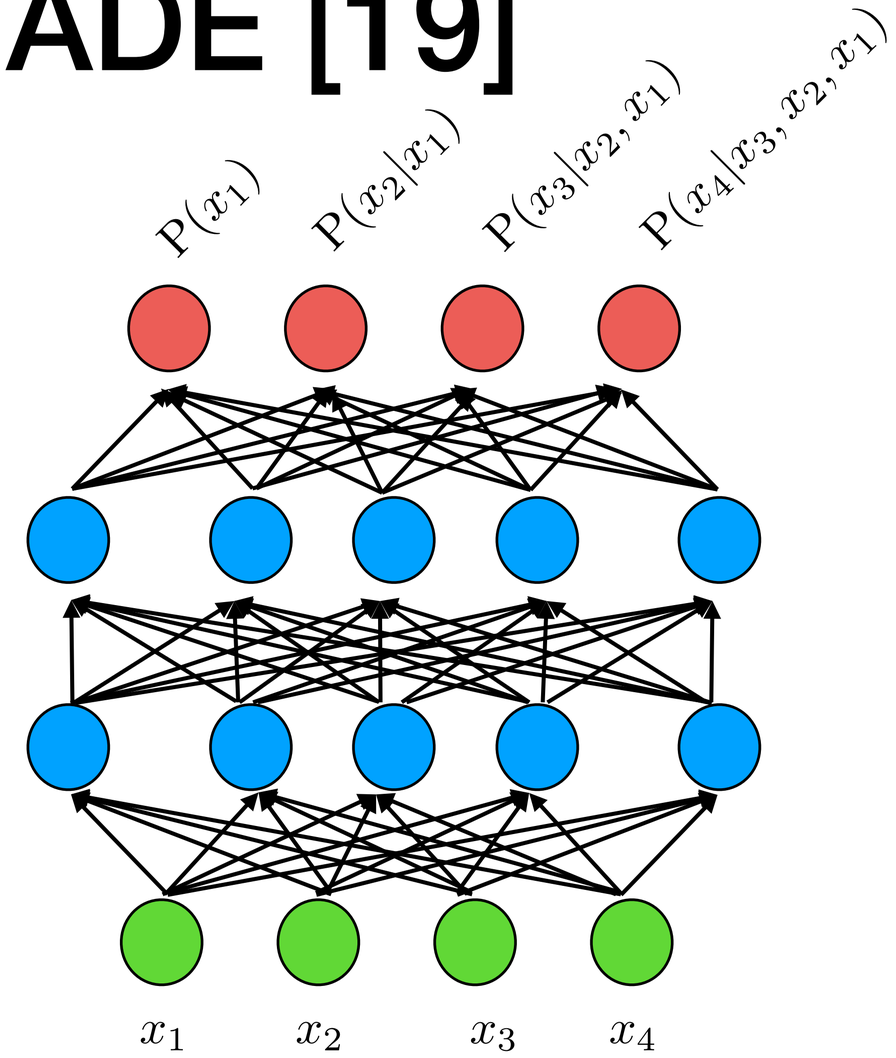
$$\sum_{i=1}^{N} \log \mathrm{P}(\mathbf{x})$$

$$\sum_{i=1}^{N} \sum_{j=1}^{n} \log \mathrm{P}(x_j | \mathbf{x}_{<j})$$

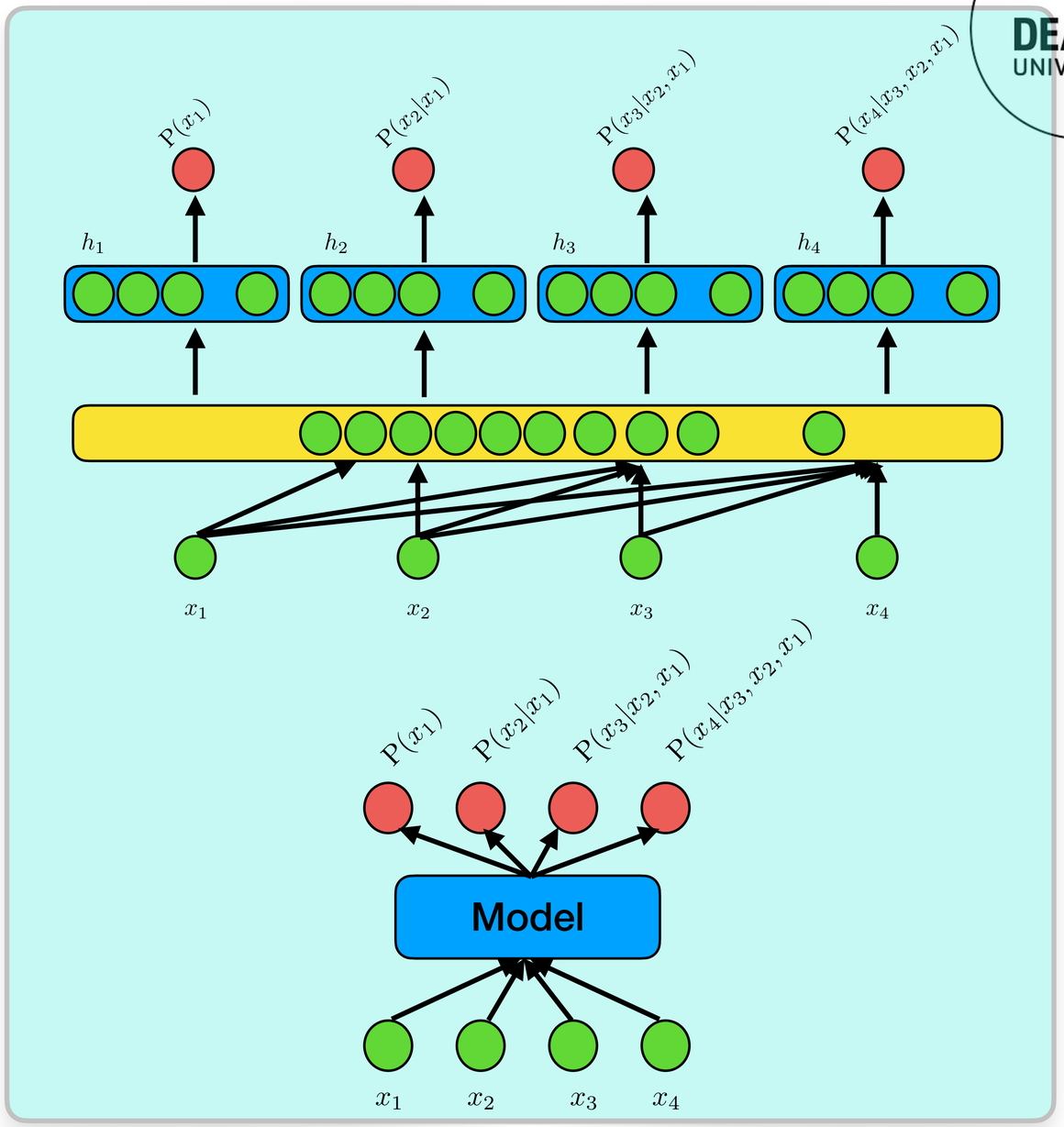


NADE's potential for tabular data generation is not fully explored, however, it provides an excellent model that can be used as the benchmark for other methods.

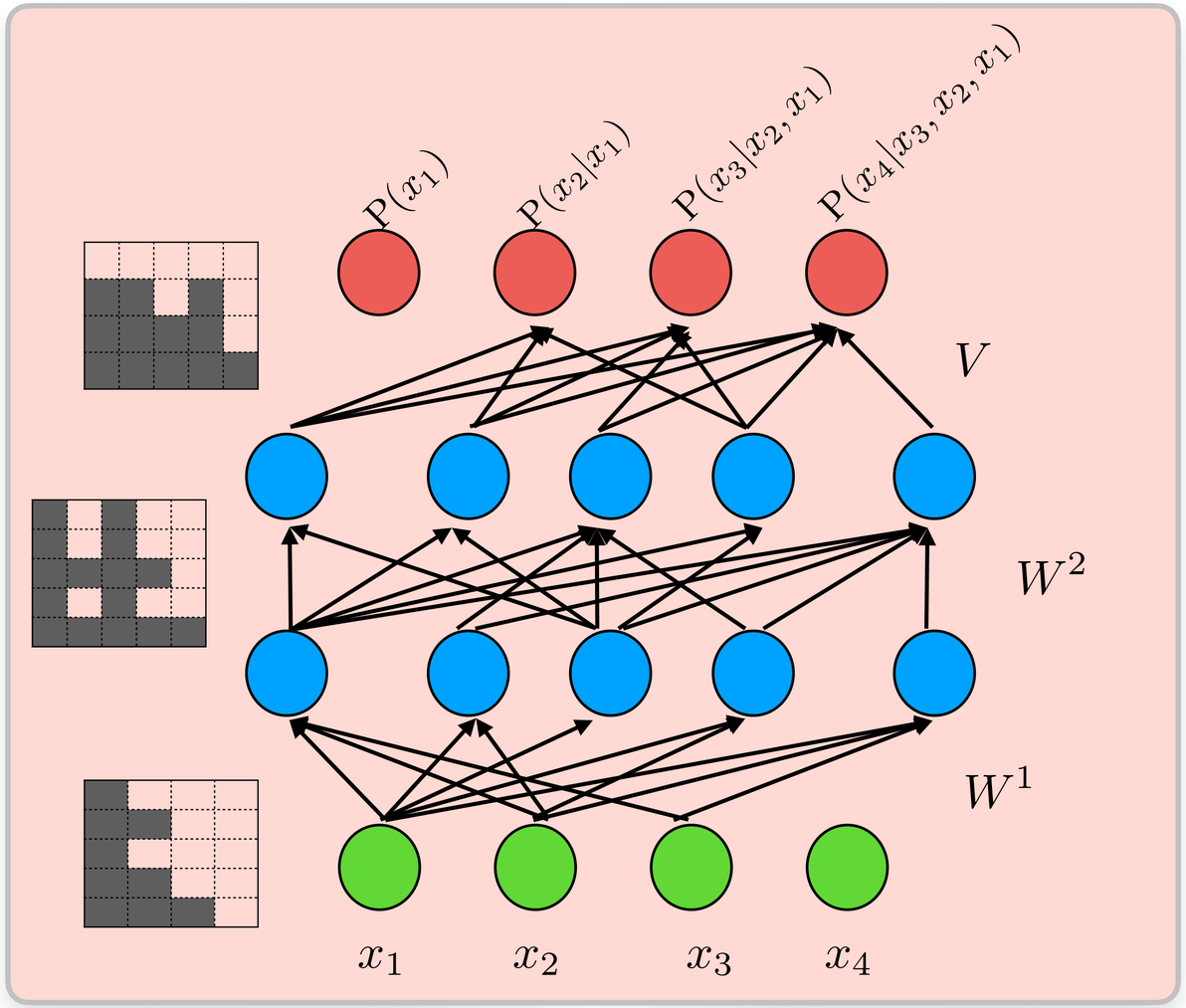
MADE [19]

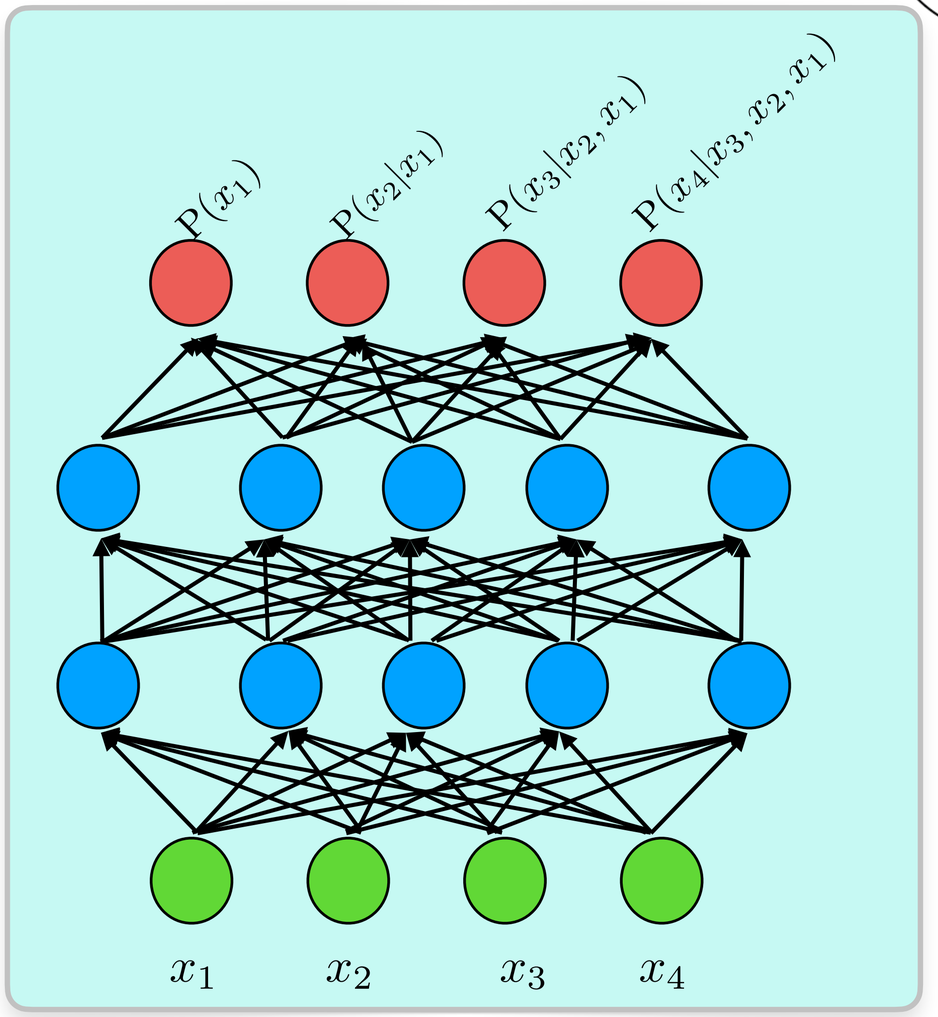


Masked Auto-Encoders Density Estimators

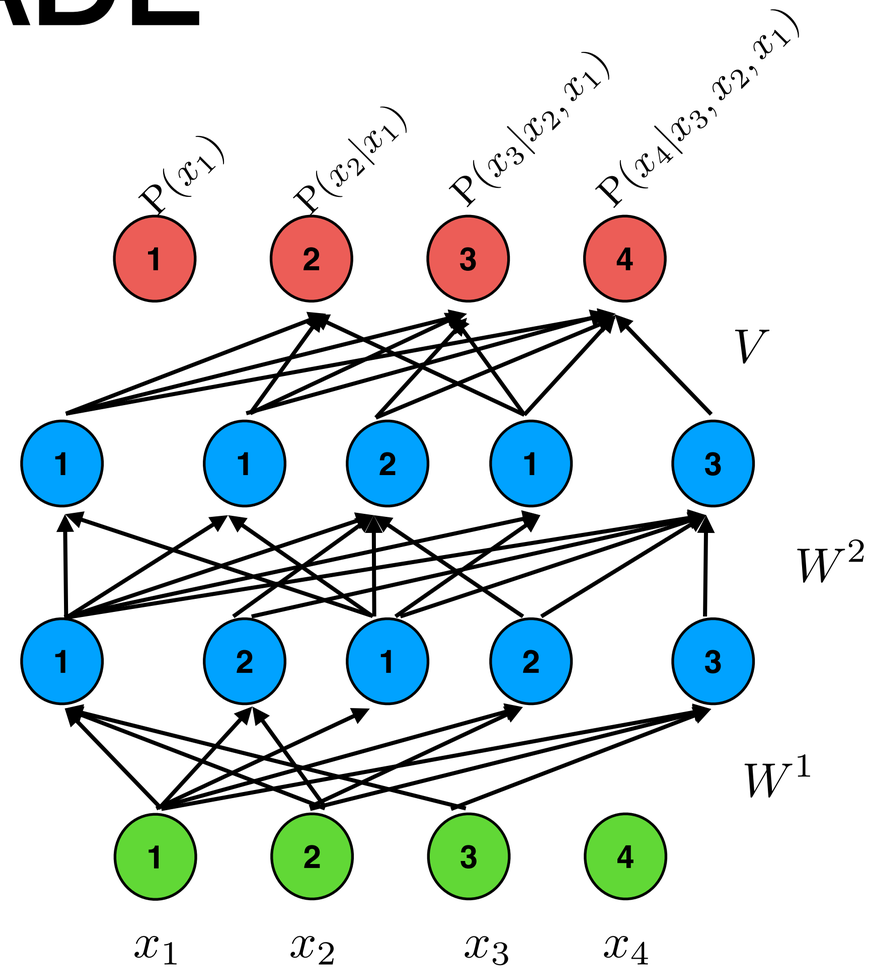


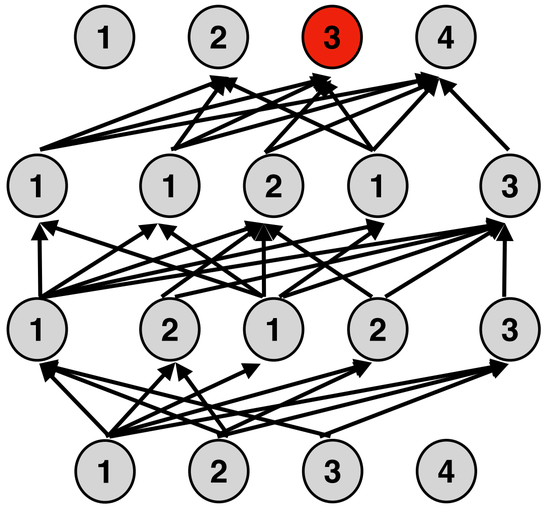
MADE

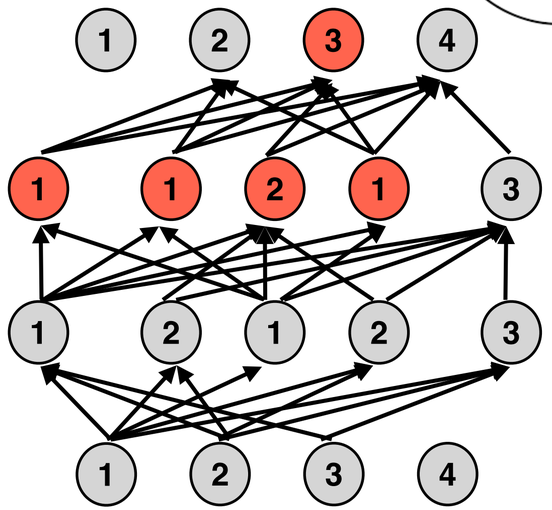

$\mathrm{P}(x_1)$
$\mathrm{P}(x_2|x_1)$
$\mathrm{P}(x_3|x_2,x_1)$
$\mathrm{P}(x_4|x_3,x_2,x_1)$
V
W^2
W^1
x_1
x_2
x_3
x_4

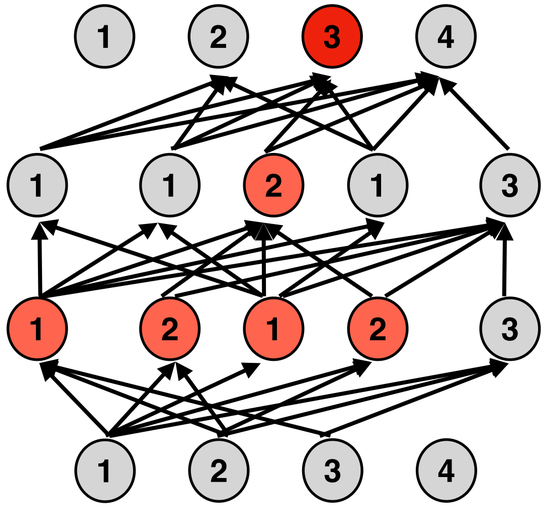

$\mathrm{P}(x_1)$
$\mathrm{P}(x_2|x_1)$
$\mathrm{P}(x_3|x_2,x_1)$
$\mathrm{P}(x_4|x_3,x_2,x_1)$
x_1
x_2
x_3
x_4

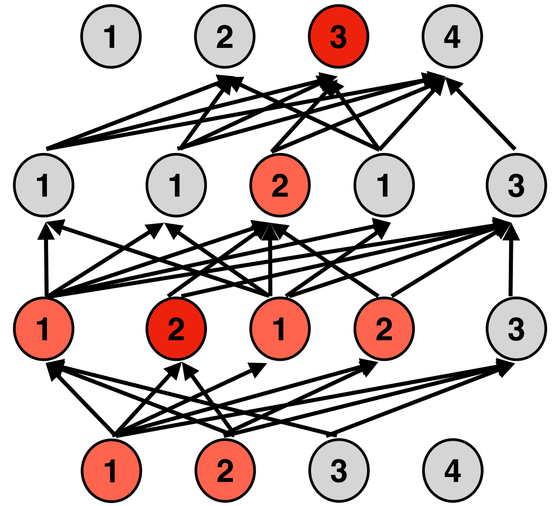
MADE


$\mathrm{P}(x_1)$
$\mathrm{P}(x_2|x_1)$
$\mathrm{P}(x_3|x_2,x_1)$
$\mathrm{P}(x_4|x_3,x_2,x_1)$
1 2 3 4
V
1 1 2 1 3
W^2
1 2 1 2 3
W^1
1 2 3 4
x_1 x_2 x_3 x_4


1 2 3 4
1 1 2 1 3
1 2 1 2 3
1 2 3 4


1 2 3 4
1 1 2 1 3
1 2 1 2 3
1 2 3 4


1 2 3 4
1 1 2 1 3
1 2 1 2 3
1 2 3 4


1 2 3 4
1 1 2 1 3
1 2 1 2 3
1 2 3 4

Training MADE

$$\begin{bmatrix} W^2_{11} & W^2_{12} & W^2_{13} & W^2_{14} & W^2_{15} \\ W^2_{21} & W^2_{22} & W^2_{23} & W^2_{24} & W^2_{25} \\ W^2_{31} & W^2_{32} & W^2_{33} & W^2_{34} & W^2_{35} \\ W^2_{41} & W^2_{42} & W^2_{43} & W^2_{44} & W^2_{45} \\ W^2_{51} & W^2_{52} & W^2_{53} & W^2_{54} & W^2_{55} \end{bmatrix} \odot \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

- Similar to NADE
  - Sum of Cross-Entropies
- Masking can be done quite efficiently
- Similar to NADE, the model is not good for abstraction
  - Excellent for generation

Various other variants of NADE exists, e.g., RNADE, DocNADE, Supervised DocNade, DeepNADE, etc.

Generative Adversarial Networks [7]

$$\min_{\phi} \int p(z) \log(1 - D_\theta(G_\phi(z)))$$

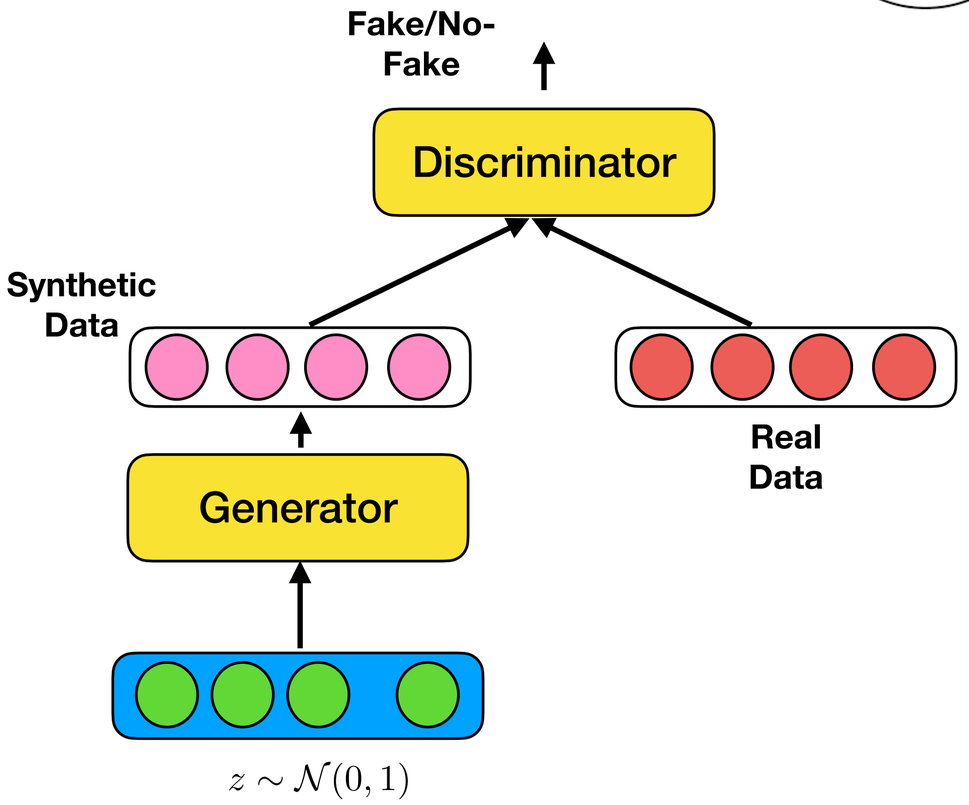
$$\min_{\phi} \mathrm{E}_{z \sim \mathrm{P}(z)}[\log(1 - D_\theta(G_\phi(z)))]$$

Generator's Objective Function

$$\max_{\theta} \quad \mathrm{E}_{x \sim \mathrm{P}_{\mathrm{data}}(.)}[\log D_\theta(\mathbf{x})] + \mathrm{E}_{z \sim \mathrm{P}(z)}[\log(1 - D_\theta(G_\phi(z)))]$$

Discriminator's Objective Function

$$\min_{\phi} \max_{\theta} \quad [\mathrm{E}_{x \sim \mathrm{P}_{\mathrm{data}}(.)} \log D_\theta(\mathbf{x}) + \mathrm{E}_{z \sim \mathrm{P}(z)} \log(1 - D_\theta(G_\phi(z)))]$$



Generative Adversarial Networks

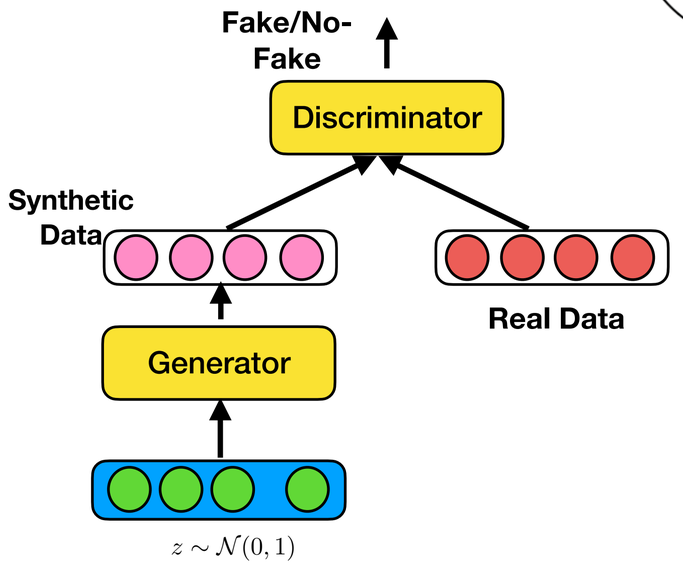
$$\min_{\phi} \max_{\theta} \; [\mathrm{E}_{x \sim \mathrm{P}_{\mathrm{data}}(.)} \log D_\theta(\mathbf{x}) + \mathrm{E}_{z \sim \mathrm{P}(z)} \log(1 - D_\theta(G_\phi(z)))]$$

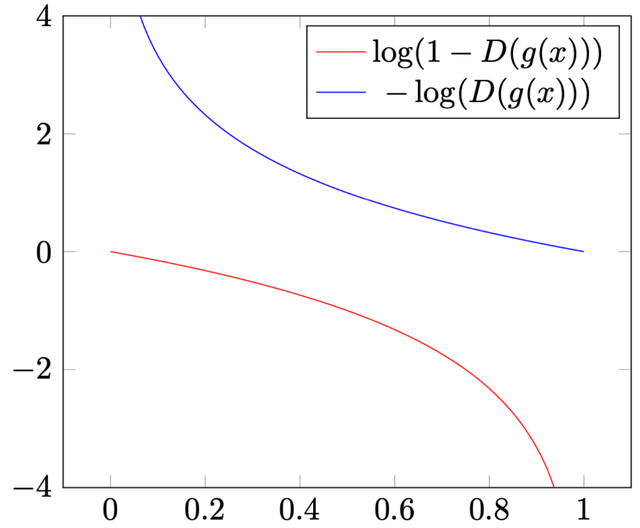
$$\max_{\theta} \; [\mathrm{E}_{x \sim \mathrm{P}_{\mathrm{data}}(.)} \log D_\theta(\mathbf{x}) + \mathrm{E}_{z \sim \mathrm{P}(z)} \log(1 - D_\theta(G_\phi(z)))]$$

Discriminator's Objective Function

$$\min_{\phi} \mathrm{E}_{z \sim \mathrm{P}(z)} [\log(1 - D_\theta(G_\phi(z)))]$$

Generator's Objective Function





Training GANs

- Instead of minimizing the likelihood of the discriminator being correct, maximize the likelihood of the discriminator being wrong, of course, the objective remains the same but the gradient signal becomes better

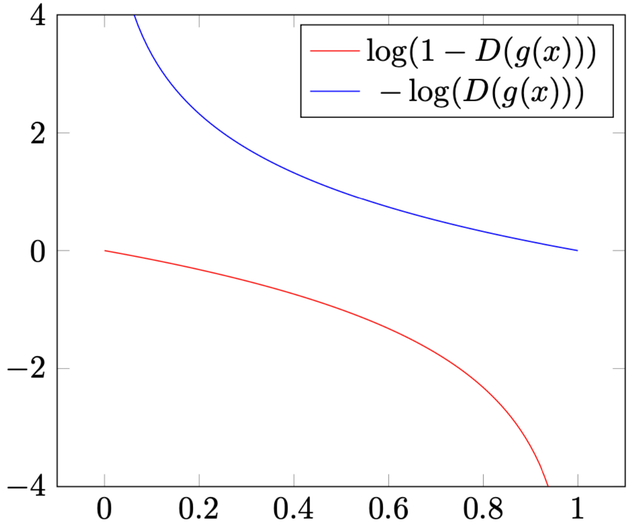
for number of training iterations **do**
  for k steps **do**
    - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

  end for
  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by ascending its stochastic gradient (improved objective):

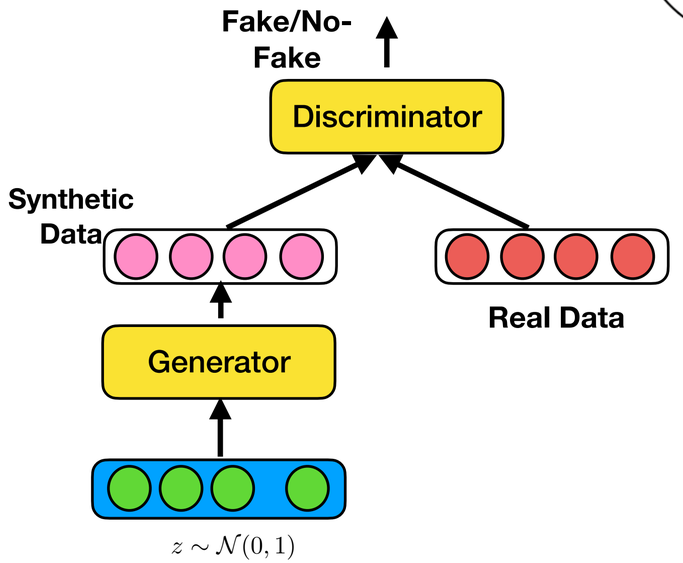
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for



GANs - Summary

- Amazing results on structured datasets
- Hundreds of applications
- Sound theoretical properties:
  - Minimising GAN's objective function (with optimal discriminator) is equivalent to minimising the Jenson-Shannon Divergence (JSD) between P(X) and P(Z)
- Issues - Mode Collapsing
- Variants:
  - Instead of minimising JSD, minimise Wasserstein Distance (WD), leading to **Wasserstein GANs**



GAN's relies heavily on neural network architecture as a generator, and for tabular data will bring the same drawbacks. However, the idea of generator and discriminator working in tandem is great and can be leveraged for tabular datasets

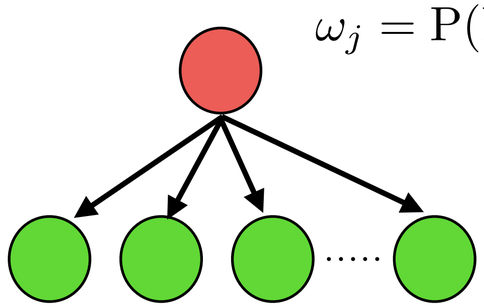
Generative Stochastic Networks [16]

- Inspired from the idea of Denoising Auto-encoders (DAE)
  - Transforms the difficult task of modelling P(X) into a supervised learning problem that may be much easier to solve
- Given X from some unknown P(X), we obtain a corrupted X', by sampling from a known corruption distribution: C(X'|X) - we then use supervised learning methods to train a function - P(X'|X) - to reconstruct, as accurately as possible, any X from the data set given only a noisy version X'

$$\mathrm{P}(X|X') = \frac{1}{Z}C(X'|X)\mathrm{P}(X)$$

TensorGen: Generating HealthCare Records [11]

- Learn a naive Bayes like model
  - Generate m instances following a generative process, however, there is one latent state and that is the class
- Similar to Restricted Boltzmann Machines, however, learning is based on Tensor Factorization
  - Constrained to binary variables
- Results comparable to state-of-the-art methods like medGAN
  - Dataset used is proprietary



Definition 2.1 A Naïve Bayes model (NBM) is a set of $d+1$ variables $(Y, x_1, \ldots, x_d)$ where Y is a hidden (unobservable) discrete variable with a finite number of possible outcomes: $Y \in \{1, \ldots, k\}$. We define $\omega_h := \mathbb{P}(Y = h)$ and $\omega := (\omega_1, \ldots, \omega_k)^\top$. The vector $X = (x_1, \ldots, x_d)$ is observable, its outcomes depend on the value of the hidden variable Y and the random variables $x_1, \ldots, x_d$ are conditionally independent given Y; we define $\mu_{i,j} = \mathbb{E}(x_i | Y = j)$, and $M = (\mu_{i,j})_{i,j} \in \mathbb{R}^{d \times k}$. Also we will denote with μ_i the set of columns of M: $M = [\mu_1|, ..., |\mu_k]$.

TensorGen: Generating HealthCare Records

- When learning the parameters of a **Latent Variable Model (LVM)** with a tensor method, we have to follow two steps:
  - We need to manipulate the observable moments in order to express them in form of a symmetric-low rank tensors, to approximate:

$$M_1 = \sum_{i=1}^{k} \omega_i \mu_i, \; M_2 = \sum_{i=1}^{k} \omega_i \mu_i \otimes \mu_i, \; M_3 = \sum_{i=1}^{k} \omega_i \mu_i \otimes \mu_i \times \mu_i, \text{ where}$$

$$M_1 \in R^d, M_2 \in R^{d \times d}, M_3 \in R^{d \times d \times d}$$

$$\tilde{M_1}^{(N)} = \frac{1}{N} \sum_{i=1}^{N} X^{(i)}$$

$$\tilde{M_2}^{(N)} = \frac{1}{N} \sum_{i=1}^{N} X^{(i)} \otimes X^{(i)}$$

$$\tilde{M_3}^{(N)} = \frac{1}{N} \sum_{i=1}^{N} X^{(i)} \otimes X^{(i)} \otimes X^{(i)}$$

  - Second, we have to use a tensor decomposition algorithm to get (M, ω)

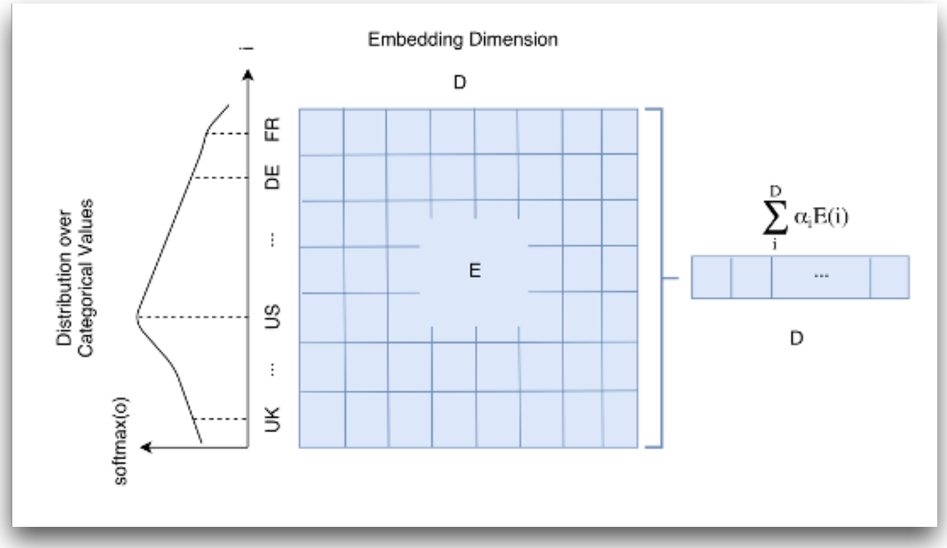
Algorithm 1 SVTD

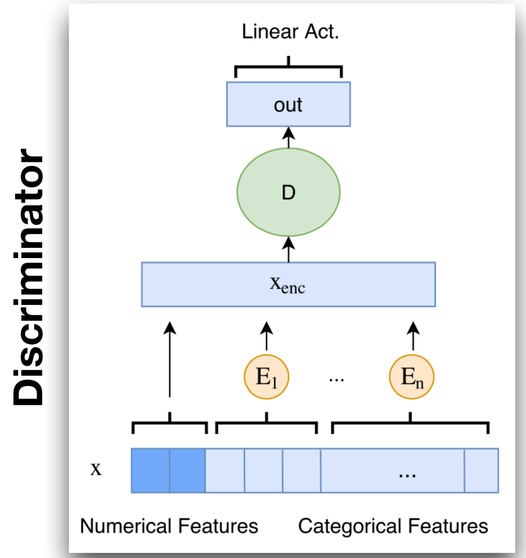
Require: M_1, M_2, M_3, and the number of latent states k

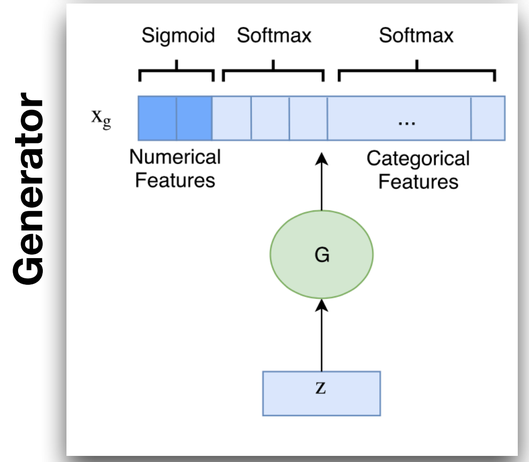
1: Decompose M_2 as $M_2 = U_k S_k U_k^\top$ with a SVD.
2: Select a feature r and compute $M_{3,r}$
3: Compute O as the singular vectors of $H_r := (S_k)^{-\frac{1}{2}} U_k^\top M_{3,r} U_k (S_k)^{-\frac{1}{2}}$
4: **for** $i = 1 \to d$ **do**
5: Compute $H_i := (S_k)^{-\frac{1}{2}} U_k^\top M_{3,i} U_k (S_k)^{-\frac{1}{2}}$
6: Obtain the $i-$th row of M as the diagonal entries of $O^\top H_i O$
7: **end for**
8: Obtain ω solving $M_1 = M\omega$
9: Return (M, ω)

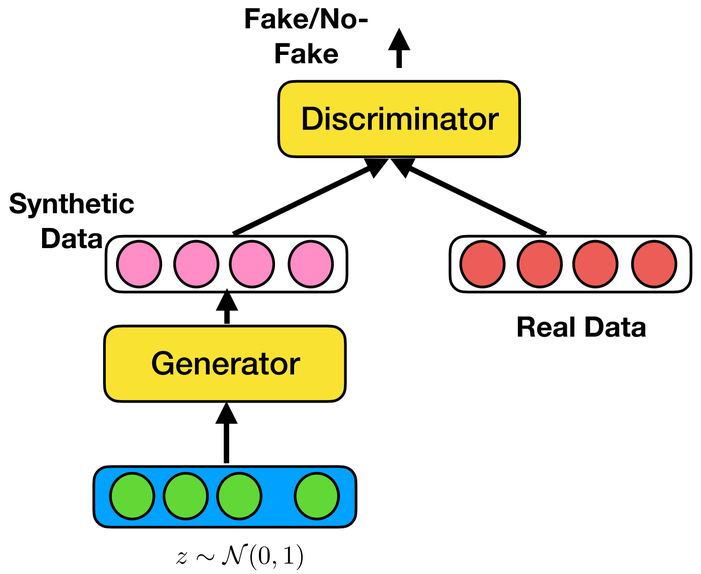
Generating Airline Passenger Records [14]

- Make use of Wasserstein GANs (WGANs)
- For unbiased gradients and better convergence:
  - Make use of Cramer distance
- Special Handling of:
  - Numeric attributes
  - Categorical attributes
  - Missing values





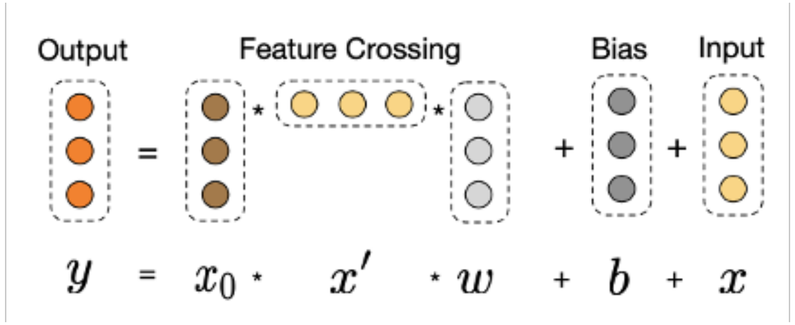




Deep and Cross-Net [17]

- Cross-Net architecture
  - DCN
  - Composed of deep layers
  - And cross-layers:

$$\mathbf{x}_{l+1} = \mathbf{x}_0 \mathbf{x}_l^T \mathbf{w}_l + \mathbf{b}_l + \mathbf{x}_l = f(\mathbf{x}_l, \mathbf{w}_l, \mathbf{b}_l) + \mathbf{x}_l,$$



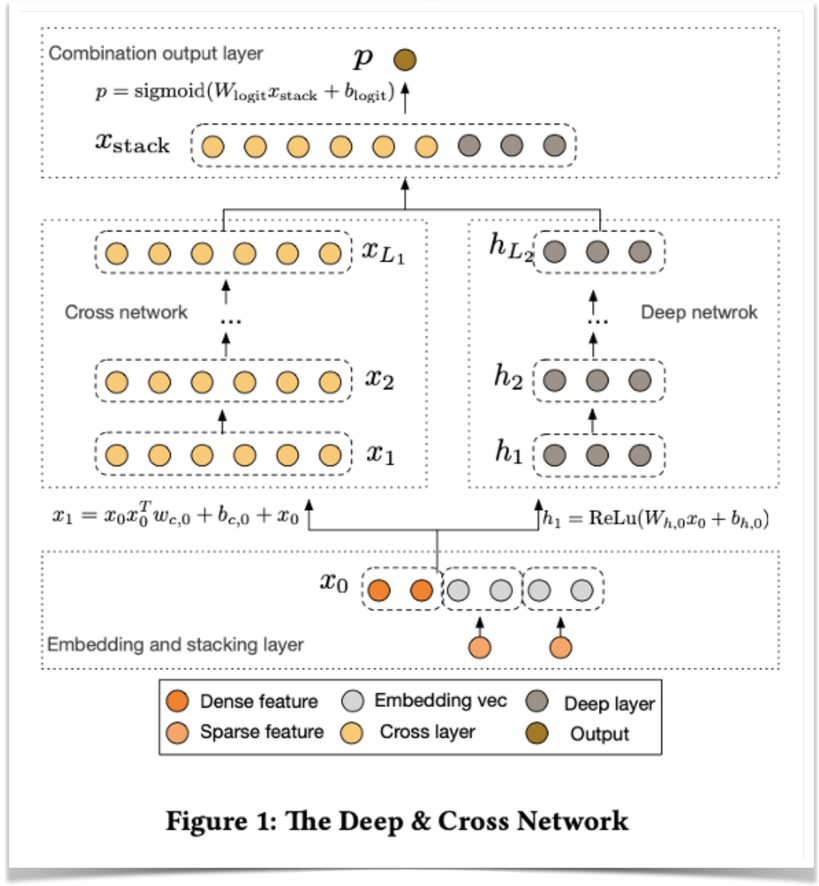
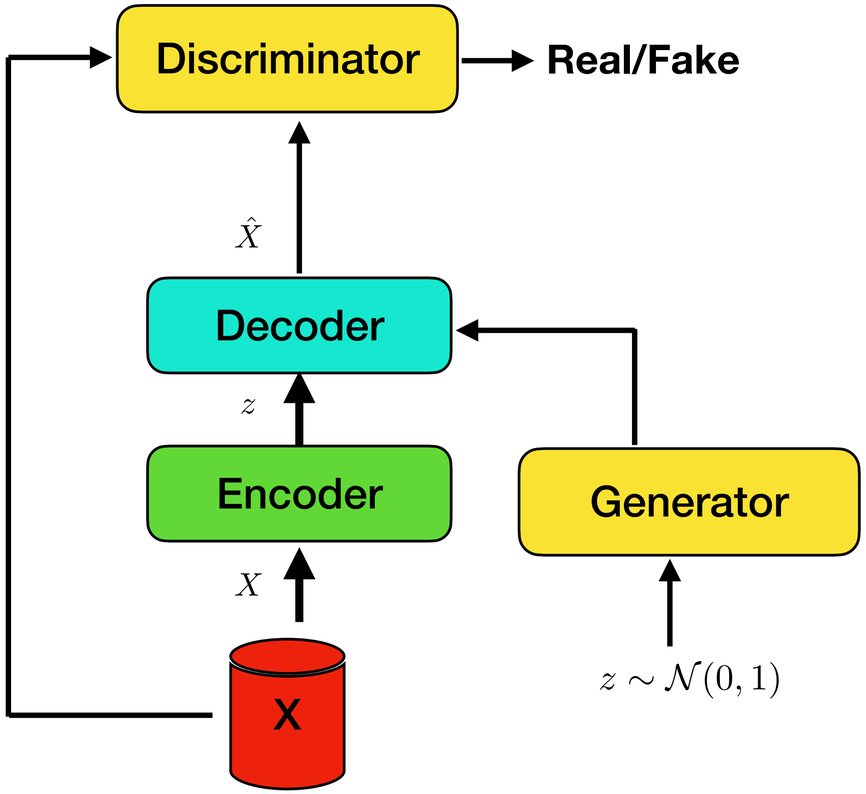
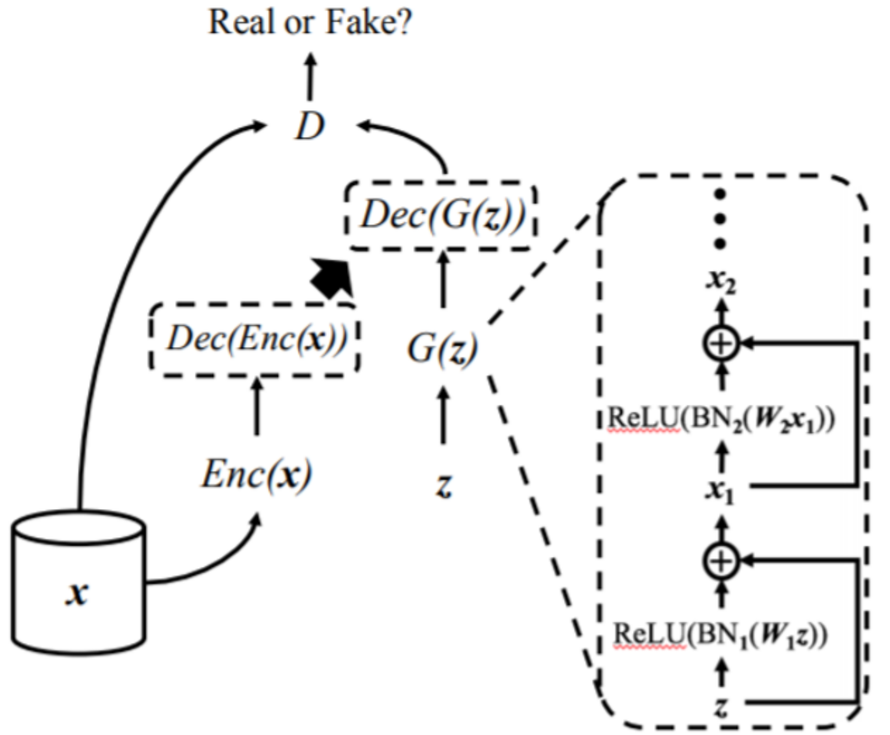


Figure 1: The Deep & Cross Network

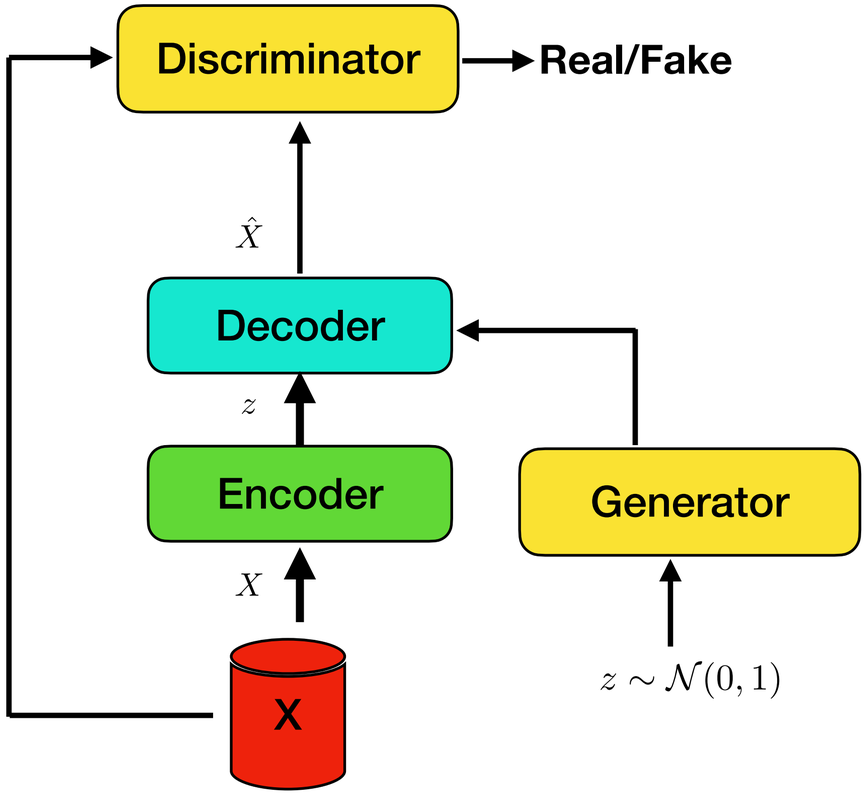
MedGAN [8]


Discriminator
Real/Fake
$\hat{X}$
Decoder
z
Encoder
Generator
X
X
$z \sim \mathcal{N}(0, 1)$


Real or Fake?
D
Dec(G(z))
Dec(Enc(x))
G(z)
Enc(x)
z
x
x_2
$\mathrm{ReLU}(\mathrm{BN}_2(W_2 x_1))$
x_1
$\mathrm{ReLU}(\mathrm{BN}_1(W_1 z))$
z

MedGAN

- Pre-trained Auto-Encoder
  - MSE loss for count variables
  - Cross entropy for binary variables
- GAN's generator aims to generate representation of patient record (that is the output of the encoder), rather than generating the actual record
- Output of decoder is rounded to their nearest integers to ensure that discriminator takes as input only discrete values
- Addressing Mode Collapse
  - Mini-batch Averaging
    - Average of the mini-batch is concatenated on the sample and is provided to the discriminator



MedGAN

$$\theta_d \leftarrow \theta_d + \alpha \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \log D(\mathbf{x}_i) + \log(1 - D(\mathbf{x}_{\mathbf{z}_i}))$$

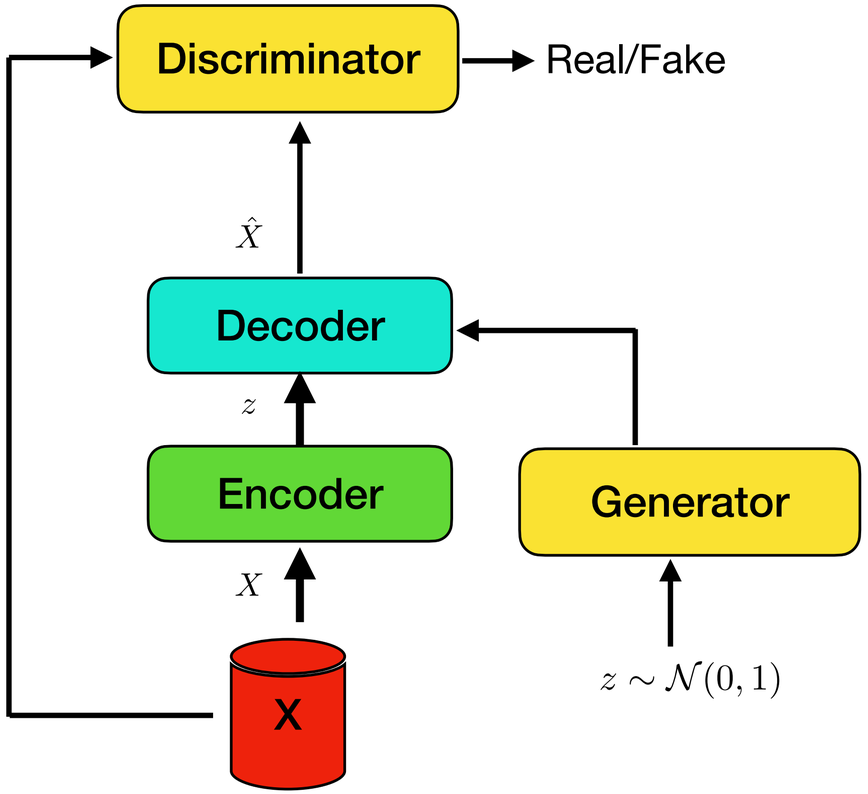
$$\theta_{g,dec} \leftarrow \theta_{g,dec} + \alpha \nabla_{\theta_{g,dec}} \frac{1}{m} \sum_{i=1}^{m} \log D(\mathbf{x}_{\mathbf{z}_i})$$

where $\mathbf{x}_{\mathbf{z}_i} = Dec(G(\mathbf{z}_i))$

$$\theta_d \leftarrow \theta_d + \alpha \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \log D(\mathbf{x}_i, \bar{\mathbf{x}}) + \log(1 - D(\mathbf{x}_{\mathbf{z}_i}, \bar{\mathbf{x}}_{\mathbf{z}}))$$

$$\theta_{g,dec} \leftarrow \theta_{g,dec} + \alpha \nabla_{\theta_{g,dec}} \frac{1}{m} \sum_{i=1}^{m} \log D(\mathbf{x}_{\mathbf{z}_i}, \bar{\mathbf{x}}_{\mathbf{z}})$$

where $\bar{\mathbf{x}} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i$, $\mathbf{x}_{\mathbf{z}_i} = Dec(G(\mathbf{z}_i))$, $\bar{\mathbf{x}}_{\mathbf{z}} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_{\mathbf{z}_i}$

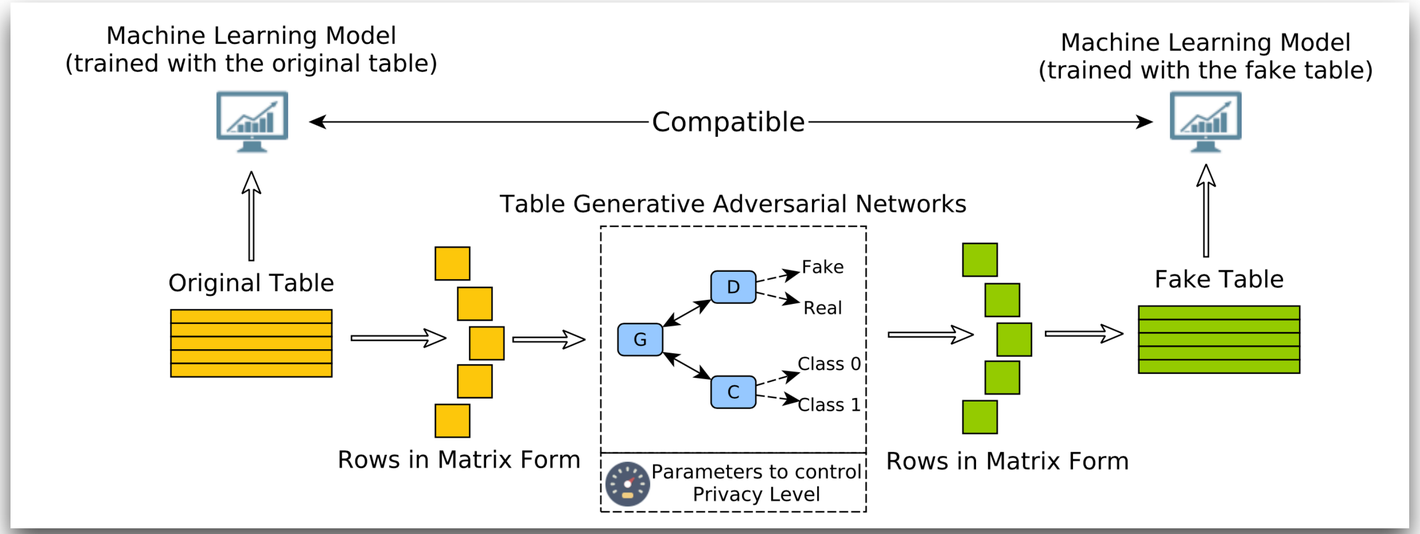


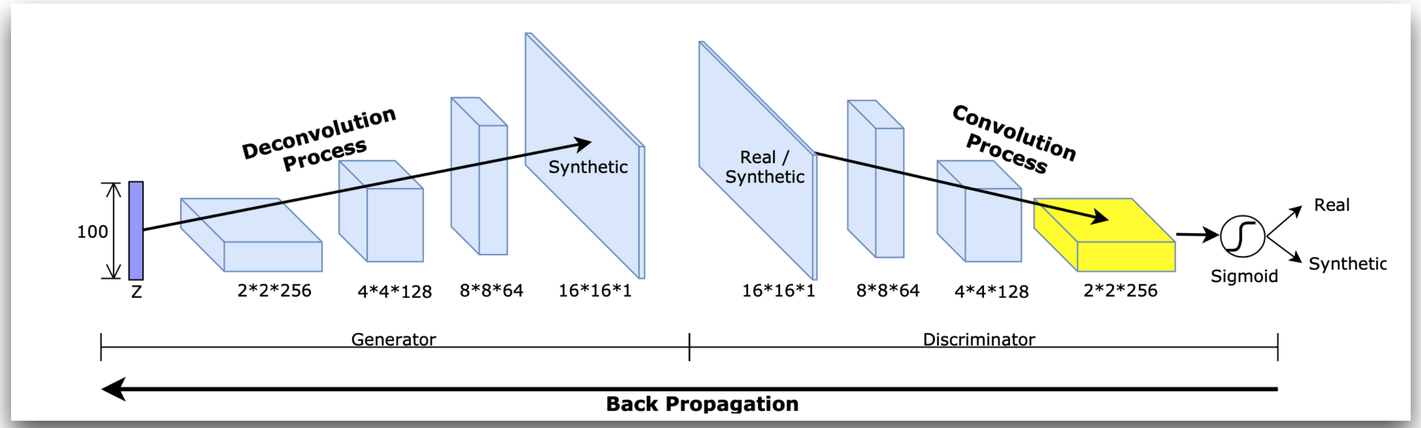
- Addressing Mode Collapse
  - Mini-batch Averaging
    - Average of the mini-batch is concatenated on the sample and is provided to the discriminator

TableGAN [9]

- Address privacy and security concerns:
  - Re-identification attack
  - Attribute disclosure
  - Membership attack

- Adopts DCGAN **[22]** architecture and have Generator and Discriminator, **but has an additional component — Classifier (C):**
  - C predicts synthetic record labels
  - Adding C helps maintain the consistency of values in the generated records
    - E.g., record with ‘Gender = Male’ and ‘Disease = Uterine Cancer’, must be prevented





TableGAN

- Three Losses
  - Typical GAN Loss
  - Information Loss
  - Classification Loss

f stands for feature extracted from last layer

Threshold for controlling the level of privacy

1

$$\min_{\phi}\max_{\theta}\ [\mathrm{E}_{x\sim \mathrm{P}_{\mathrm{data}}(.)}\log D_\theta(\mathbf{x}) + \mathrm{E}_{z\sim \mathrm{P}(z)}\log(1-D_\theta(G_\phi(z)))]$$

$$\mathcal{L}_{\mathrm{mean}} = \|\mathrm{E}[\mathbf{f}_x]_{x\sim \mathrm{P}_{\mathrm{data}}} - \mathrm{E}[\mathbf{f}_{G(z)}]_{z\sim \mathrm{P}_z}\|_2$$

$$\mathcal{L}_{\mathrm{sd}} = \|\mathrm{SD}[\mathbf{f}_x]_{x\sim \mathrm{P}_{\mathrm{data}}} - \mathrm{SD}[\mathbf{f}_{G(z)}]_{z\sim \mathrm{P}_z}\|_2$$

Controlled Learning

2

$$\mathcal{L}^G_{\mathrm{info}} = \max(0, \mathcal{L}_{\mathrm{mean}} - \delta_{\mathrm{mean}}) + \max(0, \mathcal{L}_{\mathrm{SD}} - \delta_{\mathrm{SD}})$$

3

$$\mathcal{L}^G_{\mathrm{class}} = \mathrm{E}[l(G(z)) - C(\mathrm{remove}(G(z)))]_{z\sim \mathrm{P}(z)}$$

Function l returns the label

TableGAN

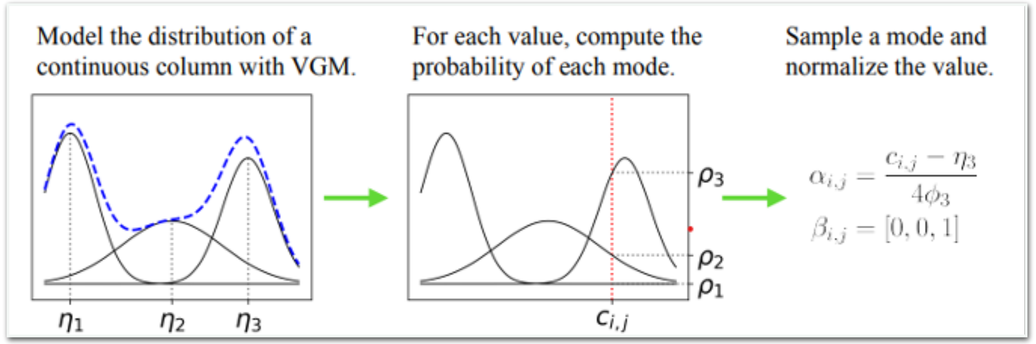
- Three Losses
  - Typical GAN Loss
  - Information Loss
  - Classification Loss

Input: real records: $\{x_1, x_2, \cdots\} \sim p(x)$
Output: a generative model G

$G \leftarrow$ a generative neural network
$D \leftarrow$ a discriminator neural network
$C \leftarrow$ a classifier neural network
/* Initializing to zero vectors */
$\mathbf{f}^X_{mean} \leftarrow \mathbf{0}; \mathbf{f}^X_{sd} \leftarrow \mathbf{0}; \mathbf{f}^Z_{mean} \leftarrow \mathbf{0}; \mathbf{f}^Z_{sd} \leftarrow \mathbf{0}$
while *until convergence of loss values* **do**
    Create a mini-batch of real records $X_{mini} = \{x_1, \cdots, x_n\}$
    Create a mini-batch of latent vector inputs for G $Z_{mini} = \{z_1, \cdots, z_n\}$
    Perform the SGD update of the discriminator D with $\mathcal{L}^D_{orig}$
    Perform the SGD update of the classifier C with $\mathcal{L}^C_{class}$
    /* Moving average update of the mean and standard deviation of features */
    $\mathbf{f}^X_{mean} = w \times \mathbf{f}^X_{mean} + (1-w) \times \mathbb{E}[\mathbf{f}_x]_{x \in X_{mini}}$
    $\mathbf{f}^X_{sd} = w \times \mathbf{f}^X_{sd} + (1-w) \times \mathbb{SD}[\mathbf{f}_x]_{x \in X_{mini}}$
    $\mathbf{f}^Z_{mean} = w \times \mathbf{f}^Z_{mean} + (1-w) \times \mathbb{E}[\mathbf{f}_{G(z)}]_{z \in Z_{mini}}$
    $\mathbf{f}^Z_{sd} = w \times \mathbf{f}^Z_{sd} + (1-w) \times \mathbb{SD}[\mathbf{f}_{G(z)}]_{z \in Z_{mini}}$
    Perform the SGD update of the generator G with $\mathcal{L}^G_{orig} + \mathcal{L}^G_{info} + \mathcal{L}^G_{class}$
end
return G

CTGAN [10]

- Mode Specific Normalisation
  - Continuous Attributes:
    - Use Variational GMM to estimate the number of modes (m), and fit a Gaussian mixture



- Discrete Attributes
  - One-hot Representation

- Final Representation

$$\mathbf{r}_j = \alpha_{1,j} \otimes \beta_{1,j} \otimes \ldots \otimes \alpha_{N_c,j} \otimes \beta_{N_c,j} \otimes \mathbf{d}_{1,j} \otimes \ldots \otimes \mathbf{d}_{N_d,j}$$

- The method of training a generator does not account for the imbalance in the categorical columns. If the training data are randomly sampled during training, the rows that fall into the minor category will not be sufficiently represented. If the training data is re-sampled, the generator learns the resampled distribution which is different from the real distribution

CTGAN

- Training-by-sampling
  - Conditional Vector
  - Create N_d number of masks: m_1, m_2, …
  - Randomly select a column with equal probability
    - Construct PMF across the range of the values
    - log(frequency) of each value
    - Select the value based on PMF, and set the mask accordingly
- Set the **condition vector**

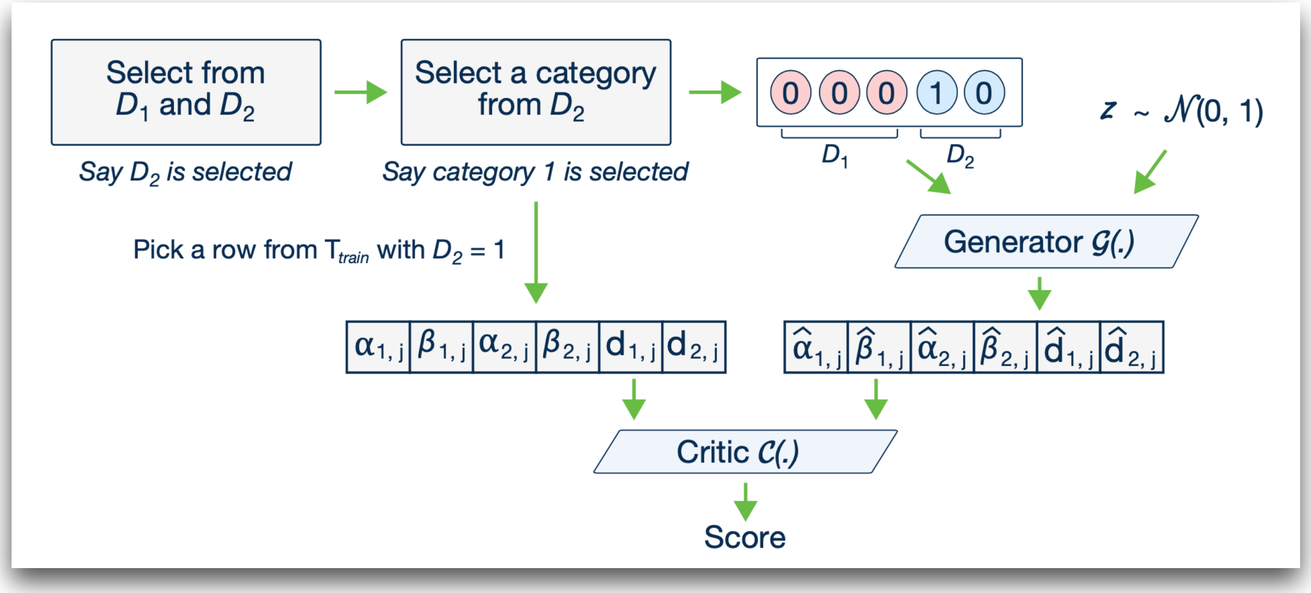
$$\text{cond} = \mathbf{m}_i \otimes \ldots \otimes \mathbf{m}_{N_d}$$

Example

$$D_1 = \{1, 2, 3\} \quad D_2 = \{1, 2\}$$

$$D_2 = 1;\ \mathbf{m}_1 = [0, 0, 0],\ \mathbf{m}_2 = [1, 0]$$

$$\text{cond} = [0, 0, 0, 1, 0]$$



CTGAN

- Generator:

$$
\begin{cases}
h_0 = z \oplus cond \\
h_1 = h_0 \oplus \texttt{ReLU}(\texttt{BN}(\texttt{FC}_{|cond|+|z|\to 256}(h_0))) \\
h_2 = h_1 \oplus \texttt{ReLU}(\texttt{BN}(\texttt{FC}_{|cond|+|z|+256\to 256}(h_1))) \\
\hat{\alpha}_i = \texttt{tanh}(\texttt{FC}_{|cond|+|z|+512\to 1}(h_2)) & 1 \le i \le N_c \\
\hat{\beta}_i = \texttt{gumbel}_{0.2}(\texttt{FC}_{|cond|+|z|+512\to m_i}(h_2)) & 1 \le i \le N_c \\
\hat{\mathbf{d}}_i = \texttt{gumbel}_{0.2}(\texttt{FC}_{|cond|+|z|+512\to |D_i|}(h_2)) & 1 \le i \le N_d
\end{cases}
$$

- Discriminator:

$$
\begin{cases}
h_0 = \mathbf{r}_1 \oplus \ldots \oplus \mathbf{r}_{10} \oplus cond_1 \oplus \ldots \oplus cond_{10} \\
h_1 = \texttt{drop}(\texttt{leaky}_{0.2}(\texttt{FC}_{10|\mathbf{r}|+10|cond|\to 256}(h_0))) \\
h_2 = \texttt{drop}(\texttt{leaky}_{0.2}(\texttt{FC}_{256\to 256}(h_1))) \\
\mathcal{C}(\cdot) = \texttt{FC}_{256\to 1}(h_2)
\end{cases}
$$

Algorithm 1: Train `CTGAN` on step.

Input: Training data $\mathbf{T}_{train}$, Conditional generator and Critic parameters Φ_G and Φ_C respectively, batch size m, pac size pac.

Result: Conditional generator and Critic parameters Φ_G, Φ_C updated.

1 Create masks $\{\mathbf{m}_1, \ldots, \mathbf{m}_{i^*}, \ldots, \mathbf{m}_{N_d}\}_j$, for $1 \le j \le m$

2 Create condition vectors $cond_j$, for $1 \le j \le m$ from masks ▷ Create m conditional vectors

3 Sample $\{z_j\} \sim \texttt{MVN}(0, \mathbf{I})$, for $1 \le j \le m$

4 $\hat{\mathbf{r}}_j \leftarrow \texttt{Generator}(z_j, cond_j)$, for $1 \le j \le m$ ▷ Generate fake data

5 Sample $\mathbf{r}_j \sim \texttt{Uniform}(\mathbf{T}_{train}|cond_j)$, for $1 \le j \le m$ ▷ Get real data

6 $cond_k^{(pac)} \leftarrow cond_{k\times pac+1} \oplus \ldots \oplus cond_{k\times pac+pac}$, for $1 \le k \le m/pac$ ▷ Conditional vector pacs

7 $\hat{\mathbf{r}}_k^{(pac)} \leftarrow \hat{\mathbf{r}}_{k\times pac+1} \oplus \ldots \oplus \hat{\mathbf{r}}_{k\times pac+pac}$, for $1 \le k \le m/pac$ ▷ Fake data pacs

8 $\mathbf{r}_k^{(pac)} \leftarrow \mathbf{r}_{k\times pac+1} \oplus \ldots \oplus \mathbf{r}_{k\times pac+pac}$, for $1 \le k \le m/pac$ ▷ Real data pacs

9 $\mathcal{L}_C \leftarrow \frac{1}{m/pac}\sum_{k=1}^{m/pac} \texttt{Critic}(\hat{\mathbf{r}}_k^{(pac)}, cond_k^{(pac)}) - \frac{1}{m/pac}\sum_{k=1}^{m/pac} \texttt{Critic}(\mathbf{r}_k^{(pac)}, cond_k^{(pac)})$

10 Sample $\rho_1, \ldots, \rho_{m/pac} \sim \texttt{Uniform}(0, 1)$

11 $\tilde{\mathbf{r}}_k^{(pac)} \leftarrow \rho_k \hat{\mathbf{r}}_k^{(pac)} + (1-\rho_k)\mathbf{r}_k^{(pac)}$, for $1 \le k \le m/pac$

12 $\mathcal{L}_{GP} \leftarrow \frac{1}{m/pac}\sum_{k=1}^{m/pac}(||\nabla_{\tilde{\mathbf{r}}_k^{(pac)}} \texttt{Critic}(\tilde{\mathbf{r}}_k^{(pac)}, cond_k^{(pac)})||_2 - 1)^2$ ▷ Gradient Penalty

13 $\Phi_C \leftarrow \Phi_C - 0.0002 \times \texttt{Adam}(\nabla_{\Phi_C}(\mathcal{L}_C + 10\mathcal{L}_{GP}))$

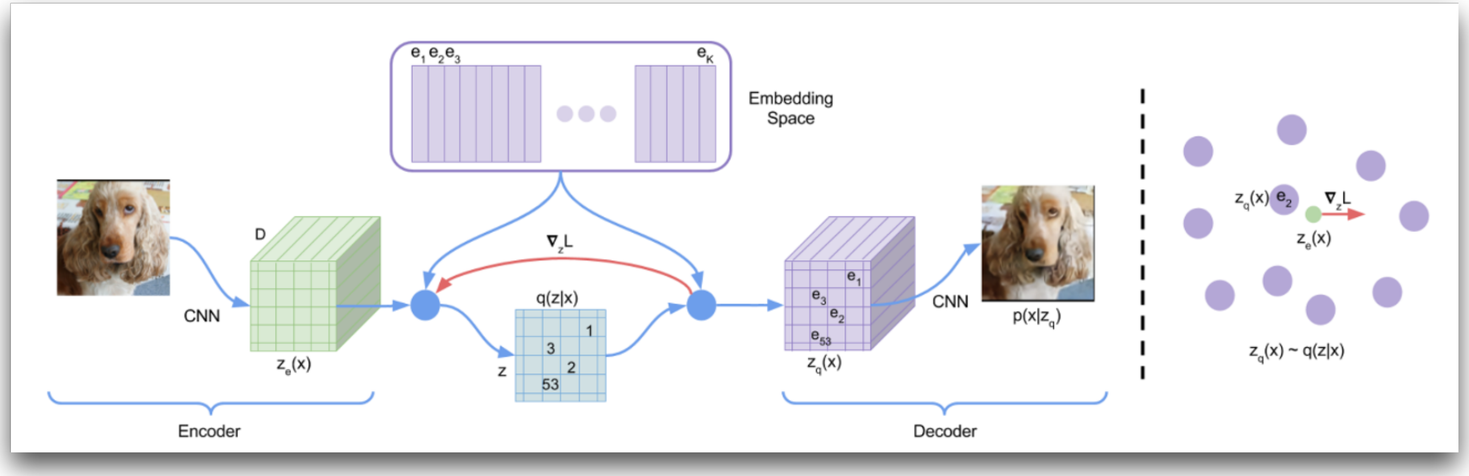
14 Regenerate $\hat{\mathbf{r}}_j$ following lines 1 to 7

15 $\mathcal{L}_G \leftarrow -\frac{1}{m/pac}\sum_{k=1}^{m/pac} \texttt{Critic}(\hat{\mathbf{r}}_k^{(pac)}, cond_k^{(pac)}) + \frac{1}{m}\sum_{j=1}^{m} \texttt{CrossEntropy}(\hat{\mathbf{d}}_{i^*,j}, \mathbf{m}_{i^*})$

16 $\Phi_G \leftarrow \Phi_G - 0.0002 \times \texttt{Adam}(\nabla_{\Phi_G}\mathcal{L}_G)$

VQ-VAE [18]

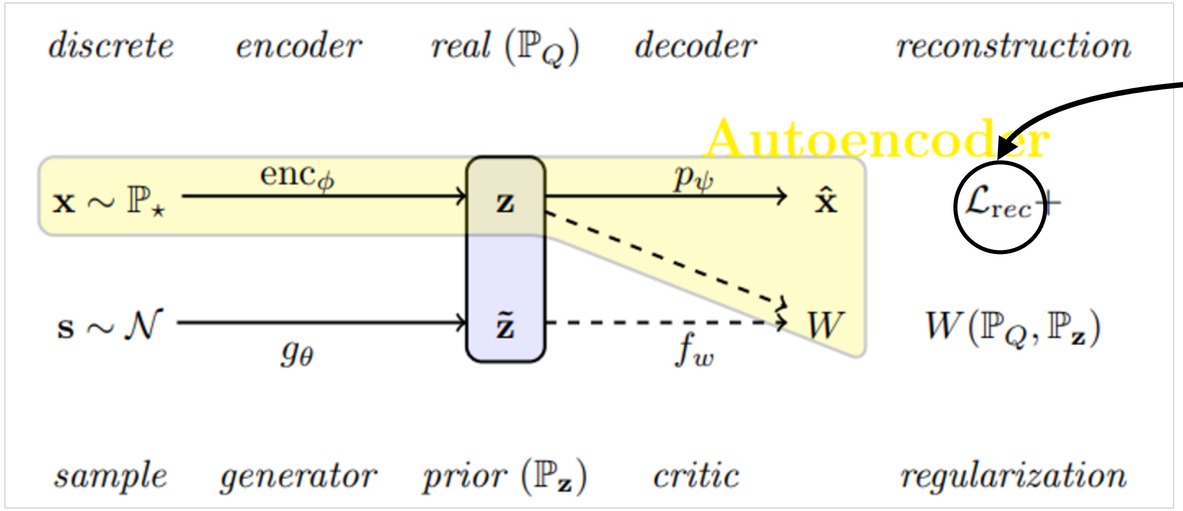
- Vector Quantised-Variational AutoEncoder (VQ-VAE), differs from VAEs in two key ways:
  - Encoder network outputs discrete, rather than continuous, codes;
  - Posteriors and priors in VAEs are assumed normally distributed with diagonal covariance, which allows for the Gaussian re-parametrisation trick to be used
  - In VQ-VAE - the posterior and prior distributions are categorical, and the samples drawn from these distributions index an embedding table
- These embeddings are then used as input into the decoder network



$$q(z=k|x) = \begin{cases} 1 & \text{for } k = \text{argmin}_j \|z_e(x) - e_j\|_2, \\ 0 & \text{otherwise} \end{cases}$$

$$L = \log p(x|z_q(x)) + \|\text{sg}[z_e(x)] - e\|_2^2 + \beta\|z_e(x) - \text{sg}[e]\|_2^2,$$

Adversarially Regularised Auto-Encoder [15]



Algorithm 1 ARAE Training

for each training iteration **do**

(1) Train the encoder/decoder for reconstruction (ϕ, ψ)

Sample $\{\mathbf{x}^{(i)}\}_{i=1}^m \sim \mathbb{P}_\star$ and compute $\mathbf{z}^{(i)} = \text{enc}_\phi(\mathbf{x}^{(i)})$

Backprop loss; $\mathcal{L}_{\text{rec}} = -\frac{1}{m}\sum_{i=1}^m \log p_\psi(\mathbf{x}^{(i)} \mid \mathbf{z}^{(i)})$

(2) Train the critic (w)

Sample $\{\mathbf{x}^{(i)}\}_{i=1}^m \sim \mathbb{P}_\star$ and $\{\mathbf{s}^{(i)}\}_{i=1}^m \sim \mathcal{N}(0, \mathbf{I})$

Compute $\mathbf{z}^{(i)} = \text{enc}_\phi(\mathbf{x}^{(i)})$ and $\tilde{\mathbf{z}}^{(i)} = g_\theta(\mathbf{z}^{(i)})$

Backprop loss $-\frac{1}{m}\sum_{i=1}^m f_w(\mathbf{z}^{(i)}) + \frac{1}{m}\sum_{i=1}^m f_w(\tilde{\mathbf{z}}^{(i)})$

Clip critic w to $[-\epsilon, \epsilon]^d$.

(3) Train the encoder/generator adversarially (ϕ, θ)

Sample $\{\mathbf{x}^{(i)}\}_{i=1}^m \sim \mathbb{P}_\star$ and $\{\mathbf{s}^{(i)}\}_{i=1}^m \sim \mathcal{N}(0, \mathbf{I})$

Compute $\mathbf{z}^{(i)} = \text{enc}_\phi(\mathbf{x}^{(i)})$ and $\tilde{\mathbf{z}}^{(i)} = g_\theta(\mathbf{s}^{(i)})$.

Backprop loss $\frac{1}{m}\sum_{i=1}^m f_w(\mathbf{z}^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(\tilde{\mathbf{z}}^{(i)})$

end for

Session III

Bayesian Networks - Parameter Learning

- Maximize Log-likelihood:

$$\begin{aligned}\mathrm{LL}(\mathcal{G}, \boldsymbol{\theta}) &= \sum_{j=1}^{N} \log \mathrm{P}(y^{(j)}, \mathbf{x}^{(j)}) \\ &= \sum_{j=1}^{N} \left(\log \theta_{y^{(j)}} + \sum_{i=1}^{n} \log \theta_{x_i^{(j)} | y^{(j)}, \Pi(x_i^{(j)})} \right)\end{aligned}$$

- With constraints:

$$\sum_{y \in \mathcal{Y}} \theta_y = 1 \text{ and } \sum_{x_i \in \mathcal{X}_i} \theta_{x_i | y, \Pi(x_i)} = 1$$

- **Theorem:** *Log-likelihood of Equation 1 is maximized given the constraints in Equation 2, when parameters corresponds to empirical probabilities computed from the sample data, i.e.,* $\theta_y = \mathrm{P}_{\mathrm{data}}(y)$ *and* $\theta_{x_i | y, \Pi(x_i)} = \mathrm{P}_{\mathrm{data}}(x_i | y, \Pi(x_i))$.

- Why maximize Log-likelihood? Why not Conditional Log-Likelihood?

Typical Bayesian Network — Parameter Learning

- How about maximizing Conditional Log-likelihood?

$$\mathrm{CLL}(\mathcal{G}, \boldsymbol{\theta}) = \sum_{j=1}^{N} \log \mathrm{P}(y^{(j)} | \mathbf{x}^{(j)})$$

$$= \sum_{j=1}^{N} \left(\log \theta_{y^{(j)}} + \sum_{i=1}^{n} \log \theta_{x_i^{(j)} | y^{(j)}, \Pi(x_i^{(j)})} \right) - \log \left(\sum_{c=1}^{\mathcal{Y}} \theta_c \prod_{i=1}^{n} \theta_{x_i^{(j)} | c, \Pi(x_i^{(j)})} \right)$$

Extending to include Normalisation term (Denominator)

$$= \sum_{j=1}^{N} \left(\log \theta_{y^{(j)}} + \sum_{i=1}^{n} \log \theta_{x_i^{(j)} | y^{(j)}} \right) - \log \left(\sum_{c=1}^{\mathcal{Y}} \theta_c \prod_{i=1}^{n} \theta_{x_i^{(j)} | c} \right)$$

Getting rid of Parents Function Π

$$= \sum_{j=1}^{N} \left(\log \theta_{y^{(j)}} + \sum_{i=1}^{n} \log \theta_{x_i^{(j)} | y^{(j)}} \right) - \log \left(\sum_{c=1}^{\mathcal{Y}} \exp \left(\log \theta_c + \sum_{i=1}^{n} \log \theta_{x_i^{(j)} | c} \right) \right)$$

Taking exp and log in the normalisation term

Typical Bayesian Network — Parameter Learning

- How about maximizing Conditional Log-likelihood?

$$\mathrm{CLL}(\mathcal{G}, \boldsymbol{\theta}) = \sum_{j=1}^{N} \log \mathrm{P}(y^{(j)} | \mathbf{x}^{(j)})$$

$$= \sum_{j=1}^{N} \left(\log \theta_{y^{(j)}} + \sum_{i=1}^{n} \log \theta_{x_i^{(j)}|y^{(j)}} \right) - \log \left(\sum_{c=1}^{\mathcal{Y}} \exp \left(\log \theta_c + \sum_{i=1}^{n} \log \theta_{x_i^{(j)}|c} \right) \right)$$

Copied from Previous Slide

$$= \sum_{j=1}^{N} \left(\beta_{y^{(j)}} + \sum_{i=1}^{n} \beta_{x_i^{(j)}|y^{(j)}} \right) - \log \left(\sum_{c=1}^{\mathcal{Y}} \exp \left(\beta_c + \sum_{i=1}^{n} \beta_{x_i^{(j)}|c} \right) \right)$$

Reparameterizing Log of thetas with betas

$$= \sum_{j=1}^{N} \log \frac{\exp \left(\beta_{y^{(j)}} + \sum_{i=1}^{n} \beta_{x_i^{(j)}|y^{(j)}} \right)}{\sum_{c=1}^{\mathcal{Y}} \exp \left(\beta_c + \sum_{i=1}^{n} \beta_{x_i^{(j)}|c} \right)}$$

Factoring the log back

Typical Bayesian Network — Parameter Learning [23]

- How about maximizing Conditional Log-likelihood?

$$\mathrm{CLL}(\mathcal{G}, \boldsymbol{\theta}) = \sum_{j=1}^{N} \log \mathrm{P}(y^{(j)} | \mathbf{x}^{(j)})$$

$$= \sum_{j=1}^{N} \left(\log \theta_{y^{(j)}} + \sum_{i=1}^{n} \log \theta_{x_i^{(j)}|y^{(j)}} \right) - \log \left(\sum_{c=1}^{\mathcal{Y}} \exp \left(\log \theta_c + \sum_{i=1}^{n} \log \theta_{x_i^{(j)}|c} \right) \right)$$

Copied from Previous Slide

$$= \sum_{j=1}^{N} \left(\beta_{y^{(j)}} + \sum_{i=1}^{n} \beta_{x_i^{(j)}|y^{(j)}} \right) - \log \left(\sum_{c=1}^{\mathcal{Y}} \exp \left(\beta_c + \sum_{i=1}^{n} \beta_{x_i^{(j)}|c} \right) \right)$$

Copied from Previous Slide

Over-parameterizing

$$= \sum_{j=1}^{N} \left(\beta_{y^{(j)}} \log \theta_{y^{(j)}} + \sum_{i=1}^{n} \beta_{x_i^{(j)}|y^{(j)}} \log \theta_{x_i^{(j)}|y^{(j)}} \right) - \log \left(\sum_{c=1}^{\mathcal{Y}} \exp \left(\beta_c \log \theta_c + \sum_{i=1}^{n} \beta_{x_i^{(j)}|c} \log \theta_{x_i^{(j)}|c} \right) \right)$$

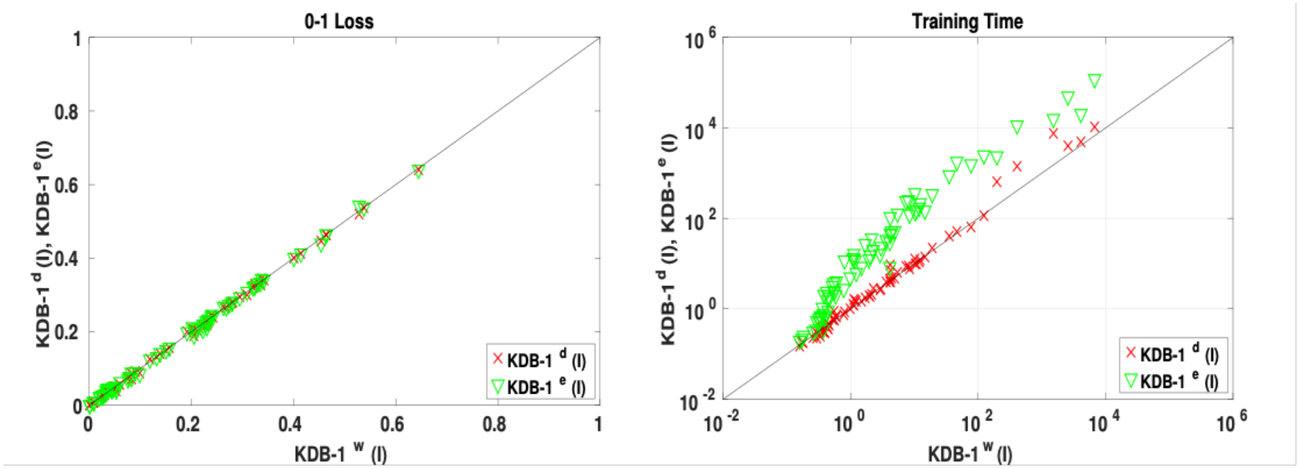
Typical Bayesian Network — Parameter Learning [23]

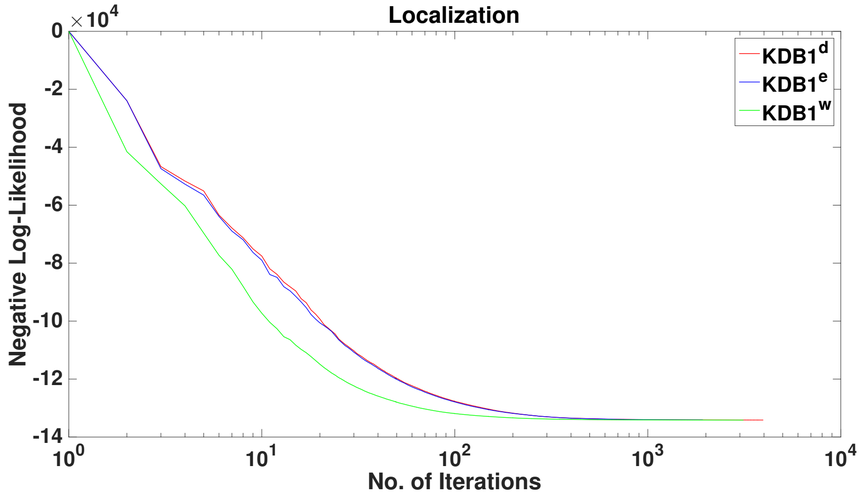
- How about maximizing Conditional Log-likelihood?

$$\mathrm{CLL}(\mathcal{G}, \boldsymbol{\theta}) = \sum_{j=1}^{N} \log \mathrm{P}(y^{(j)} | \mathbf{x}^{(j)})$$

Over-parameterizing

$$= \sum_{j=1}^{N} \left(\beta_{y^{(j)}} \log \theta_{y^{(j)}} + \sum_{i=1}^{n} \beta_{x_i^{(j)}|y^{(j)}} \log \theta_{x_i^{(j)}|y^{(j)}} \right) - \log \left(\sum_{c=1}^{\mathcal{Y}} \exp \left(\beta_c \log \theta_c + \sum_{i=1}^{n} \beta_{x_i^{(j)}|c} \log \theta_{x_i^{(j)}|c} \right) \right)$$





Typical Bayesian Network — Parameter Learning

- How about maximizing Conditional Log-likelihood?

$$\mathrm{CLL}(\mathcal{G}, \boldsymbol{\theta}) = \sum_{j=1}^{N} \log \mathrm{P}(y^{(j)} | \mathbf{x}^{(j)})$$

$$= \sum_{j=1}^{N} \left(\beta_{y^{(j)}} + \sum_{i=1}^{n} \beta_{x_i^{(j)} | y^{(j)}} \right) - \log \left(\sum_{c=1}^{\mathcal{Y}} \exp \left(\beta_c + \sum_{i=1}^{n} \beta_{x_i^{(j)} | c} \right) \right)$$

Copied from Previous Slide

$$= \sum_{j=1}^{N} \left(\beta_{y^{(j)}} + \sum_{i=1}^{n} \beta_{x_i^{(j)} | y^{(j)}, \Pi(x_i^{(j)})} \right) - \log \left(\sum_{c=1}^{\mathcal{Y}} \exp \left(\beta_c + \sum_{i=1}^{n} \beta_{x_i^{(j)} | c, \Pi(x_i^{(j)})} \right) \right)$$

Introducing Parent Function Π

$$= \sum_{j=1}^{N} \left(\theta_{y^{(j)}} + \sum_{i=1}^{n} \theta_{x_i^{(j)} | y^{(j)}, \Pi(x_i^{(j)})} \right) - \log \left(\sum_{c=1}^{\mathcal{Y}} \exp \left(\theta_c + \sum_{i=1}^{n} \theta_{x_i^{(j)} | c, \Pi(x_i^{(j)})} \right) \right)$$

Changing betas to thetas

Typical Bayesian Network — Parameter Learning

- How about maximizing Conditional Log-likelihood?

$$\mathrm{CLL}(\mathcal{G}, \boldsymbol{\theta}) = \sum_{j=1}^{N} \log \mathrm{P}(y^{(j)} | \mathbf{x}^{(j)})$$

$$= \sum_{j=1}^{N} \left(\theta_{y^{(j)}} + \sum_{i=1}^{n} \theta_{x_i^{(j)} | y^{(j)}, \Pi(x_i^{(j)})} \right) - \log \left(\sum_{c=1}^{\mathcal{Y}} \exp \left(\theta_c + \sum_{i=1}^{n} \theta_{x_i^{(j)} | c, \Pi(x_i^{(j)})} \right) \right)$$

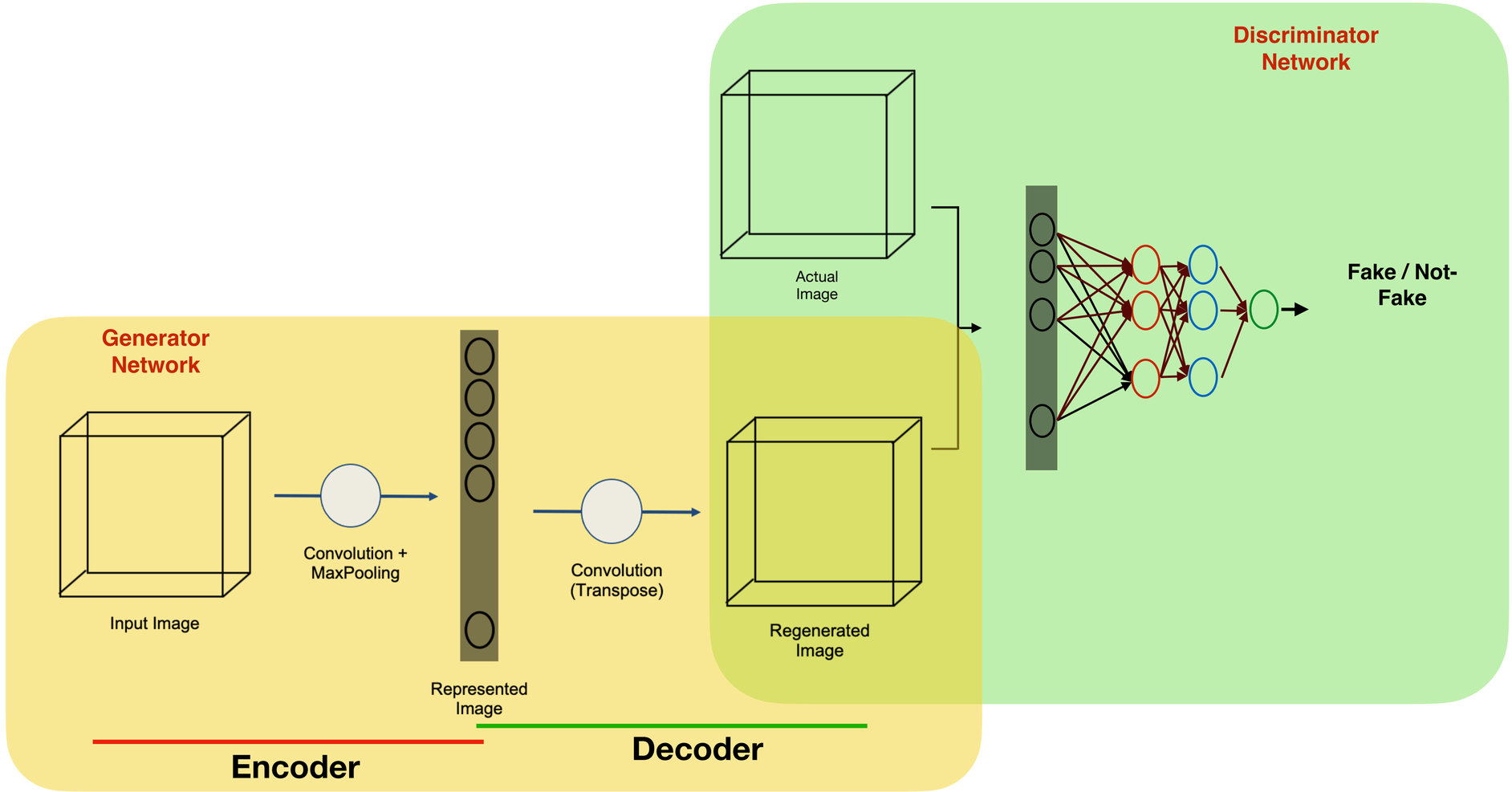
Constraints

$$\sum_{y \in \mathcal{Y}} \theta_y = 1 \text{ and } \sum_{x_i \in \mathcal{X}_i} \theta_{x_i | y, \Pi(x_i)} = 1$$

No Constraints

- Iteration-based Optimization - SGD (Adam)

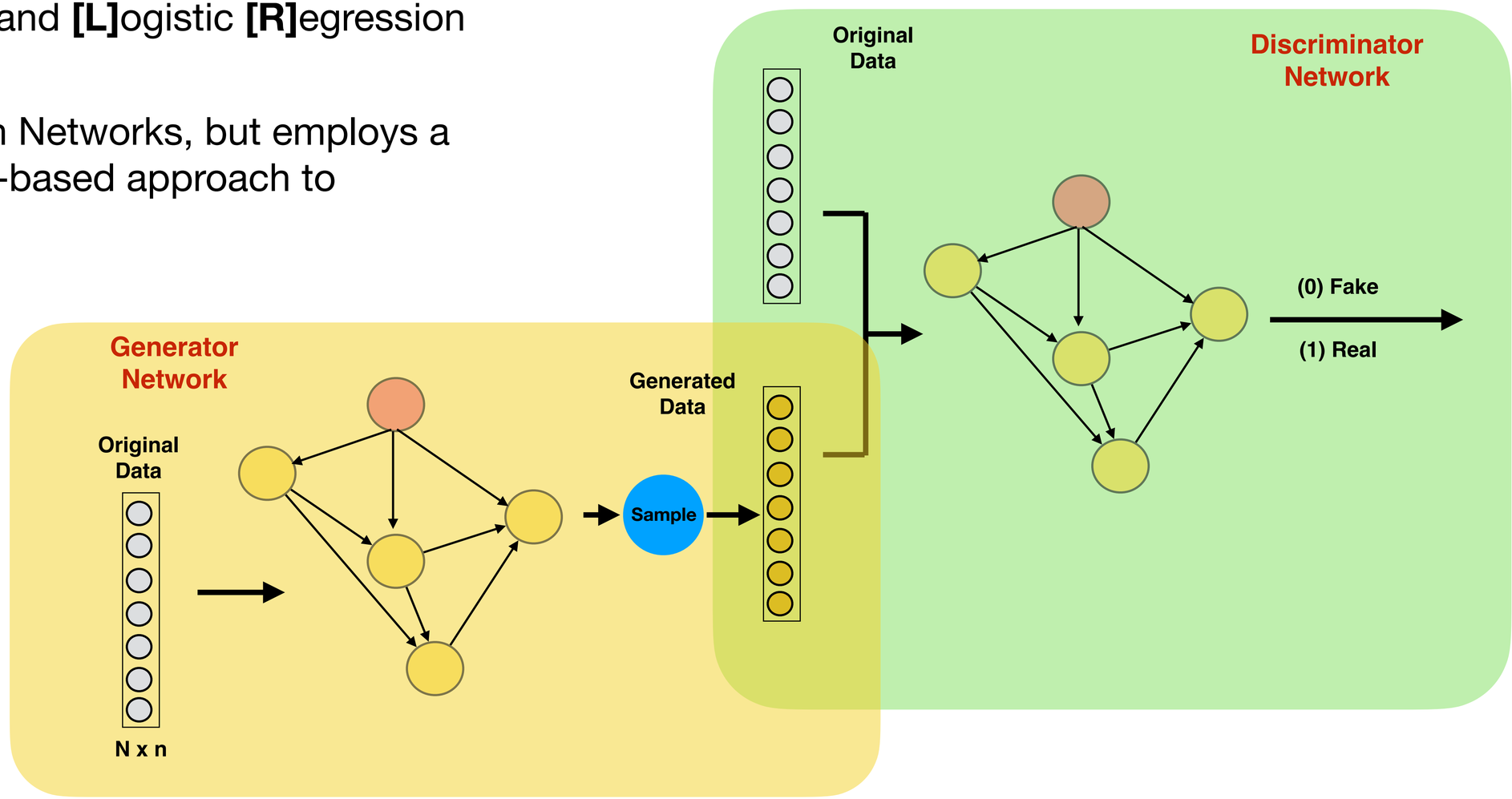
Models for Structured Data



GANBLR [24]

- **[G]**enerative **[A]**dversarial **[N]**etworks Inspired from **[N]**aive **[B]**ayes and **[L]**ogistic **[R]**egression

- Make use of Bayesian Networks, but employs a game-theoretic, GAN-based approach to learning



GANBLR

GANBLR's Objective Function

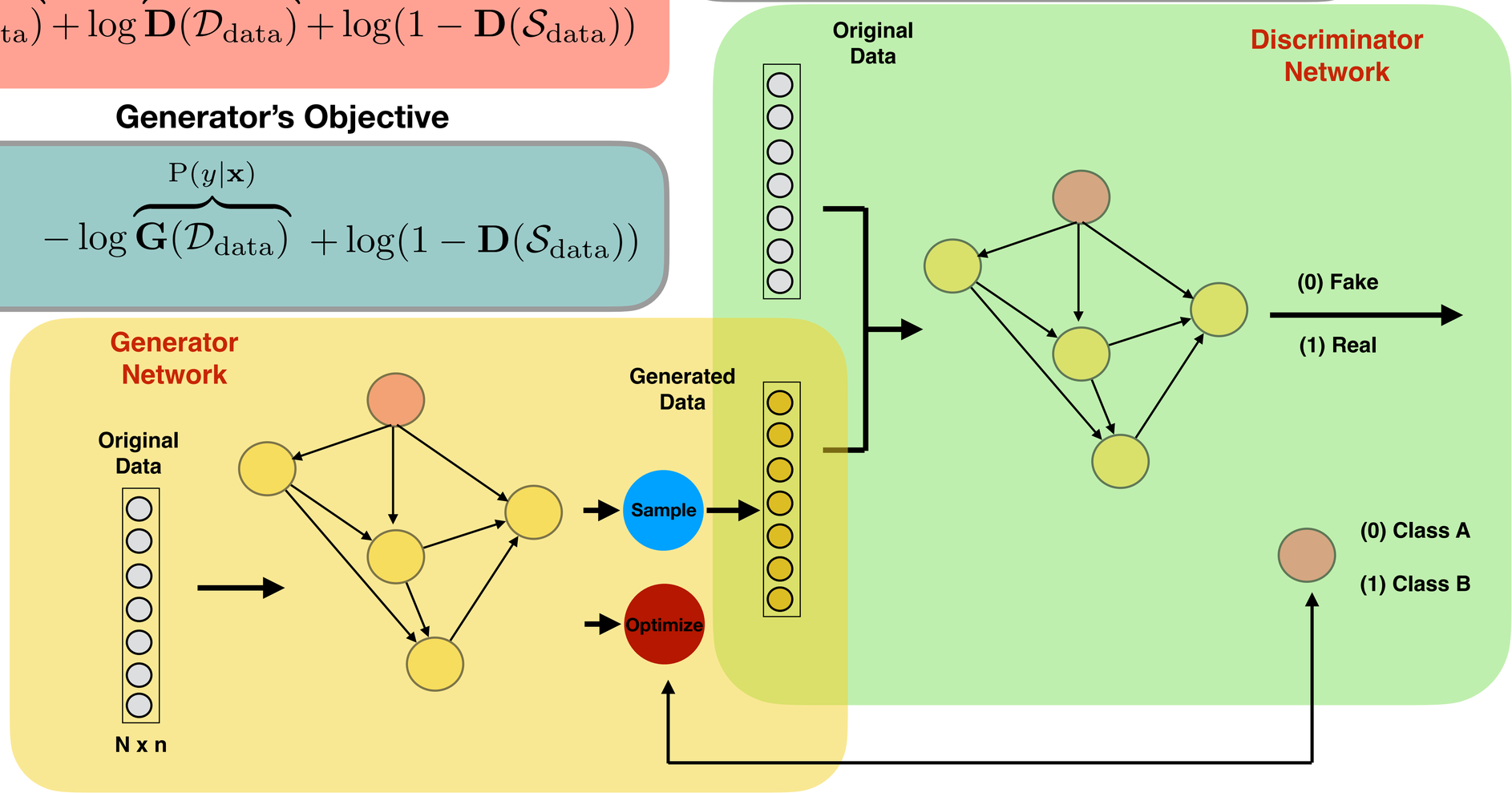
$$\min_{\boldsymbol{\theta_g}} \max_{\boldsymbol{\theta_d}} \quad -\log \overbrace{\mathbf{G}(\mathcal{D}_{\text{data}})}^{\mathrm{P}(y|\mathbf{x})} + \log \overbrace{\mathbf{D}(\mathcal{D}_{\text{data}})}^{\mathrm{P}(z|\mathbf{x})} + \log(1 - \mathbf{D}(\mathcal{S}_{\text{data}}))$$

Discriminator's Objective

$$\max_{\boldsymbol{\theta_d}} \quad \log \overbrace{\mathbf{D}(\mathcal{D}_{\text{data}})}^{\mathrm{P}(z|\mathbf{x})} + \log(1 - \mathbf{D}(\overbrace{\mathcal{S}_{\text{data}}}^{\mathbf{G}(.)}))$$

Generator's Objective

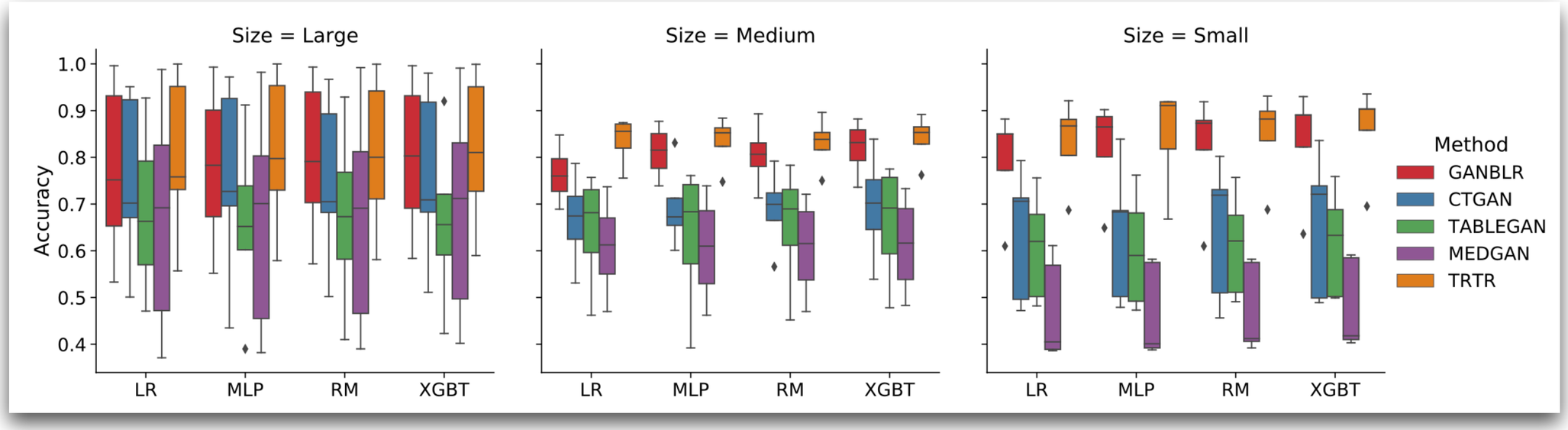
$$\min_{\boldsymbol{\theta_g}} \quad -\log \overbrace{\mathbf{G}(\mathcal{D}_{\text{data}})}^{\mathrm{P}(y|\mathbf{x})} + \log(1 - \mathbf{D}(\mathcal{S}_{\text{data}}))$$



Comparative Analysis — Performance

Method	L˜Accuracy%	M˜Accuracy%	S˜Accuracy%
	TRTR		
	80.84%	86.80%	84.62%
	TSTR		
GANBLR	**78.85%**	**84.02%**	**81.67%**
CTGAN	75.27%	74.88%	64.37%
TableGAN	66.95%	71.72%	60.87%
MedGAN	67.28%	68.58%	47.36%

Method	Large		Medium		Small	
	JSD	WD	JSD	WD	JSD	WD
GANBLR	0.136	**0.619**	0.301	0.662	**0.263**	**0.783**
CTGAN	**0.133**	0.621	**0.287**	0.672	0.330	0.810
TableGAN	0.142	0.701	0.350	**0.613**	0.531	1.220
MedGAN	0.190	0.730	0.392	0.690	0.380	1.021



Interpretability

Class (C)	Buying (B=3)	Maint (M=2)	Doors (D=0)	Persons (P=1)	Lug_boot (L=2)	Safety (S=0)
P(C=0|B, M, D, P, L, S) = **0.4892**	P(C=0|B=3) = 0.131	P(C=0|M=2) = 0.352	P(C=0|D=0)=0.186	P(C=0|P=1)=**0.460**	P(C=0|L=2)=0.205	P(C=0|S=0)=**0.572**
P(C=2|B, M, D, P, L, S)=0.4211	P(C=2|B=3) = **0.288**	P(C=2|M=2) = 0.202	P(C=2|D=0)=0.246	P(C=2|P=1)=0.154	P(C=2|L=2)=**0.307**	P(C=2|S=0)=0.200
P(C=1|B, M, D, P, L, S)=0.0462	P(C=1|B=3) = 0.099	P(C=1|M=2) = 0.242	P(C=1|D=0)=0.241	P(C=1|P=1)=0.340	P(C=1|L=2)=0.322	P(C=1|S=0)=0.374
P(C=3|B, M, D, P, L, S)=0.0436	P(C=3|B=3) = 0.124	P(C=3|M=2) = 0.266	P(C=3|D=0)=0.172	P(C=3|P=1)=0.405	P(C=3|L=2)=0.165	P(C=3|S=0)=0.682

Interpretation of GANBLR's generator weights at $epoch = 50$ on one instance of car dataset.

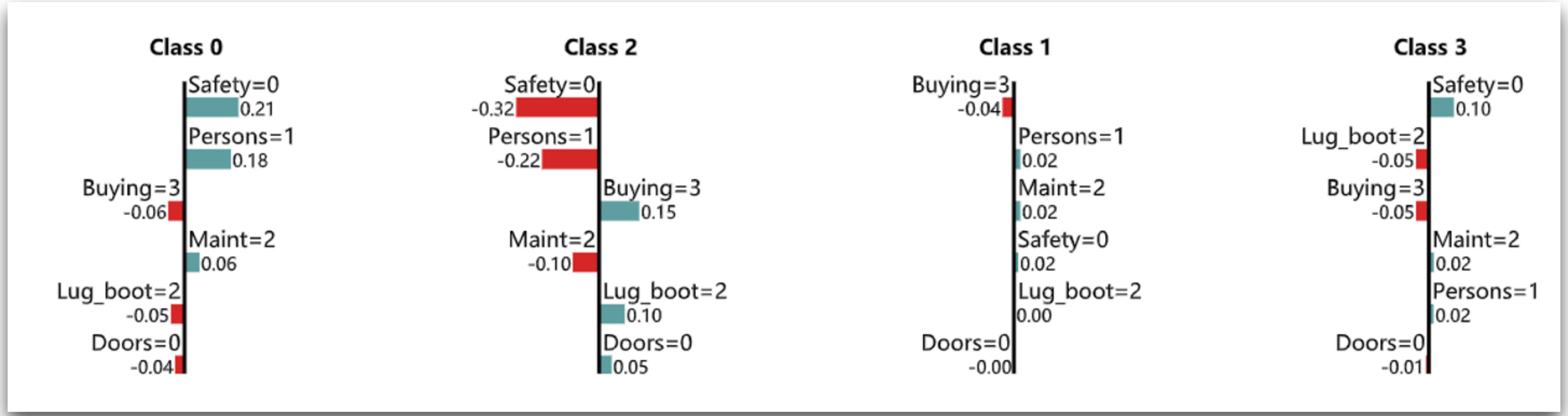
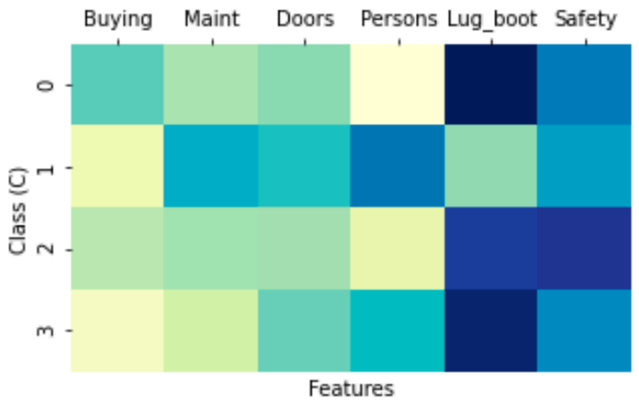


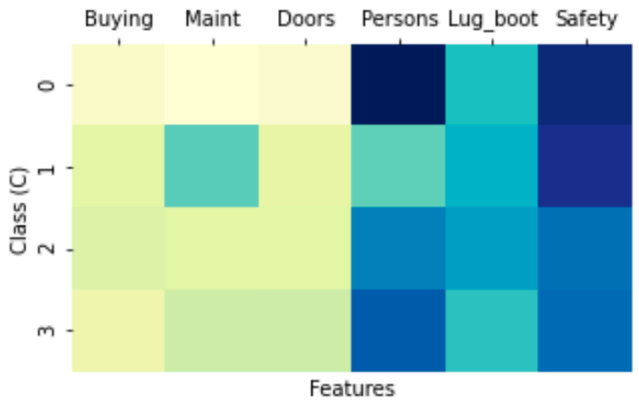
Figure 3: LIME explanation for synthetic data after training using XGBT*

* Probability of each class value are $P(C = 0) = 0.76, P(C = 2) = 0.22, P(C = 1) = 0.02, P(C = 3) = 0.01$

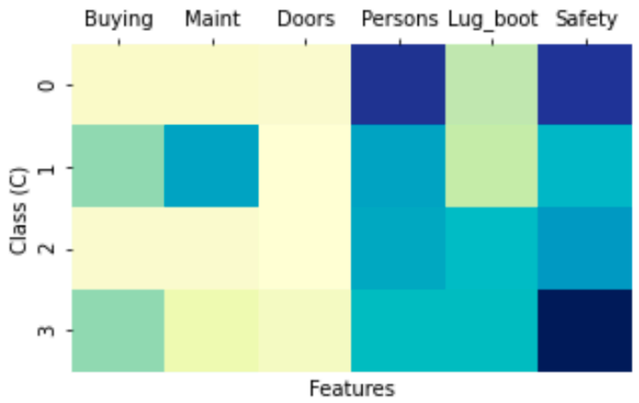
Interpretability



(a) $epoch = 1$, training accuracy $= 0.26$



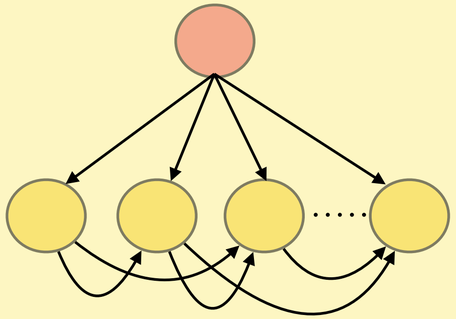
(b) $epoch = 50$, training accuracy $= 0.76$



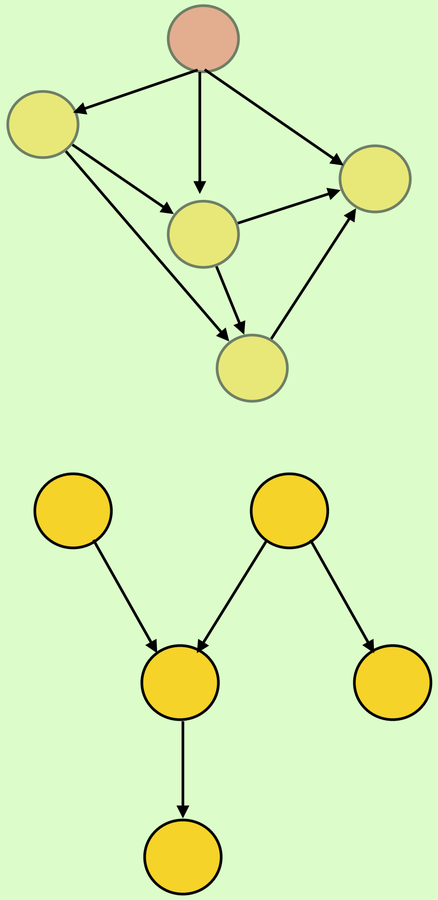
(c) $epoch = 100$, training accuracy $= 0.85$

Overall feature impact during GANBLR training

Use of Unrestricted Bayesian Network



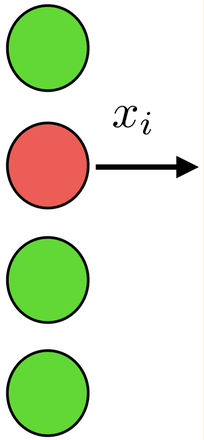
Restricted



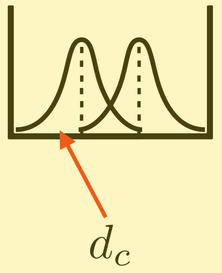
Un-Restricted

Method	L˜Accuracy%	M˜Accuracy%	S˜Accuracy%
	TRTR		
	80.84%	86.80%	84.62%
	TSTR		
GANBLR++ (Disc)	**79.32%**	**86.10%**	**82.56%**
GANBLR	78.85%	84.02%	81.67%
CTGAN	75.27%	74.88%	64.37%
TableGAN	66.95%	71.72%	60.87%
MedGAN	67.28%	68.58%	47.36%

Handling Numeric Values



x_i



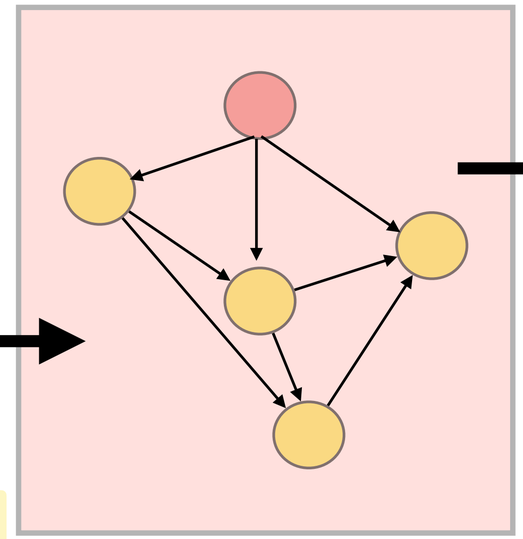
d_c

Cluster Center

$$\alpha_c = \phi(d_c),$$

$$d_c = \left(\frac{x_i - \mu_c}{S_c}\right),$$

$$\phi(d_c) = \begin{cases} +1|-1 & |d_c| \leq S_c \wedge d_c > 0 | d_c < 0; \\ +2|-2 & S_c < |d_c| \leq 2S_c \wedge d_c > 0 | d_c < 0; \\ +3|-3 & 2S_c < |d_c| \leq 3S_c \wedge d_c > 0 | d_c < 0; \\ \infty|-\infty & |d_c| \geq 3S_c \wedge d_c > 0 | d_c < 0. \end{cases}$$



Sample

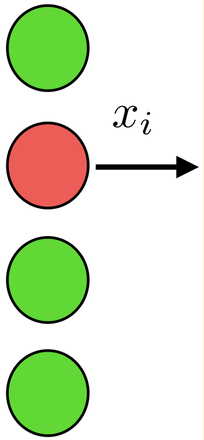
$$\bar{\mathbf{x}} \sim \mathcal{N}(\bar{\mathbf{x}}|\mu_c, S_c)$$

$$\text{Such that} \quad \alpha_c < |\frac{\bar{\mathbf{x}} - \mu_c}{S_c}| \leq \alpha_c + 1$$

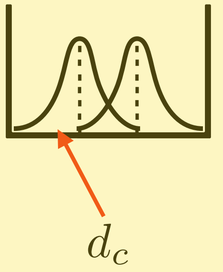
$$\text{sign}(\bar{\mathbf{x}} - \mu_c) = \text{sign}(\alpha_c)$$

Mode Specific Sampling

Handling Numeric Values



x_i

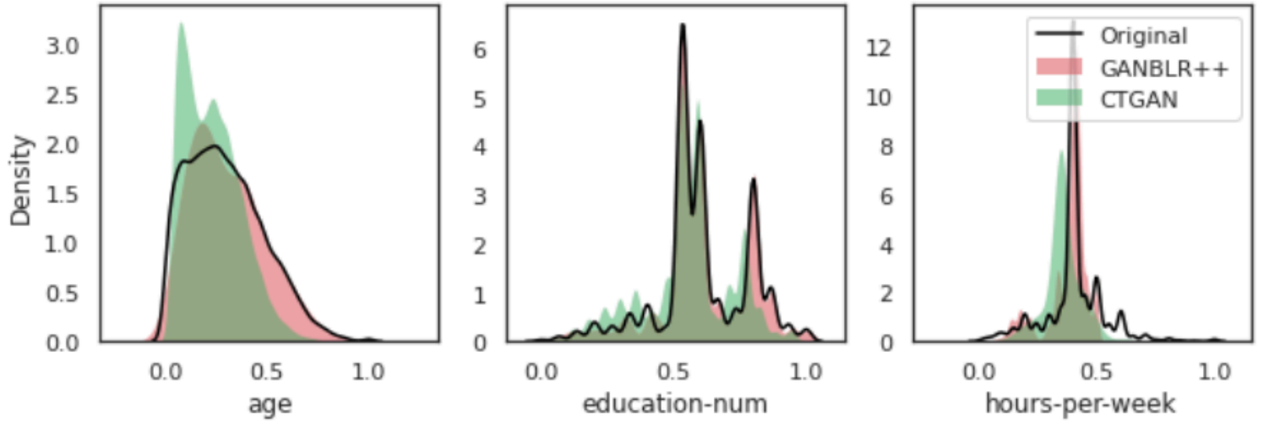


Cluster Center

$$\alpha_c = \phi(d_c),$$

$$d_c = \left(\frac{x_i - \mu_c}{S_c}\right),$$

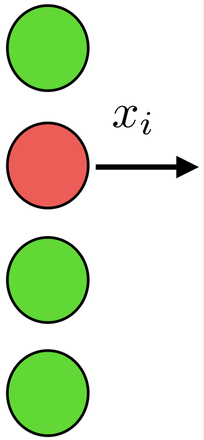
$$\phi(d_c) = \begin{cases} +1|-1 & |d_c| \leq S_c \wedge d_c > 0 | d_c < 0; \\ +2|-2 & S_c < |d_c| \leq 2S_c \wedge d_c > 0 | d_c < 0; \\ +3|-3 & 2S_c < |d_c| \leq 3S_c \wedge d_c > 0 | d_c < 0; \\ \infty|-\infty & |d_c| \geq 3S_c \wedge d_c > 0 | d_c < 0. \end{cases}$$



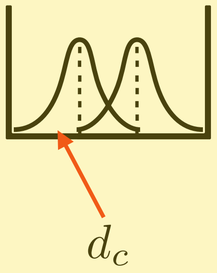
	Large	Medium	Small
GANBLR++ Performance	$\tau = 1/\tau = 0$	$\tau = 1/\tau = 0$	$\tau = 1/\tau = 0$
	-8.13%	-5.70%	-7.82%

Cluster Center vs. Dirichlet Mode Discretization

Handling Numeric Values



x_i



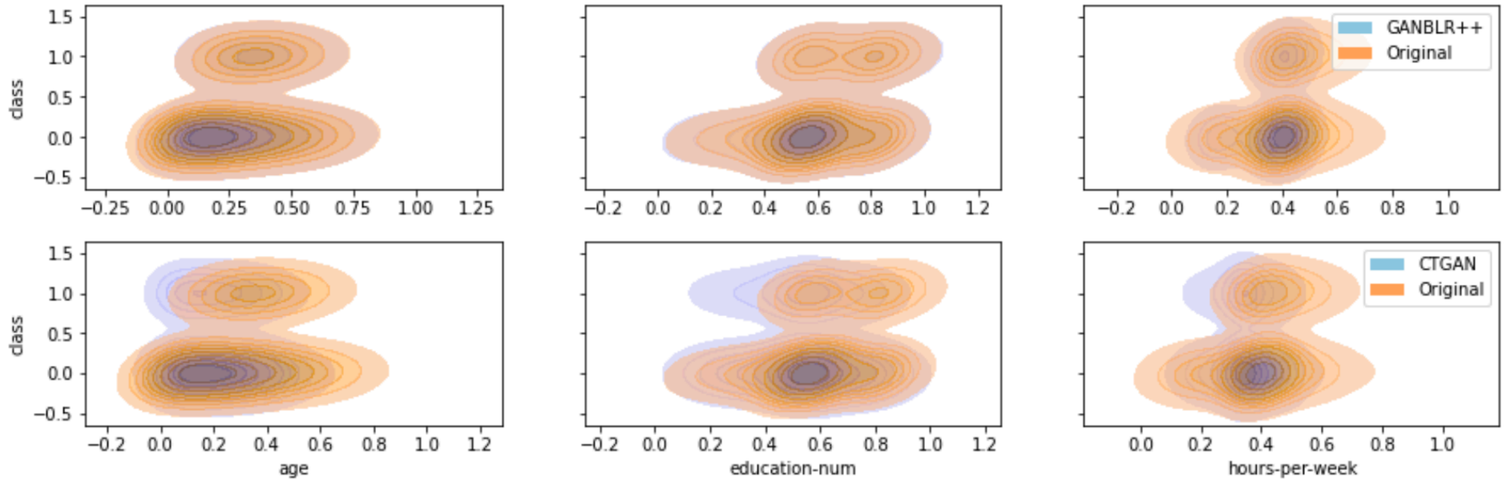
d_c

Cluster Center

$$\alpha_c = \phi(d_c),$$

$$d_c = \left(\frac{x_i - \mu_c}{S_c}\right),$$

$$\phi(d_c) = \begin{cases} +1|-1 & |d_c| \leq S_c \wedge d_c > 0 | d_c < 0; \\ +2|-2 & S_c < |d_c| \leq 2S_c \wedge d_c > 0 | d_c < 0; \\ +3|-3 & 2S_c < |d_c| \leq 3S_c \wedge d_c > 0 | d_c < 0; \\ \infty|-\infty & |d_c| \geq 3S_c \wedge d_c > 0 | d_c < 0. \end{cases}$$



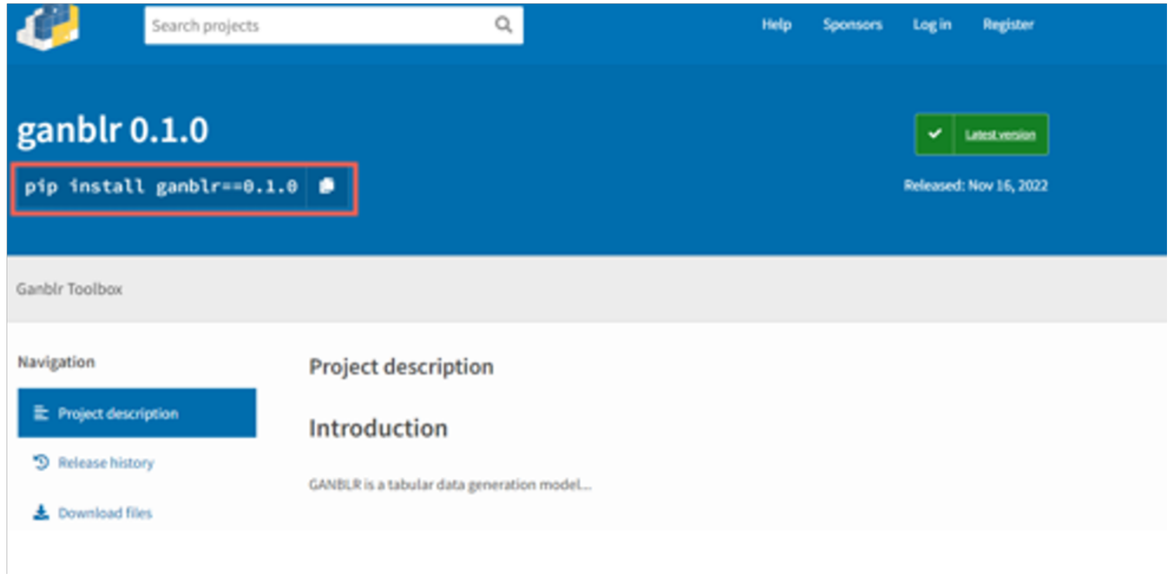
Method	L˜Accuracy%	M˜Accuracy%	S˜Accuracy%
		TRTR	
	78.90%	86.27%	82.69%
		TSTR	
GANBLR++	**78.88%**	**85.57%**	**81.30%**
CTGAN	77.27%	72.11%	66.77%
TableGAN	63.65%	70.29%	58.29%
MedGAN	65.71%	66.36%	48.29%

Session IV (Lab)

Pip Download

- GANBLR is a library using python to enable the synthetic tabular data generation
- It is available to install via pip command and become the repository of python package index group
- The dependencies are listed as below which will be installed via the pip command.

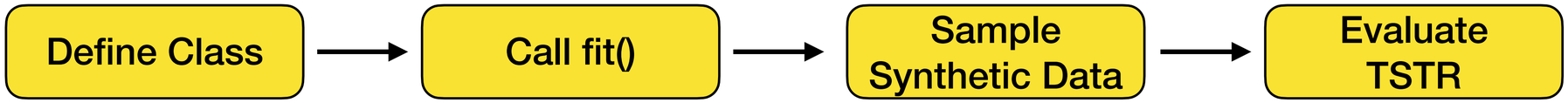
```
install_requires = ["numpy", "pandas", "tensorflow>=2.3", "scikit-learn>=0.24", "pyitlib", "pgmpy"]
```



- There are three major class under the GANBLR library:
  - **GANBLR** – The main class to set up the GANBLR model.
  - **DMMdiscritizer** – The class to convert the numerical column to discretised column via Dirichlet Mode Discretization
  - **GANBLRPP** – The main class to set up the GANBLR++ model

GANBLR Class

- There are 4 main parameters in this class:
  - **K** – the order of the feature interaction during the generation
  - **Batch-size** – the size of the batch for training the generator for GANBLR
  - **Epochs** – Number of epochs, to train both generator and discriminator
  - **Warmup_epochs** – The epochs needed from total to run the generator solely




```
model = GANBLR()
```

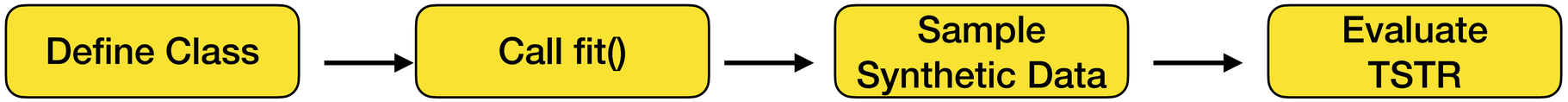
```
model.fit(X_train, y_train, k = 0, epochs = 10, batch_size=64)
```

```
size = 1000

syn_data = model.sample(size)
```

```
acc_score_lr = model.evaluate(X_test, y_test, model='lr')
acc_score_mlp = model.evaluate(X_test, y_test, model='mlp')
acc_score_rf = model.evaluate(X_test, y_test, model='rf')
```

GANBLRPP Class




```
from ganblr import GANBLRPP
ganblrpp = GANBLRPP(numerical_columns)
ganblrpp.fit(X_train, y_train, epochs=10)
```

```
size = 1000
syn_data = ganblrpp.sample(size)
```

```
acc_score_lr  = ganblrpp.evaluate(X_test, y_test, model='lr')
acc_score_mlp = ganblrpp.evaluate(X_test, y_test, model='mlp')
acc_score_rf  = ganblrpp.evaluate(X_test, y_test, model='rf')
```

DMMDiscritizer Class

- It is a class used inside of the GANBLR++
    - The DMMdiscritizer is to discretize the numerical column into Dirichlet Mode Discretization

```
class GANBLRPP:

    def __init__(self, numerical_columns, random_state=None):
        self.__discritizer = DMMDiscritizer(random_state)
        self.__ganblr = GANBLR()
        self._numerical_columns = numerical_columns
        pass
```

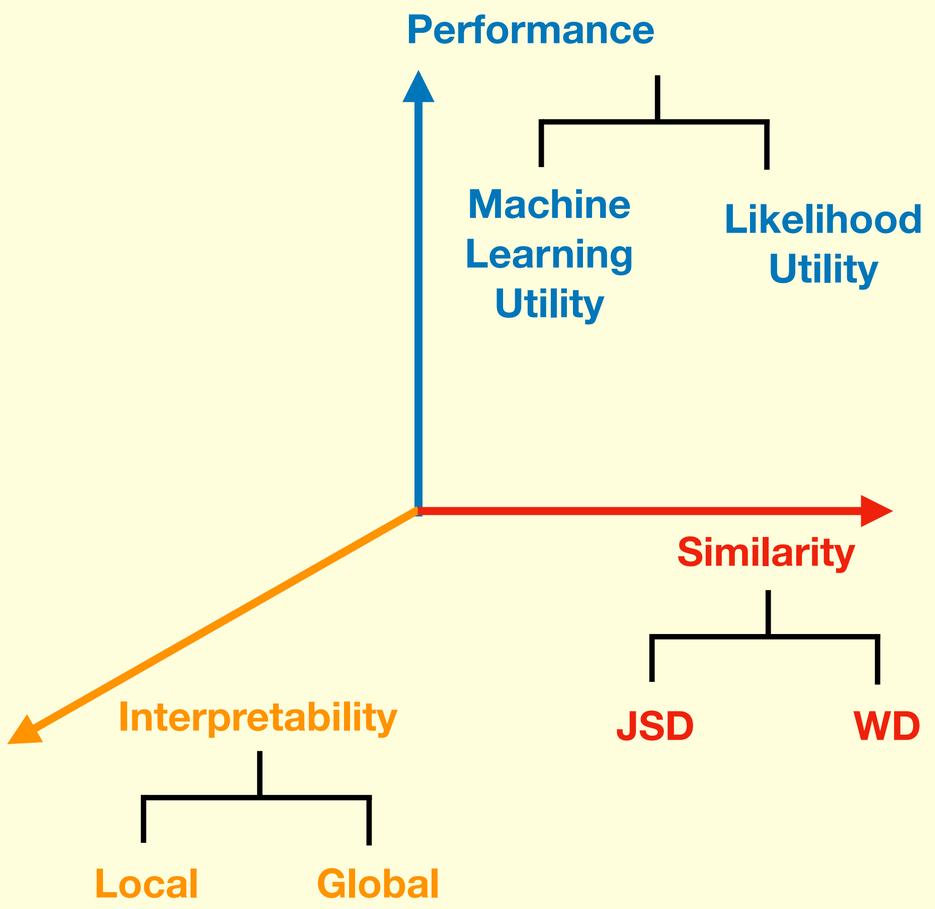
```
numerical_columns = self._numerical_columns
x[:,numerical_columns] = self.__discritizer.fit_transform(x[:,numerical_columns])
return self.__ganblr.fit(x, y, k, batch_size, epochs, warmup_epochs, verbose)
```

Session V

Miscellaneous Issues

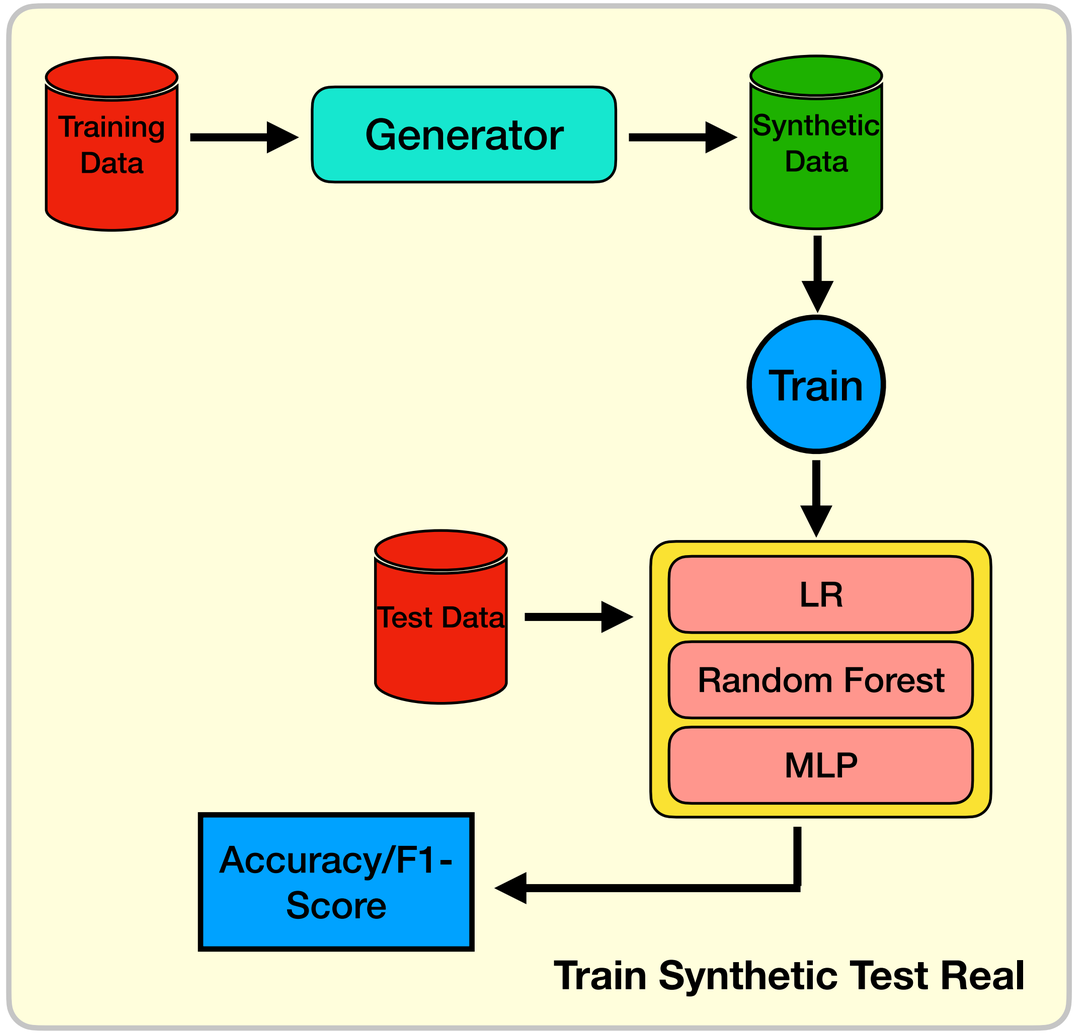
- How to evaluate data generation model?
  - Machine Learning Utility
  - Similarity measurement
  - Interpretability
  - Other measures
- Need for a Unified Framework
- Application
  - I - Record generation
  - II - Privacy Preserving
  - III - Fairness Enforcement
  - IV - Missing values Imputation
  - V - Adversarial Generation
- Conclusion / Q&A

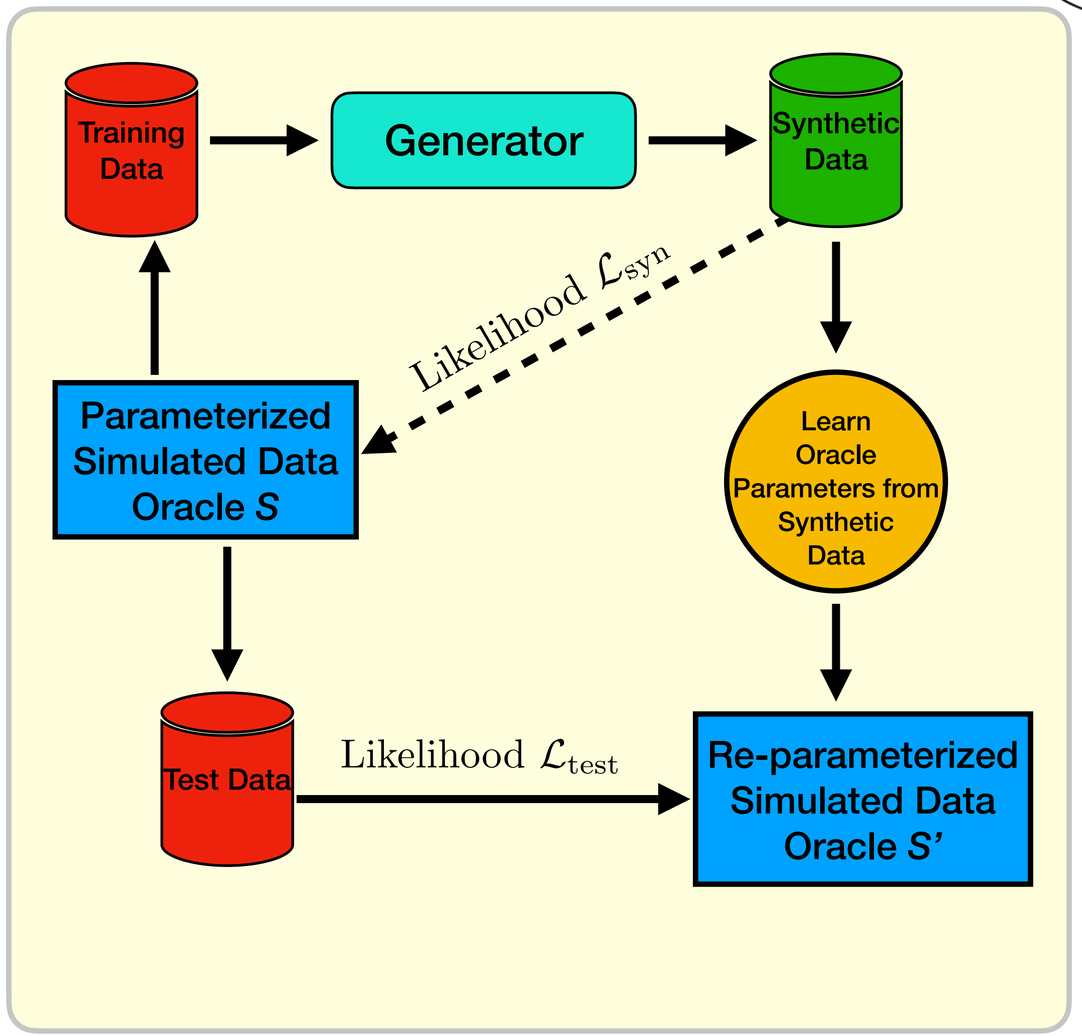
Performance Evaluation



Is new data differentially private?	How much Fairness is enforced?
How much new data is robust to adversarial attacks?	Can new data do a better job at anomaly detection?
Can missing values be treated?	Many Others

Performance





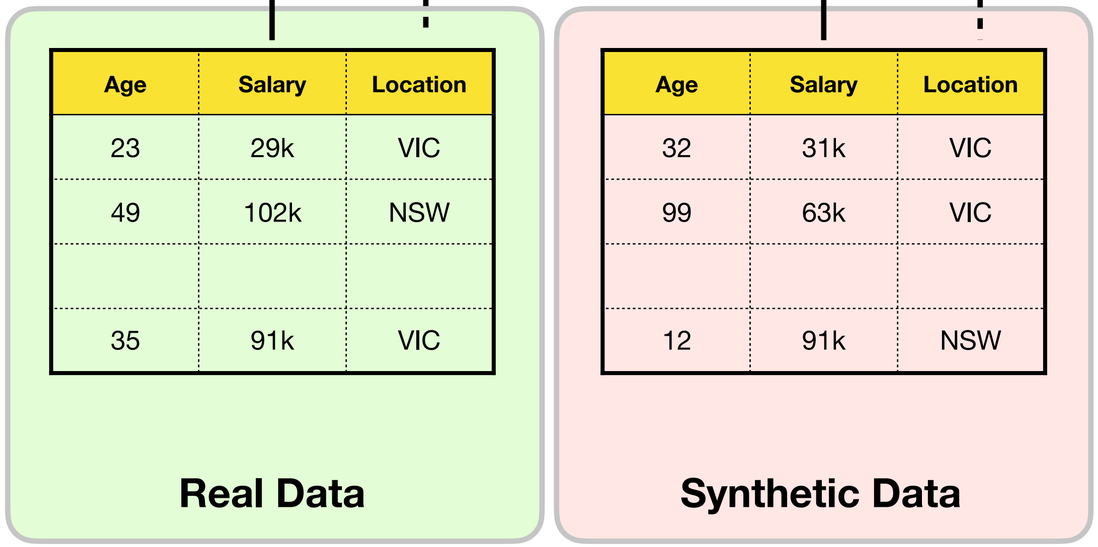
Machine Learning Utility

Real Data

Likelihood Utility

Simulated Data

Similarity



Age	Salary	Location
23	29k	VIC
49	102k	NSW
35	91k	VIC

Real Data

Age	Salary	Location
32	31k	VIC
99	63k	VIC
12	91k	NSW

Synthetic Data

$$\text{Similarity}(\text{Location}_{\text{Real}}, \text{Location}_{\text{Synthetic}})$$

$$\text{Similarity}(\text{Salary}_{\text{Real}}, \text{Salary}_{\text{Synthetic}})$$

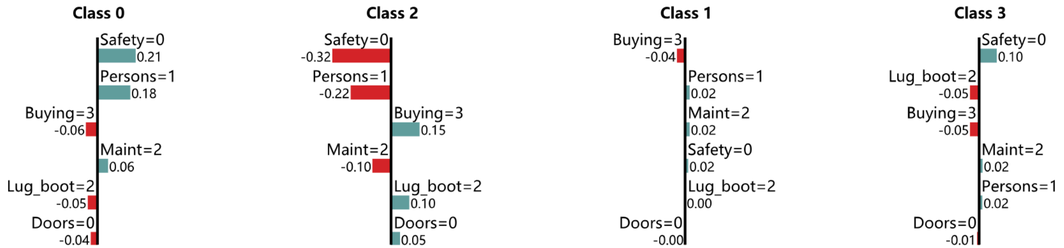
$$\text{Similarity}(\text{Real}, \text{Synthetic}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{S}(F_i^{\text{R}}, F_i^{\text{S}})$$

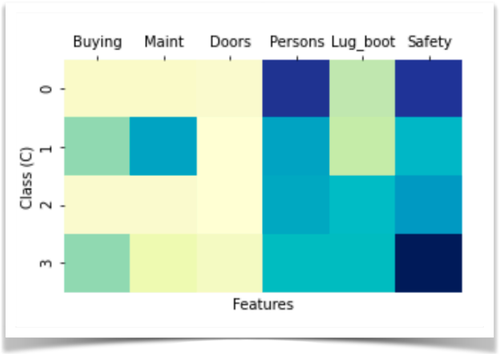
Jensen-Shannon Divergence (JSD)

Wasserstein Distance (WD)

Interpretation

- Local interpretability:
  - Understanding of why a synthetic data point belongs to the generated label (**at any time during the training**)
  - Or, given a generated synthetic data instance, the probability of each possible label is present
- Global interpretability:
  - How different features impact the synthetic label generation:
    - E.g., which feature has the largest impact on the synthetic label





The need for a Framework

- Classification Frameworks:
  - **Weka**:
    - buildClassifier()
    - distributionForInstance()
  - **Scikit-Learn**:
    - built()
    - evaluate()
  - **Many Others**

- Data Generation Frameworks:
  - **Katabatic**:
    - buildGenerator()
    - generateData()
    - evaluateModel()

Application I: Record Generation

- Record Generation:
  - Medical health records
  - Airline passenger records
- Why and how to use the generated records?
  - Address scarcity in data sources
  - Can we use it in supervised settings?
- Anomaly Detection
- Interactive Learning
- Reinforcement Learning

Data-set Version Info	ICU Patients (Rows)	ICD9 Diagnoses (Columns)
MIMIC-III 1.4	46,520	1,071

	ICD9 - 1	ICD9 - 2	ICD9 - 3	...	ICD9 - 1071
Patient - 1	0	0	0	...	0
Patient - 2	1	0	1	...	0
Patient - 3	0	0	0	...	1
. . .	. . .	. . .	. . .	. ... .	. . .
Patient - 46,520	0	1	1	...	1

```
Patient 18 has the following icd9 diagnoses:

D_519 -- "Tracheostomy complication, unspecified"
D_425 -- Endomyocardial fibrosis
D_584 -- Acute kidney failure with lesion of tubular necrosis
D_286 -- Congenital factor VIII disorder
D_573 -- Chronic passive congestion of liver
D_599 -- "Urinary tract infection, site not specified"
D_767 -- Subdural and cerebral hemorrhage
```

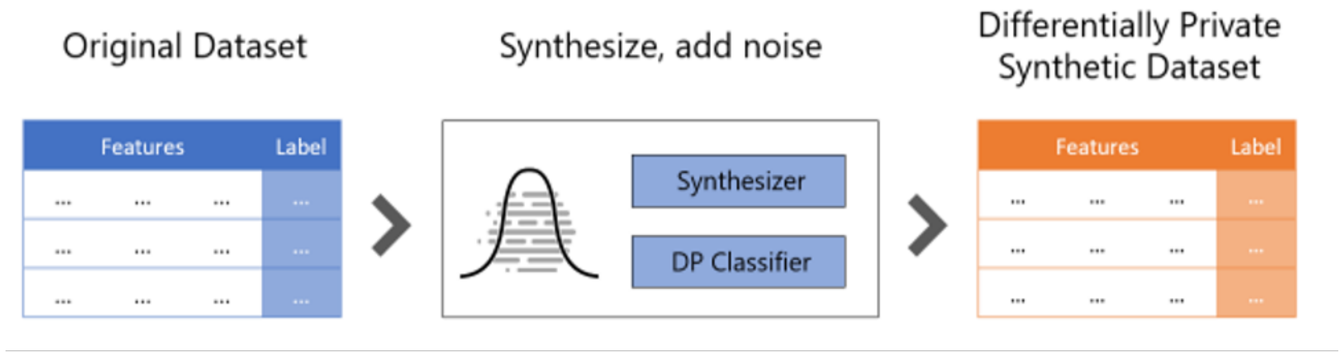
Feature	Type	Range/Card.
Country Origin	Cat.	81
Country Destination	Cat.	95
Country Office Id.	Cat.	65
Stay Saturday	Binary	{0,1}
Purchase Anticipation	Num.	[0,364]
Number Passengers	Num.	[1,9]
Stay duration days	Num.	[0,90]
Gender	Binary	{0,1}
PNR With Children	Binary	{0,1}
Age (years)	Num.	[0,99]
Nationality	Cat.	76
Business/Leisure	Binary	{0,1}

Application II: Privacy Preserving

- Differential privacy guarantees the output of the model f be insensitive to the presence or absence of any one individual in the dataset
- We use D_{data} and D'_{data}, to denote two neighbouring datasets which differ exactly in one instance

- **Definition:** *A randomised algorithm M satisfies ϵ-differential privacy, if for all neighbouring datasets D_{data} and D'_{data} and all subsets z, we have:* $P(M(D_{data}) \in z) = e^{\epsilon} P(M(D'_{data}) \in z)$

- ϵ is the privacy budget,
- Smaller value dictates:
  - A strong privacy guarantee
  - However, machine learning utility can be low



Application II: Privacy Preserving

For a given deterministic function f , DP is of- ten achieved by injecting random noise into f 's output, while the noise magnitude is determined by f 's sensitivity

Global Sensitivity

$$\Delta = \max_{\mathcal{D}_{\text{data}}, \mathcal{D}'_{\text{data}}} |f(\mathcal{D}_{\text{data}}) - f(\mathcal{D}'_{\text{data}})|_1$$

$$\mathcal{M}(\mathcal{D}_{\text{data}}) = f(\mathcal{D}_{\text{data}}) + \mathcal{N}(0, \Delta\sigma^2)$$

Gaussian Noise Mechanism

$$\mathcal{M}(\mathcal{D}_{\text{data}}) = f(\mathcal{D}_{\text{data}}) + \text{Lap}(0, \frac{\Delta}{\epsilon})$$

Laplace Mechanism

Privacy Preserving [33]

C and σ are different for various Layers in the network

- Differential Privacy SGD Algorithm
  - One might attempt to protect the privacy of training data by working only on the final parameters that result from the training process, treating this process as a black box
    - One may not have a useful, tight characterization of the dependence of these parameters on the training data; adding overly conservative noise to the parameters, where the noise is selected according to the worst-case analysis, would destroy the utility of the learned model
  - One must control the influence of the training data during the training process, specifically during the SGD computation

Algorithm 1 Differentially private SGD (Outline)

Input: Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate η_t, noise scale σ, group size L, gradient norm bound C.

Initialize θ_0 randomly

for $t \in [T]$ **do**

Take a random sample L_t with sampling probability L/N

Compute gradient

For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$

Clip gradient

$\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i) / \max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$

Add noise

$\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\left(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$

Descent

$\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$

Output θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

DP-GAN [34]

- Is it straightforward to use DP SGD to use with-in GAN framework?
- Not Straightforward
- Solution: DP-GAN
- Moment Accountant
  - Let us talk about moment accountant

Algorithm 1: Basic dp-GAN

Input: n - number of samples; λ - coefficient of gradient penalty; n_{critic} - number of critic iterations per generator iteration; n_{param} - number of discriminator's parameters; m - batch size; $(\alpha, \beta_1, \beta_2)$ - Adam hyper-parameters; C - gradient clipping bound; σ - noise scale; (ϵ_0, δ_0) - total privacy budget

Output: differentially private generator G

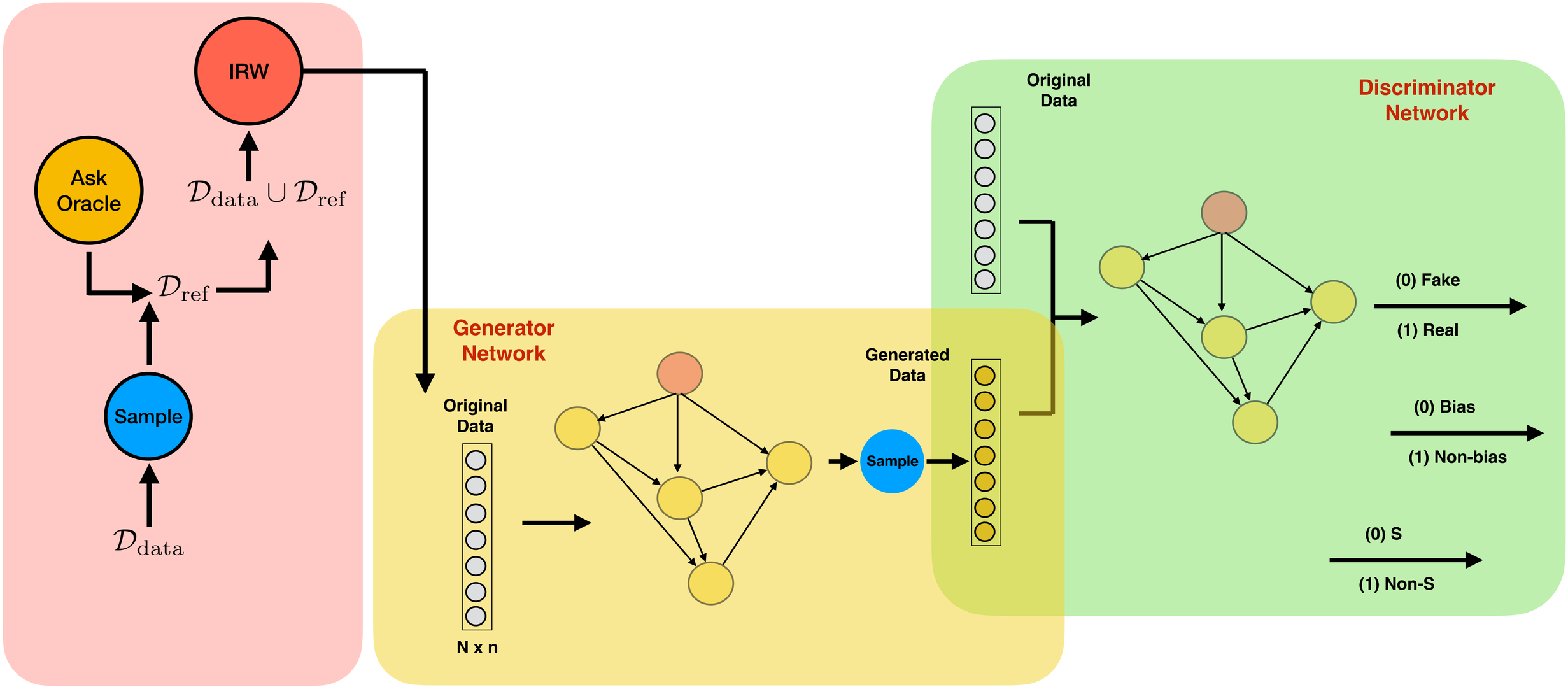
while θ *has not converge* **do**
  for $t = 1, \cdots, n_{critic}$ **do**
    for $i = 1, \cdots, m$ **do**
      sample $x \sim p_{\text{data}}, z \sim p_z, \rho \sim \mathcal{U}[0, 1]$;
      $\hat{x} \leftarrow \rho x + (1 - \rho) G(z)$;
      $\ell^{(i)} \leftarrow D(G(z)) - D(x) + \lambda \left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1 \right)^2$;
      // computing discriminator's gradients
      $g^{(i)} \leftarrow \nabla_w \ell^{(i)}$;
      // clipping and perturbation $(\xi \sim \mathcal{N}(0, (\sigma C)^2 \mathcal{I}))$
      $g^{(i)} \leftarrow g^{(i)} / \max(1, ||g^{(i)}||_2 / C) + \xi$;
    // updating privacy accountant
    update $\mathcal{A}$ with $(\sigma, m, n_{\text{param}})$;
    // updating discriminator
    $w \leftarrow \text{Adam}\left(\frac{1}{m} \sum_{i=1}^{m} g^{(i)}, w, \alpha, \beta_1, \beta_2 \right)$;
  sample $\{z^{(i)}\}_{i=1}^{m} \sim p_z$;
  // updating generator
  $\theta \leftarrow \text{Adam}\left(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D(G(z^{(i)})), \theta, \alpha, \beta_1, \beta_2 \right)$;
  // computing cumulative privacy loss
  $\delta \leftarrow$ query $\mathcal{A}$ with ϵ_0;
  if $\delta > \delta_0$ **then** break;
return G

DP Enforced only in Discriminator

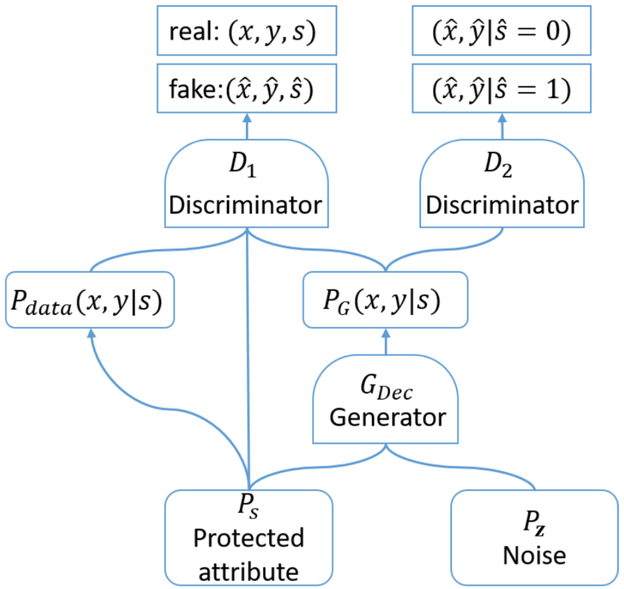
Application III: Fairness Enforcement

- Bias - Latent/Explicit
  - **Latent Bias:**
    - Weak Supervision — un-bias/reference dataset D_{ref} is built, and is used in conjunction with bias dataset D_{bias}
  - **Explicit bias:**
    - Enforce fairness constraints — e.g., $\frac{P(Y|S=0)}{P(Y|S=1)}$, where Y is the dependent variable, and S denotes some sensitive feature in the data
- How is un-bias/reference dataset D_{ref} is built? De-bias Sampling
- Train on a model on:
  - D_{ref} U D_{bias}
    - Combining D_{ref} and D_{bias} is a reasonable strategy, but what is needed is a way of systematic weighting that selects samples from D_{ref} U D_{bias} such that fairness is not compromised
    - Image domain - Conditional Modelling, Importance Re-weighting

Fairness Enforcement



FairGAN [25]



Algorithm 1 Minibatch stochastic gradient descent training of FairGAN.

for number of pre-training iterations **do**
  Sample a batch of m examples $(\mathbf{x}, y, s) \sim P_{\text{data}}(\mathbf{x}, y, s)$
  Update Autoencoder by descending its stochastic gradient:

$$\nabla_{\theta_{ae}} \frac{1}{m} \sum_{i=1}^{m} ||\mathbf{x}' - \mathbf{x}||_2^2$$

end for
for number of training iterations **do**
  Sample a batch of m examples $(\mathbf{x}, y, s) \sim P_{\text{data}}(\mathbf{x}, y, s)$
  Sample a batch of m examples $(\hat{\mathbf{x}}, \hat{y}, \hat{s}) \sim P_G(\mathbf{x}, y, s)$ from generator $G_{Dec}(\mathbf{z}, s)$ by first drawing $s \sim P_G(s)$ and noise samples $\mathbf{z} \sim P_{\mathbf{z}}(\mathbf{z})$
  Update D_1 by ascending its stochastic gradient:

$$\nabla_{\theta_{d_1}} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_1(\mathbf{x}, y, s) + \log(1 - D_1(\hat{\mathbf{x}}, \hat{y}, \hat{s})) \right]$$

  Update G_{Dec} by descending along its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D_1(\hat{\mathbf{x}}, \hat{y}, \hat{s}))$$

  Sample a batch of m examples $(\hat{\mathbf{x}}, \hat{y}|\hat{s} = 1) \sim P_G(\mathbf{x}, y|s = 1)$ and sample another batch of m examples $(\hat{\mathbf{x}}, \hat{y}|\hat{s} = 0) \sim P_G(\mathbf{x}, y|s = 0)$
  Update D_2 by ascending its stochastic gradient:

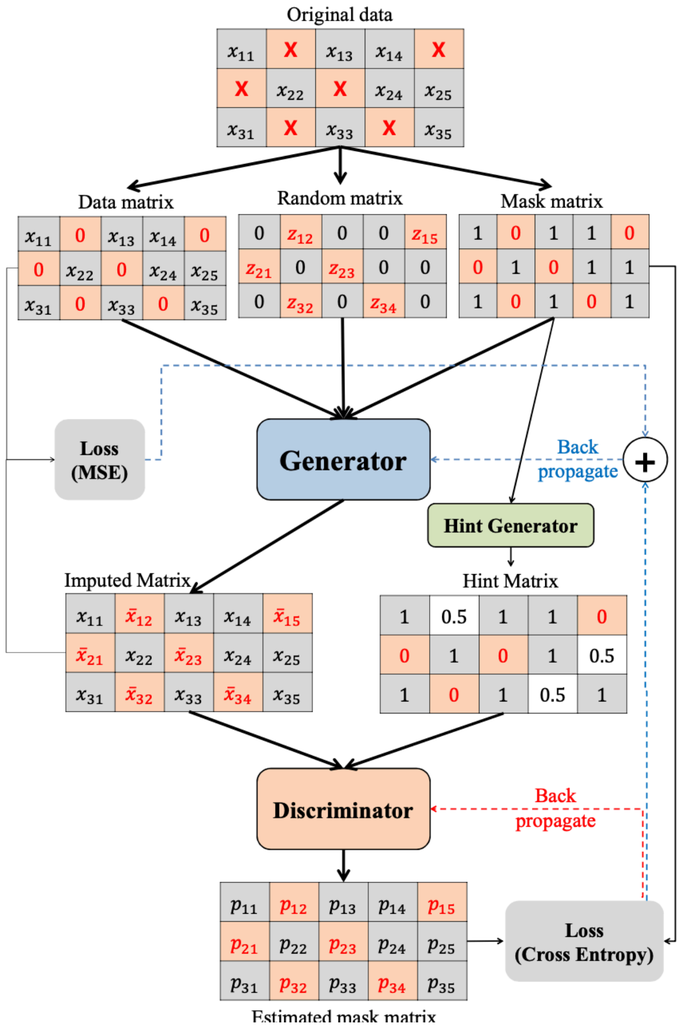
$$\nabla_{\theta_{d_2}} \frac{1}{2m} \sum_{i=1}^{2m} \left[\log D_2(\hat{\mathbf{x}}, \hat{y}) + \log(1 - D_2(\hat{\mathbf{x}}, \hat{y})) \right]$$

  Update G_{Dec} by descending along its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{2m} \sum_{i=1}^{2m} \left[\log D_2(\hat{\mathbf{x}}, \hat{y}) + \log(1 - D_2(\hat{\mathbf{x}}, \hat{y})) \right]$$

end for

Application IV: Missing Value Imputation [35]



Algorithm 1 Pseudo-code of GAIN

while training loss has not converged **do**

(1) Discriminator optimization

Draw k_D samples from the dataset $\{(\tilde{\mathbf{x}}(j), \mathbf{m}(j))\}_{j=1}^{k_D}$

Draw k_D i.i.d. samples, $\{\mathbf{z}(j)\}_{j=1}^{k_D}$, of $\mathbf{Z}$

Draw k_D i.i.d. samples, $\{\mathbf{b}(j)\}_{j=1}^{k_D}$, of $\mathbf{B}$

for $j = 1, ..., k_D$ **do**

$\bar{\mathbf{x}}(j) \leftarrow G(\tilde{\mathbf{x}}(j), \mathbf{m}(j), \mathbf{z}(j))$

$\hat{\mathbf{x}}(j) \leftarrow \mathbf{m}(j) \odot \tilde{\mathbf{x}}(j) + (\mathbf{1} - \mathbf{m}(j)) \odot \bar{\mathbf{x}}(j)$

$\mathbf{h}(j) = \mathbf{b}(j) \odot \mathbf{m}(j) + 0.5(\mathbf{1} - \mathbf{b}(j))$

end for

Update D using stochastic gradient descent (SGD)

$$\nabla_D - \sum_{j=1}^{k_D} \mathcal{L}_D(\mathbf{m}(j), D(\hat{\mathbf{x}}(j), \mathbf{h}(j)), \mathbf{b}(j))$$

(2) Generator optimization

Draw k_G samples from the dataset $\{(\tilde{\mathbf{x}}(j), \mathbf{m}(j))\}_{j=1}^{k_G}$

Draw k_G i.i.d. samples, $\{\mathbf{z}(j)\}_{j=1}^{k_G}$ of $\mathbf{Z}$

Draw k_G i.i.d. samples, $\{\mathbf{b}(j)\}_{j=1}$ of $\mathbf{B}$

for $j = 1, ..., k_G$ **do**

$\mathbf{h}(j) = \mathbf{b}(j) \odot \mathbf{m}(j) + 0.5(\mathbf{1} - \mathbf{b}(j))$

end for

Update G using SGD (for fixed D)

$$\nabla_G \sum_{j=1}^{k_G} \mathcal{L}_G(\mathbf{m}(j), \hat{\mathbf{m}}(j), \mathbf{b}(j)) + \alpha \mathcal{L}_M(\mathbf{x}(j), \tilde{\mathbf{x}}(j))$$

end while

Application IV: Missing Value Imputation [35]

Table 2. Imputation performance in terms of RMSE (Average ± Std of RMSE)

Algorithm	Breast	Spam	Letter	Credit	News
GAIN	**.0546 ± .0006**	**.0513± .0016**	**.1198± .0005**	**.1858 ± .0010**	**.1441 ± .0007**
MICE	.0646 ± .0028	.0699 ± .0010	.1537 ± .0006	.2585 ± .0011	.1763 ± .0007
MissForest	.0608 ± .0013	.0553 ± .0013	.1605 ± .0004	.1976 ± .0015	.1623 ± 0.012
Matrix	.0946 ± .0020	.0542 ± .0006	.1442 ± .0006	.2602 ± .0073	.2282 ± .0005
Auto-encoder	.0697 ± .0018	.0670 ± .0030	.1351 ± .0009	.2388 ± .0005	.1667 ± .0014
EM	.0634 ± .0021	.0712 ± .0012	.1563 ± .0012	.2604 ± .0015	.1912 ± .0011

Table 1. Source of gains in GAIN algorithm (Mean ± Std of RMSE (Gain (%)))

Algorithm	Breast	Spam	Letter	Credit	News
GAIN	**.0546 ± .0006**	**.0513± .0016**	**.1198± .0005**	**.1858 ± .0010**	**.1441 ± .0007**
GAIN w/o $\mathcal{L}_G$	.0701 ± .0021 **(22.1%)**	.0676 ± .0029 **(24.1%)**	.1344 ± .0012 **(10.9%)**	.2436 ± .0012 **(23.7%)**	.1612 ± .0024 **(10.6%)**
GAIN w/o $\mathcal{L}_M$	.0767 ± .0015 **(28.9%)**	.0672 ± .0036 **(23.7%)**	.1586 ± .0024 **(24.4%)**	.2533 ± .0048 **(26.7%)**	.2522 ± .0042 **(42.9%)**
GAIN w/o Hint	.0639 ± .0018 **(14.6%)**	.0582 ± .0008 **(11.9%)**	.1249 ± .0011 **(4.1%)**	.2173 ± .0052 **(14.5%)**	.1521 ± .0008 **(5.3%)**
GAIN w/o Hint & $\mathcal{L}_M$	.0782 ± .0016 **(30.1%)**	.0700 ± .0064 **(26.7%)**	.1671 ± .0052 **(28.3%)**	.2789 ± .0071 **(33.4%)**	.2527 ± .0052 **(43.0%)**

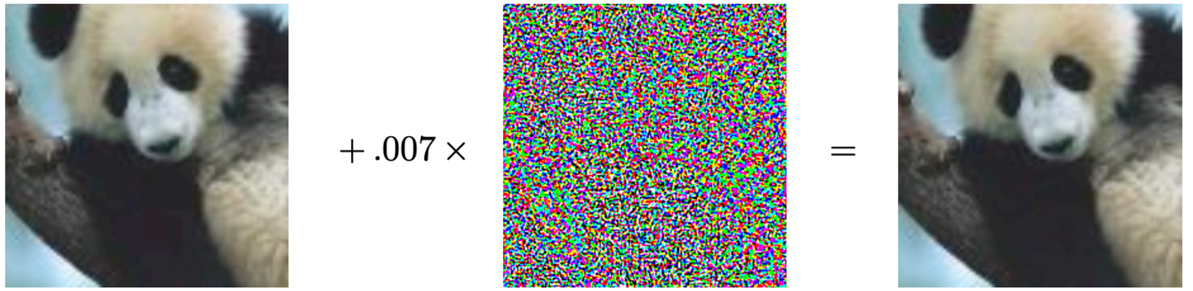
Missing Value Imputation

- **HI-VAIE**
  - Handling Incomplete Heterogeneous Data using VAEs
- **MIDA**
  - MIDA: Multiple Imputation using Denoising Autoencoders

Application V: Adversarial Generation

- Ethical Adversarial Attacks:
  - FGSM (Fast Gradient Sign Method)
  - BIM (Basic Iterative Method)
  - PGD (Projected Gradient Descent)
  - DeepFool
  - LowProFool
  - Blackbox-Tabular-Data-Attack (BTDA)
  - Universal Attack

FGSM [26]



- If θ be the parameters of the model, x is the input and y is its associate label, and J(θ, x, y) be the cost used to train the network
- We can linearise the cost function around the current value of θ, obtaining an optimal max-norm constrained perturbation of:

$$\eta = \epsilon \, \text{sign}(\nabla_{\mathbf{x}}(\boldsymbol{\theta}, \mathbf{x}, y))$$

- Required gradient can be computed efficiently using back-propagation

$$\mathbf{x}^{\text{adv}} = \mathbf{x} + \eta$$

BIM and PGD [27,28]

- **BIM**
  - Apply FGSM multiple times with small step size
    - α is set to 1
    - Number of iterations: min(ϵ+4, 1.25ϵ)
    - $\text{Clip}_{X,\epsilon}(A)$ is an element-wise clipping of A, where each element A_{ij}, is clipped to the range:
      - $[X_{ij} - \epsilon, X_{ij} + \epsilon]$
- **PGD**
  - Similar to BIM but initialize randomly

BIM

$$\mathbf{x}_0^{\text{adv}} = \mathbf{x}$$

$$\mathbf{x}_{N+1}^{\text{adv}} = \text{Clip}_{X,\epsilon}\{\mathbf{x}_N^{\text{adv}} + \alpha \text{sign}(\nabla_{\mathbf{x}}(\boldsymbol{\theta}, \mathbf{x}_N^{\text{adv}}, y))\}$$

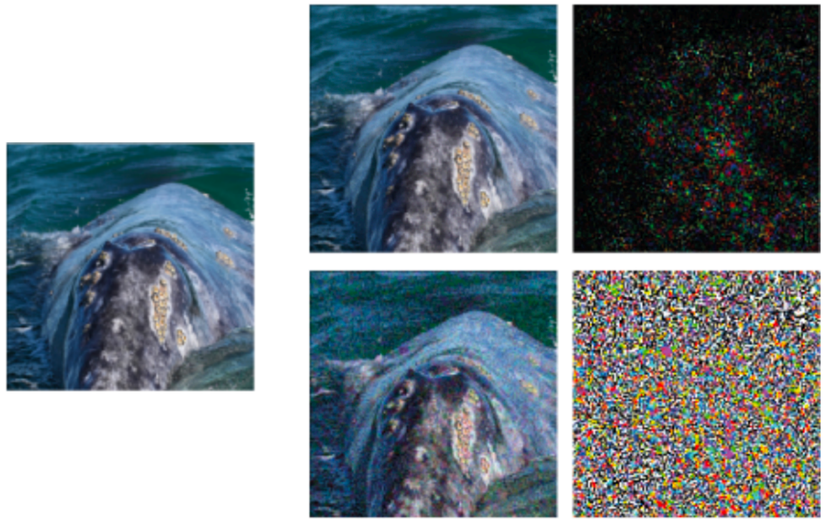
PGD

$$\mathbf{x}_0^{\text{adv}} = \mathbf{x}^{\text{Random}}$$

$$\mathbf{x}_{N+1}^{\text{adv}} = \text{Clip}_{X,\epsilon}\{\mathbf{x}_N^{\text{adv}} + \alpha \text{sign}(\nabla_{\mathbf{x}}(\boldsymbol{\theta}, \mathbf{x}_N^{\text{adv}}, y))\}$$

DeepFool [29]

- DeepFool for Binary Classifier:
  - Minimal perturbation r*(x) for the classifier f to misclassify the input x is the orthogonal projection to the hyperplane of the binary classifier



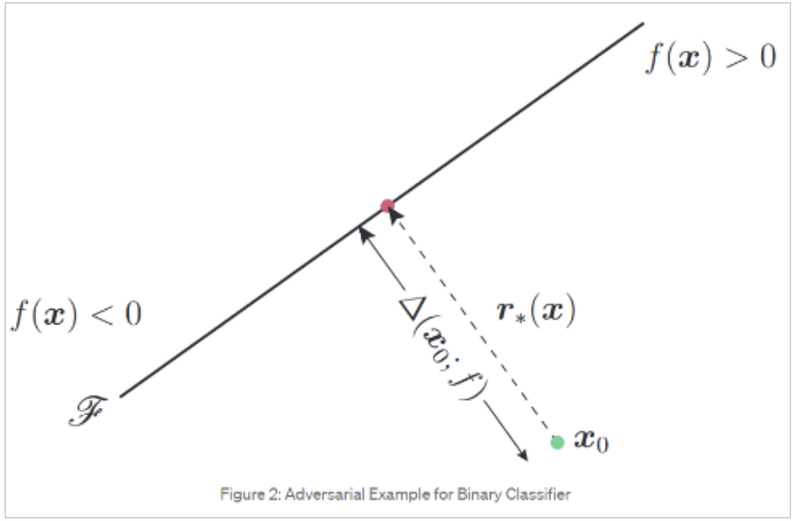


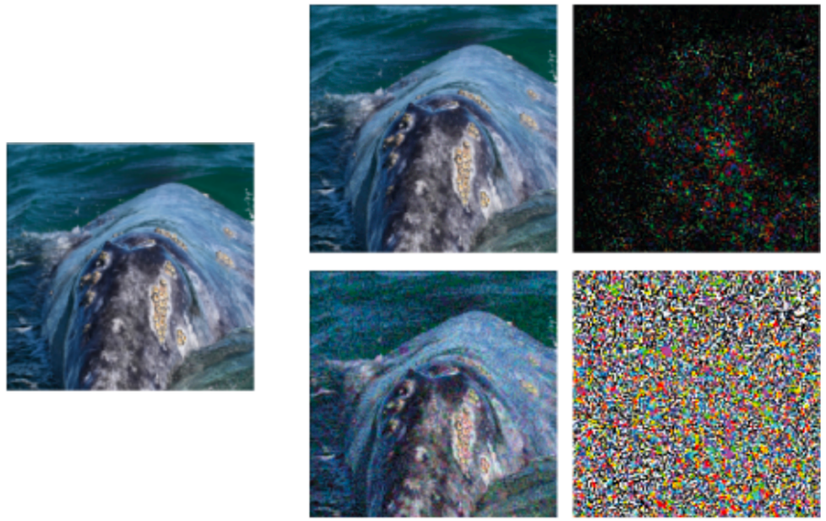
Figure 2: Adversarial Example for Binary Classifier

$$
\begin{aligned}
r^*(\mathbf{x}^0) &= \arg\min \|\mathbf{r}\|_2 \\
&\quad \text{subject to } \operatorname{sign}(f(\mathbf{x}_0 + \mathbf{r})) \neq \operatorname{sign}(f(\mathbf{x}_0)) \\
&= -f(\mathbf{x}_0)\frac{\mathbf{w}}{\|\mathbf{w}\|_2^2}
\end{aligned}
$$

DeepFool [29]

Algorithm 1 DeepFool for binary classifiers

1: **input:** Image $\boldsymbol{x}$, classifier f.
2: **output:** Perturbation $\hat{\boldsymbol{r}}$.
3: Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
4: **while** $\text{sign}(f(\boldsymbol{x}_i)) = \text{sign}(f(\boldsymbol{x}_0))$ **do**
5: $\quad \boldsymbol{r}_i \leftarrow -\frac{f(\boldsymbol{x}_i)}{\|\nabla f(\boldsymbol{x}_i)\|_2^2} \nabla f(\boldsymbol{x}_i)$,
6: $\quad \boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$,
7: $\quad i \leftarrow i + 1$.
8: **end while**
9: **return** $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$.



$$
\begin{aligned}
r^*(\mathbf{x}^0) &= \arg\min \|\mathbf{r}\|_2 \\
&\quad \text{subject to } \text{sign}(f(\mathbf{x}_0 + \mathbf{r})) \neq \text{sign}(f(\mathbf{x}_0)) \\
&= -f(\mathbf{x}_0) \frac{\mathbf{w}}{\|\mathbf{w}\|_2^2}
\end{aligned}
$$

$$
\arg\min_{\mathbf{r}_i} \|\mathbf{r}_i\|_2, \text{subject to } f(\mathbf{x}_i) + \nabla f(\mathbf{x}_i)^T \mathbf{r}_i = 0
$$

LowProFool [30]

- Apply FGSM multiple times with small step size:

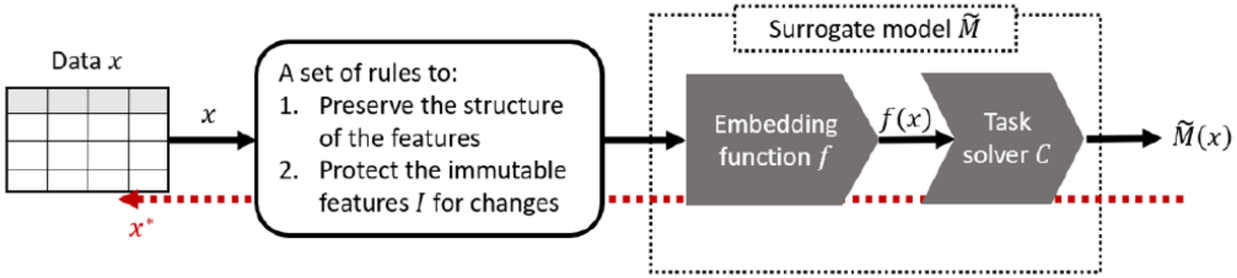
Algorithm 1 LowProFool

Input: Classifier f, sample $\boldsymbol{x}$, target label t, loss function $\mathcal{L}$, trade-off factor λ, feature importance $\boldsymbol{v}$, maximum number of iterations N, scaling factor α

Output: Adversarial example $\boldsymbol{x}'$

$\boldsymbol{r} \leftarrow [0, 0, \ldots, 0]$
$\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$
for i in $0 \ldots N-1$ **do**
  $\boldsymbol{r}_i \leftarrow -\nabla_{\boldsymbol{r}}(\mathcal{L}(\boldsymbol{x}_i, t) + \lambda ||\boldsymbol{v} \odot \boldsymbol{r}||_p)$
  $\boldsymbol{r} \leftarrow \boldsymbol{r} + \alpha \boldsymbol{r}_i$
  $\boldsymbol{x}_{i+1} \leftarrow \text{clip}(\boldsymbol{x} + \boldsymbol{r})$
end for
$\boldsymbol{x}' \leftarrow \arg\min_{\boldsymbol{x}_i} d_{\boldsymbol{v}}(\boldsymbol{x}_i) \quad \forall i \in [0 \ldots N-1] \quad \text{s.t.} \quad f(\boldsymbol{x}_i) \neq f(\boldsymbol{x}_0)$
return $\boldsymbol{x}'$

Blackbox Attack and Universal Attack [31,32]



Algorithm 1: Crafting an adversarial example. $\widetilde{M}$ is the surrogate model, x is the candidate benign sample, y its label, $\mathcal{L}$ is the original task's loss function, $\mathcal{L}^*$ is the adversarial objective function composed of $\mathcal{L}_{adv}$ and $\mathcal{L}_{spatial}$, I is the set of immutable features, and λ is the maximum distortion (ℓ_0-wise).

Input: $\widetilde{M}$, x, y, $\mathcal{L}$, $\mathcal{L}^*$, I, λ

$x^* \leftarrow x$
$S \leftarrow \emptyset$
while $\widetilde{M}(x^*) = y$ *and* $|S| < \lambda$ **do**
  $G \leftarrow \nabla_{x^*} \mathcal{L}(\widetilde{M}(x^*), y)$
  $S \leftarrow S \cup \{\arg\max_{i \notin S \cup I} G_i\}$
  $\alpha \leftarrow \textsc{ComputeStep}(x^*, \mathcal{L}^*)$
  forall $i \in S$ **do**
    $x_i^* \leftarrow x_i^* + \alpha_i$
  $x^* \leftarrow \textsc{Project}(x^*)$
return x^*

Algorithm 1 Computation of universal perturbations.

input: Data points X, classifier $\hat{k}$, desired ℓ_p norm of the perturbation ξ, desired accuracy on perturbed samples δ.
output: Universal perturbation vector v.
Initialize $v \leftarrow 0$.
while $\text{Err}(X_v) \leq 1 - \delta$ **do**
  for each datapoint $x_i \in X$ **do**
    if $\hat{k}(x_i + v) = \hat{k}(x_i)$ **then**
      Compute the *minimal* perturbation that sends $x_i + v$ to the decision boundary:

$$\Delta v_i \leftarrow \arg\min_r \|r\|_2 \text{ s.t. } \hat{k}(x_i + v + r) \neq \hat{k}(x_i).$$

      Update the perturbation:

$$v \leftarrow \mathcal{P}_{p,\xi}(v + \Delta v_i).$$

    end if
  end for
end while

Adversarial Attacks on Tabular Data [36]

Algorithm 1: Algorithm D2A3 and D2A3N Training

Input : $\mathcal{S}_{\text{data}} = [(\mathbf{x}^1, \mathbf{y}^1), ..., (\mathbf{x}^m, \mathbf{y}^m)]$, Discretization type $- \mathcal{C}$

Initial parameter $\boldsymbol{\theta}_s$ of the model $f(\cdot)$
Run discretization method $\mathcal{C}$ to obtain cut-points for each feature with $\Phi(\cdot)$
for iteration $q \subset Q$ in training model $f(\cdot)$ **do**
  for sample $\mathbf{x}^t$ in batch $\mathcal{X} \subset \mathcal{S}_{\text{data}}$ and $\mathbf{y}^t$ **do**
    if *D2A3* **then**
      Discretize with one-hot encoding $\Phi(\mathbf{x}^t)$
      Train $f(\cdot)$ with $\Phi(\mathbf{x}^t)$ and $\mathbf{y}^t$ via gradient descent to minimize:
        $\mathcal{L}^{\boldsymbol{\theta}_s, \nabla_s}(f(\Phi(\mathbf{x}^t)), \mathbf{y}^t)$; `// Training Loss`
      Obtain adversarial sample $\tilde{\mathbf{x}}^t \in \mathcal{S}_{\text{adv}}$; `// Adversarial Training`
      Discretize with one-hot encoding $\Phi(\tilde{\mathbf{x}}^t)$
      Train $f(\cdot)$ with $\Phi(\tilde{\mathbf{x}}^t)$ via gradient descent to minimize:
        $\mathcal{L}^{\boldsymbol{\theta}_s, \nabla_s}(f(\Phi(\tilde{\mathbf{x}}^t)), \mathbf{y}^t)$
    else
      Train $f(\cdot)$ with $\mathbf{x}^t$ and $\mathbf{y}^t$ via gradient descent to minimize:
        $\mathcal{L}^{\boldsymbol{\theta}_s, \nabla_s}(f(\mathbf{x}^t), \mathbf{y}^t)$
      Obtain adversarial sample $\tilde{\mathbf{x}}^t \in \mathcal{S}_{\text{adv}}$; `// Adversarial Training`
      Obtain data transformation: $\mathcal{M}(\Phi(\tilde{\mathbf{x}}^t))$
      Train $f(\cdot)$ with $\mathcal{M}(\Phi(\tilde{\mathbf{x}}^t))$ via gradient descent to minimize:
        $\mathcal{L}^{\boldsymbol{\theta}_s, \nabla_s}(f(\mathcal{M}(\Phi(\tilde{\mathbf{x}}^t))), \mathbf{y}^t)$; `// Adversarial Training Loss`
return $f(\cdot)$, $\Phi(\cdot)$, $\mathcal{M}(\cdot)$

Table 2: Performance Comparison for D2A3.

Models	Standard Accuracy (Avg)	Robust Accuracy (Avg)			Success Rate(SR) (Avg)		
		FGSM	DeepFool	LowProFool	FGSM	DeepFool	LowProFool
Clean	0.895	0.813	0.272	0.282	0.135	0.793	0.769
Thermometer	0.836	0.759	0.761	0.726	0.196	0.160	0.194
D2A3-EF	0.870	0.792	0.768	0.722	0.153	0.165	0.217
D2A3-EW	0.854	0.793	0.788	0.753	0.136	0.143	0.161
D2A3-MDL	**0.918**	**0.897**	**0.884**	**0.869**	**0.000**	**0.025**	**0.030**

Conclusion & Discussion

Summary

- Tabular Data Generation is extremely important in the context of Controlled Learning
- **Massive stride has been made with coming off GAN and related deep data generation models**
  - Community needs to work on the standards/evaluations
  - Simple methods should be considered and evaluated
  - There is a need for unified framework
  - At the end of the day, we need something other than the given data - be it an auxiliary data source, expert advise or else - research need to focus on integration of such sources
  - Need for investigating models that has little to no reliance on the data

Future Collaboration

Nayyar Zaidi

Gang Li

http://www.nayyarzaidi.com

Google: {Nayyar, Zaidi, Deakin}

Email: nayyar.zaidi@deakin.edu.au

- Looking for a Phd in this area
- Building on research collaboration
- Interested in writing competitive research grant applications like ARC Linkage
- Industry willing to fund a project

References

[1] Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. arXiv preprint arXiv:1701.00160.

[2] Hastie, T., Tibshirani, R., & Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction.* 2nd ed. New York, Springer.

[3] Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. Machine learning, 29(2), 131-163.

[4] Chen, H., & Murray, A. F. (2003). Continuous restricted Boltzmann machine with an implementable training algorithm. IEE Proceedings-Vision, Image and Signal Processing, 150(3), 153-158.

[5] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

[6] Hugo Larochelle and Ian Murray. The Neural Autoregressive Distribution Estimator. In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS 2011), volume 15, pages 29–37, Ft. Lauderdale, USA, 2011. JMLR W&CP.

[7] Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. arXiv preprint arXiv:1701.00160.

[8] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating multi-label discrete electronic health records using generative adversarial networks. arXiv preprint arXiv:1703.06490, 2017.

[9] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. In International Conference on Very Large Data Bases, 2018.

[10] Xu, L., Skoularidou, M., Cuesta-Infante, A., & Veeramachaneni, K. (2019). Modeling tabular data using conditional gan. Advances in Neural Information Processing Systems, 32.

[11] Avino, L. and Ruffini, M. and Gavalda, R. Generating Synthetic but Plausible Healthcare Records Datasets, MLMH 2018, London, UK

[11a] Ruffini, M. and Gavalda, R. and Simon, E. Clustering Patients with Tensor Decomposition, Proceedings of Machine Learning for Healthcare 2017, JMLR W&C Track Volume 68

[12] Lian, J. and Zhou, X. and Zhang, F. and Chen, Z. and Xie, X. and Sun, G. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems, KDD '18: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining

[13] Pfister, T. and Arik, S. TabNet: Attentive Interpretable Tabular Learning, https://arxiv.org/abs/1908.07442

[14] Alejandro Mottini and Alix Lhritier and Rodrigo Acuna-Agost, Airline Passenger Name Record Generation using Generative Adversarial Networks, Arxiv, abs/1807.06657 (2018)

[15] Zhao, J., Kim, Y., Zhang, K., Rush, A. & LeCun, Y.. (2018). Adversarially Regularized Autoencoders. Proceedings of the 35th International Conference on Machir in Proceedings of Machine Learning Research 80:5902-5911

[16] Alain, G., Bengio, Y., Yao, L., Yosinski, J., Thibodeau-Laufer, E., Zhang, S., & Vincent, P. (2016). GSNs: generative stochastic networks. Information and Inference: A the IMA, 5(2), 210-249.

[17] Wang. R and Fu, B. and Fu, G and Wang. M, Deep and Cross Network for Ad Click Prediction, KDD (2017)

[18] Van Den Oord, A., & Vinyals, O. (2017). Neural discrete representation learning. Advances in neural information processing systems, 30.

[19] Uria, B., Côté, M. A., Gregor, K., Murray, I., & Larochelle, H. (2016). Neural autoregressive distribution estimation. The Journal of Machine Learning Research, 17(1), 7184-7220.

[20] Zaidi, N. and Webb, G. and Carman, M. and Petitjean, F. and Buntine, W. and Hynes, M. and De Sterck, H. Efficient Parameter Learning of Bayesian Network Classifiers, Machine Learning, Volume 106, Issue 9-10, pp. 1289-1329 (2017)

[21] Zaidi, N. and Carman, M. and Cerquides, J. and Webb, G. Naive-Bayes Inspired Effective Pre-Conditioners for Speeding-up Logistic Regression ICDM2014: IEEE International Conference on Data Mining, pp. 1097-1102 (2014)

[22] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier GANs. In Proceedings of the 34th International Conference on Machine Learning, pages 2642–2651, 2017.

[23] Zaidi, N. and Carman, M. and Cerquides, J. and Webb, G. Naive-Bayes Inspired Effective Pre-Conditioners for Speeding-up Logistic Regression. ICDM2014: IEEE International Conference on Data Mining, pp. 1097-1102 (2014)

[24] Zhang, Y. and Zaidi, N. and Zhou, J. and Li, G. GANBLR: A Tabular Data Generation Model. ICDM2021: IEEE International Conference on Data Mining (2021)

[25] Li, Jie and Ren, Yongli and Deng, Ke, FairGAN: GANs-Based Fairness-Aware Learning for Recommendations with Implicit Feedback, 2022, Association for Computing Machinery

[26] Goodfellow, I. and Shlens, J. And Szegedy, C. Explaining and Harnessing Adversarial Examples, ICLR2015

[27] Keratin, A. and Goodfellow, I. and Bengio, S. Adversarial Machine Learning at Scale, ICLR 2017

[28] Madry, A. and Makelov, A. And Schmidt, L. Towards Deep Learning Models Resistant to Adversarial Attacks, ICLR 2018

[29] Dezfooli, S. and Fawazi, A. and Frossard, P. DeepFool: a simple and accurate method to fool deep neural networks, CVPR 2016

[30] Ballet, V. and Renard, X. and Aigrain, J. and Laurel, T. and Frossard. P. And Detyniecki, M. Imperceptible Adversarial Attacks on Tabular Data, NIPS 2019

[31] Mathov, Y. and Levy, E. and Katzir, Z. and Shabtai, A. and Elovici, Y. Not All Datasets Are Born Equal: On Heterogeneous Data and Adversarial Examples (https://arxiv.org/abs/2010.03180)

[32] Dezfooli, S. and Fawzi, A. Universal adversarial perturbations, CVPR 2017

[33] Abadi, M. and Chu, A. Goodfellow, I. Deep Learning with Differential Privacy, ACM CCS 2016

[34] Zhang, X. and Ji, S. Wang, T. Differentially Private Releasing via Deep Generative Model, https://arxiv.org/abs/1801.01594v2

[35] Yoon, J. and Jordan, J. Schaar, M. GAIN: Missing Data Imputation using Generative Adversarial Nets, ICML 2018

[36] Zhou, J. and Zhang, Y. and Zaidi, N. and Li, G. Discretization inspired defence Algorithm against Adversarial Attacks on Tabular Data, PAKDD2022