

Learning From Networks

—*Algorithms, Theory, & Applications*

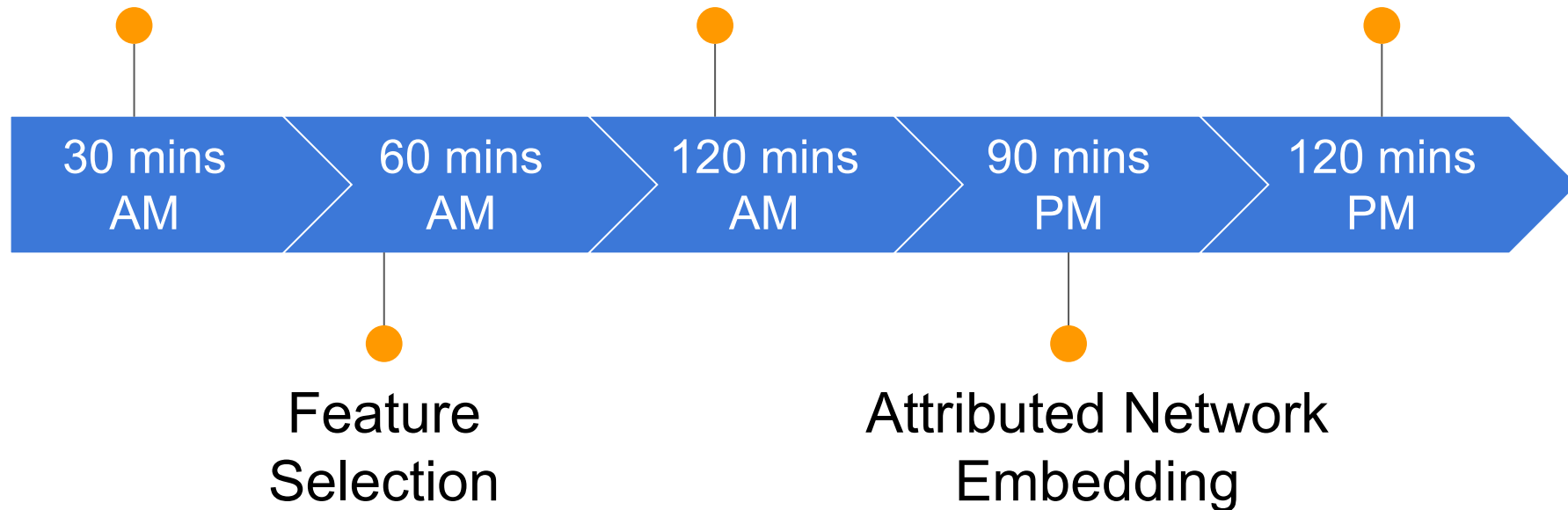
Xiao Huang, Peng Cui, Yuxiao Dong, Jundong Li, Huan Liu, Jian Pei, Le Song,
Jie Tang, Fei Wang, Hongxia Yang, Wenwu Zhu

xhuang@tamu.edu; cuip@tsinghua.edu.cn; yuxdong@microsoft.com; jundongli@asu.edu;
huan.liu@asu.edu; jpei@cs.sfu.ca; le.song@antfin.com; jietang@tsinghua.edu.cn;
few2001@med.cornell.edu; yang.yhx@alibaba-inc.com; wwzhu@tsinghua.edu.cn;

Motivations

Network
Embedding

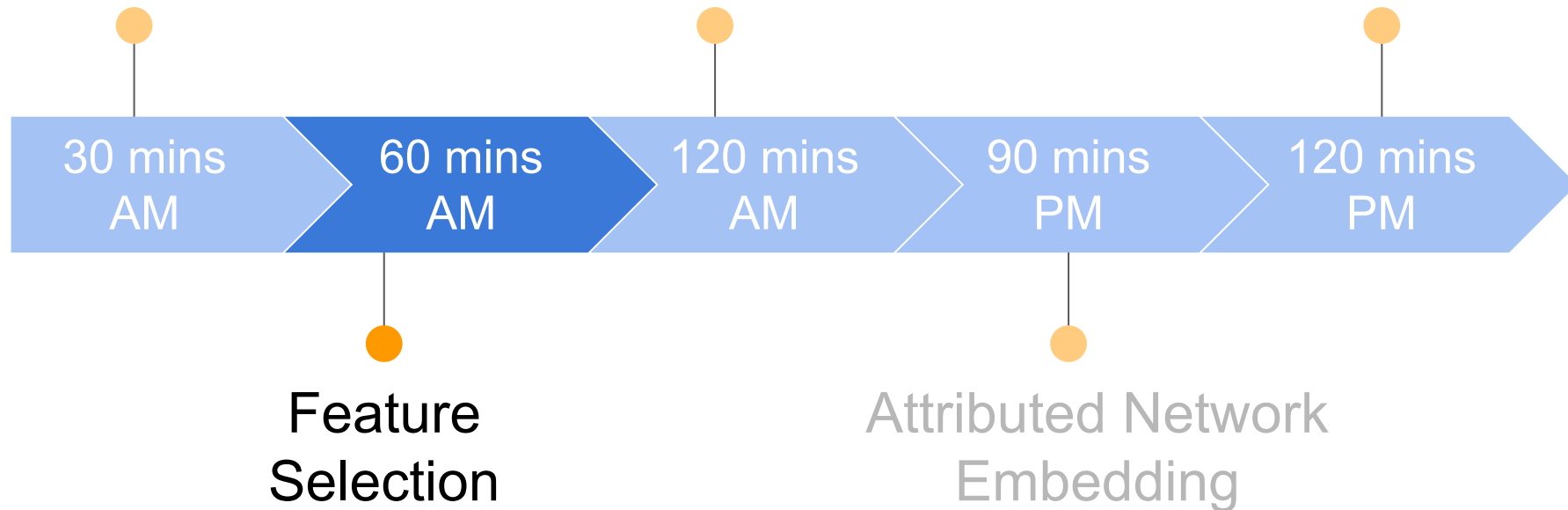
Graph Neural
Networks



Motivations

Network
Embedding

Graph Neural
Networks



Features from networks

To facilitate various graph mining algorithms, the first step is to obtain the graph features (we focus on node features)

- Scenario 1: w/o explicit node features (**plain networks**)
 - Extract hand-crafted features
 - E.g., node degree, clustering coefficient, pagerank score, ...
- Scenario 2: w/ explicit node features (**attributed networks**)
 - Leverage the observed node features
 - E.g., profiles of users in social networks, gene expression of proteins in PPI networks, research interests of scholars in collaboration networks, ...

High-dimensional features

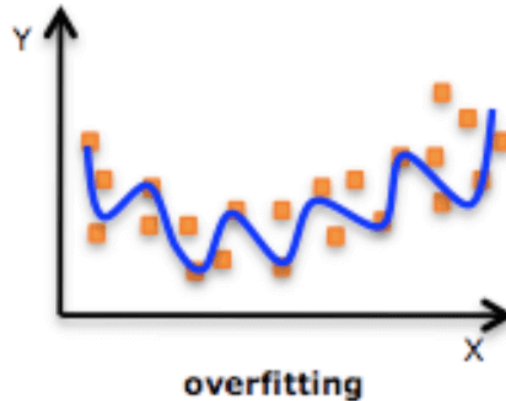
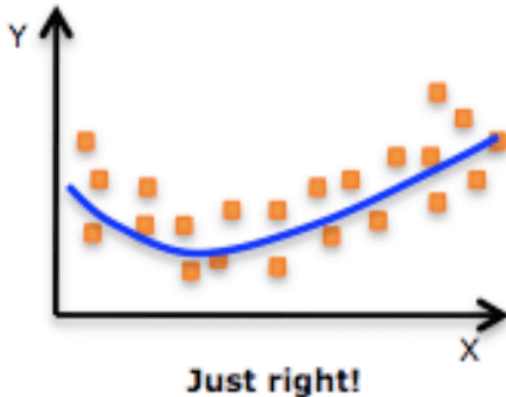
The features of nodes are often in a high-dimensional feature space

- Scenario 1: w/o explicit node features
 - Manual feature engineering generate **a large number** of features
 - Not clear what features could be useful for learning on graphs
- Scenario 2: w/ explicit node features
 - Observed features are very **high-dimensional, noisy, and sparse**
 - The intrinsic dimension of data may be small, e.g., the number of genes responsible for a certain disease

High-dimensional data is often notorious to tackle due to the *curse of dimensionality*

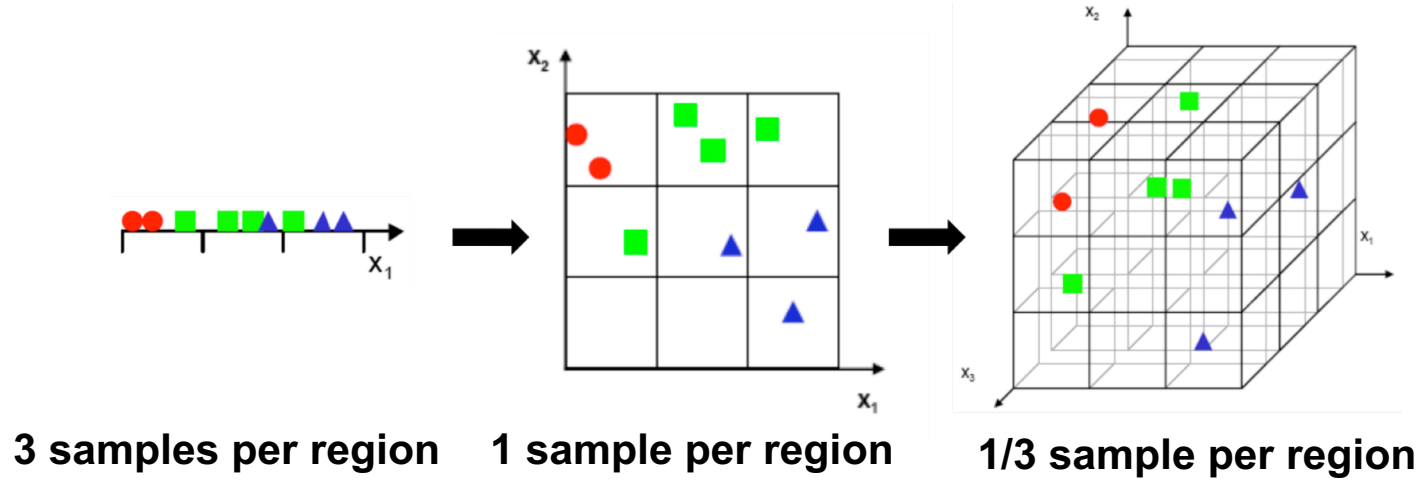
Curse of dimensionality - overfitting

- If d (the number of features) is large, the model can be overfitting as n (the number of nodes) is insufficient for parameter estimation
- For instance, to estimate the covariance matrix with d^2 parameters, we need $n > d^2$; otherwise we have less than one node (on average) per parameter



Curse of dimensionality – require more samples

- Data sparsity becomes exponentially worse as the feature dimension increases



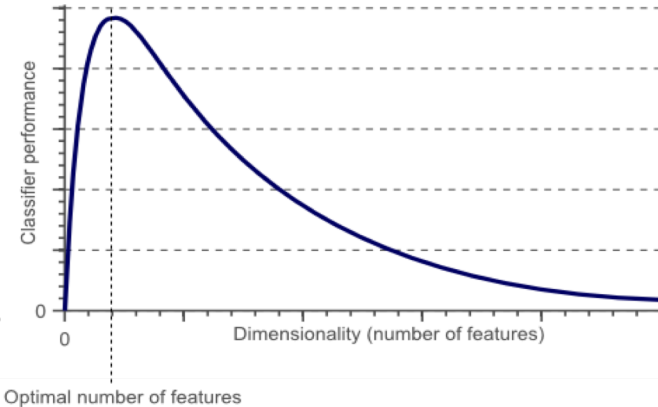
<http://nikhilbuduma.com/2015/03/10/the-curse-of-dimensionality/>

- Conventional distance metrics become ineffective

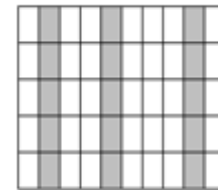
Curse of dimensionality - summary

- The curve of learning performance w.r.t. the feature dimension d
- PAC learning theory - sample complexity grows exponentially w.r.t. dimension d

$$N \geq \frac{1}{\epsilon} \left(\ln \frac{1}{\delta} + \ln |\mathbf{H}| \right) \quad |\mathbf{H}| = 2^{2^d} \quad \text{if } \mathbf{H} \text{ is the set of Boolean functions}$$



- If we reduce d , we can make data “**bigger**”
 - Before reducing d , $5/2^{10}=5/2^{10}$
 - After reducing d , $5/2^3=5/8$



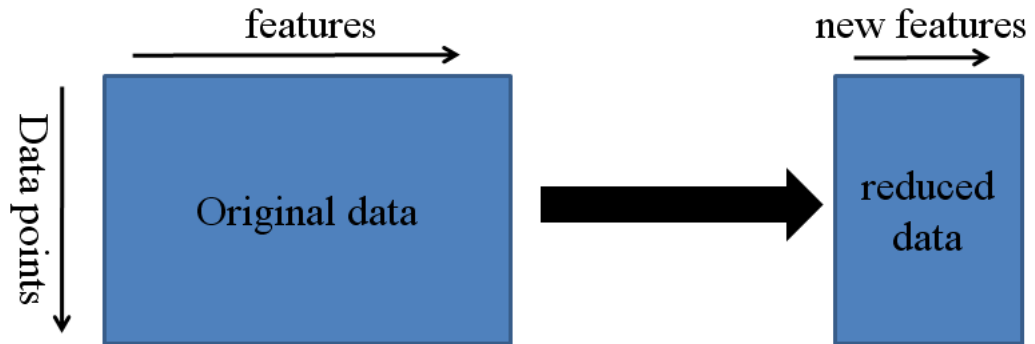
$$\mathbf{X} \in \mathbb{R}^{5 \times 10}$$



$$\mathbf{X}_{new} \in \mathbb{R}^{5 \times 3}$$

Dimensionality reduction

- Dimensionality reduction is a good way to combat the ***curse of dimensionality***
- Represent instances (nodes) with fewer features
- Dimensionality reduction algorithms
 - Feature extraction
 - Feature selection



Feature extraction

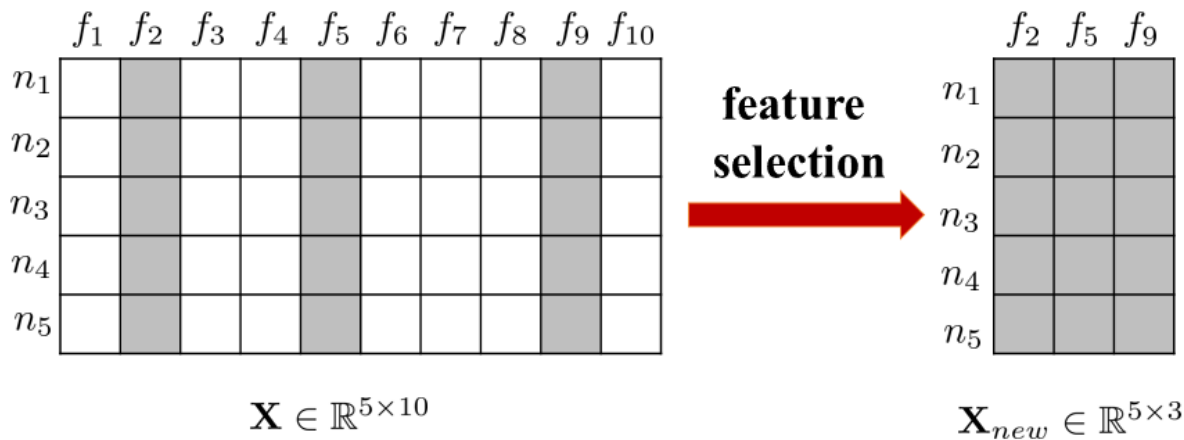
- Project the original high-dimensional data into a new feature space of low dimensionality
- Given a set of n nodes $\{x_1, x_2, \dots, x_n\}$ with d features, obtain the low-dimensional representations:

$$\mathbf{x}_i \in \mathbb{R}^d \rightarrow \mathbf{y}_i \in \mathbb{R}^p \quad (p \ll d)$$

- The new feature space is usually a linear or a nonlinear combination of the previous feature space
- The new features often do not have physical meanings

Feature selection

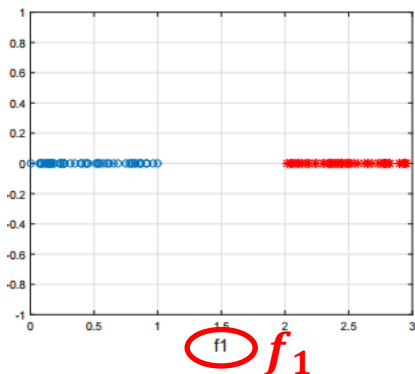
- Feature selection selects an “optimal” subset of features from the original high-dimensional feature set with a ***certain criterion***



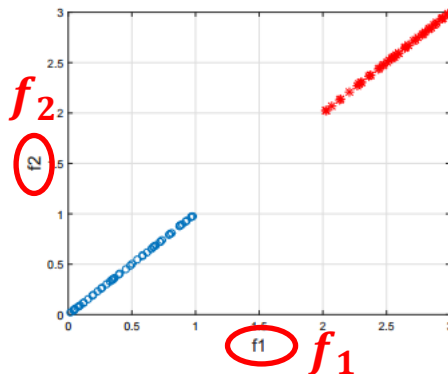
- Compared with feature extraction, feature selection gives models better ***readability*** and ***interpretability***

Relevant, redundant and irrelevant features

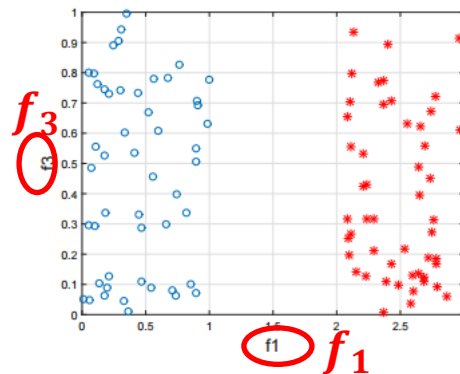
- Feature selection keeps relevant features for learning and removes redundant and irrelevant features
- For example, for a binary classification task (f_1 is relevant; f_2 is redundant given f_1 ; f_3 is irrelevant)



(a) relevant feature f_1



(b) redundant feature f_2



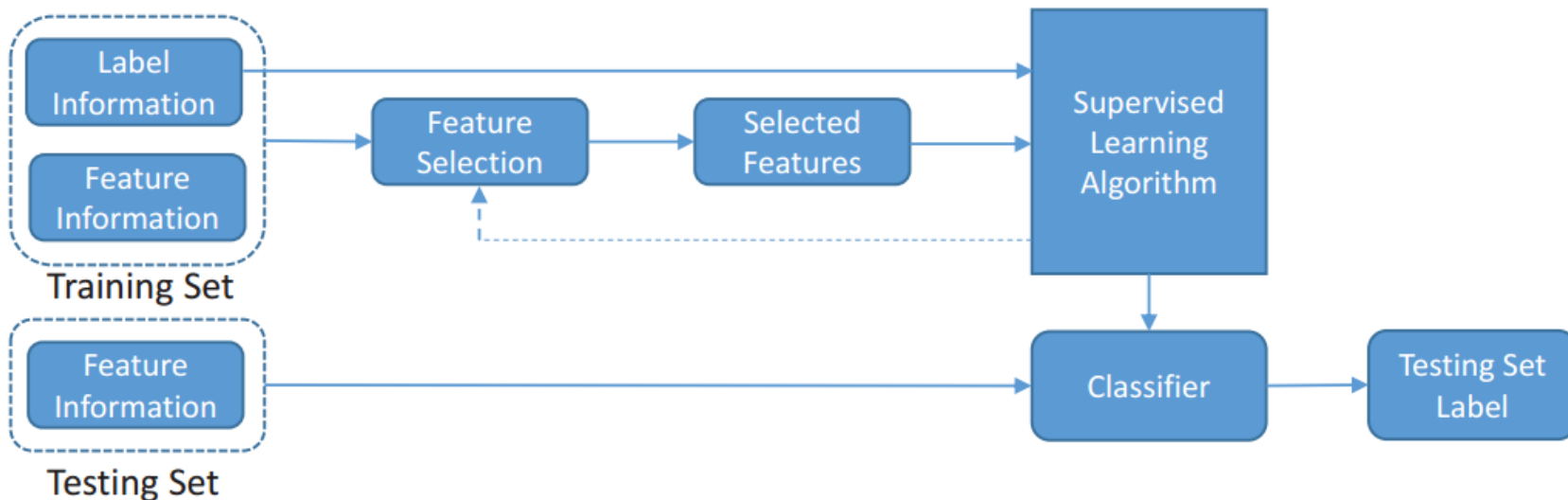
(c) irrelevant feature f_3

Categorization of feature selection algorithms

- From the label perspective:
 - Supervised
 - Unsupervised
 - Semi-Supervised
- From the selection strategy perspective:
 - Wrapper methods
 - Filter methods
 - Embedded methods

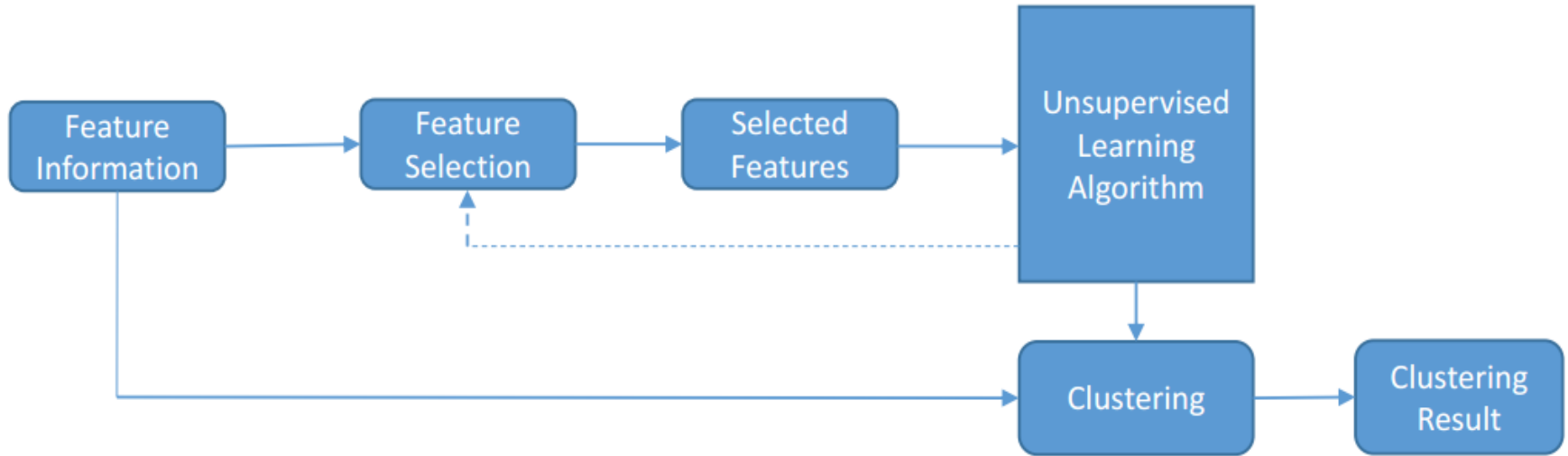
Supervised feature selection

- Supervised feature selection is often for classification or regression
- Find discriminative features that separate samples from different classes (classification) or approximate target variables (regression)



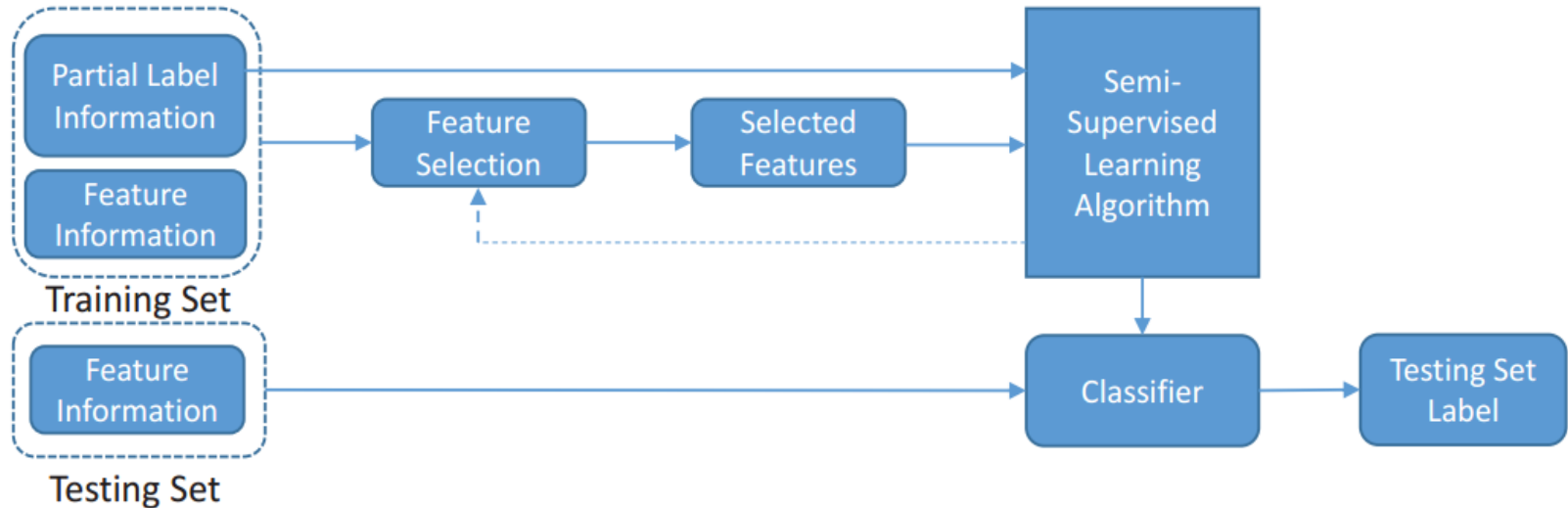
Unsupervised feature selection

- Label information is expensive to obtain
- Unsupervised methods seek alternative criteria to define the relevance of features



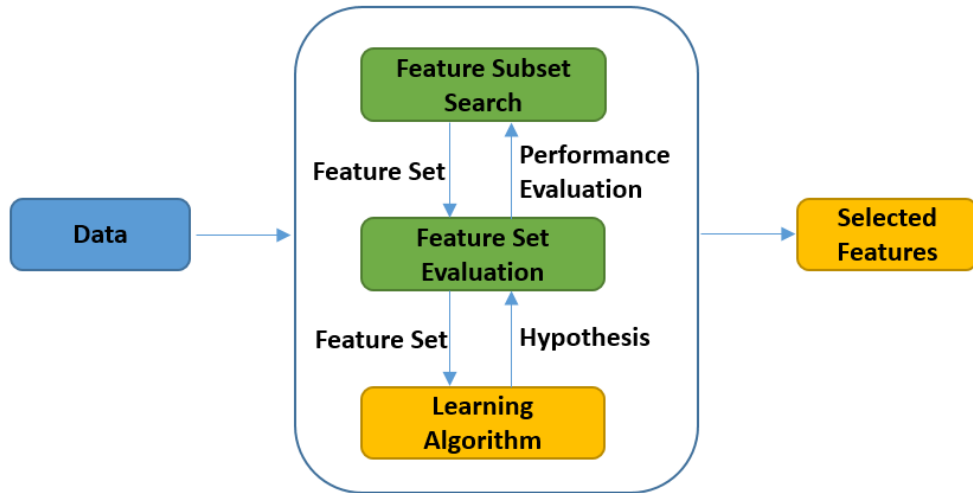
Semi-supervised feature selection

- We often have a small amount of labeled data and a large amount of unlabeled data
- Semi-supervised methods exploit both labeled and unlabeled data to find relevant features



Wrapper methods

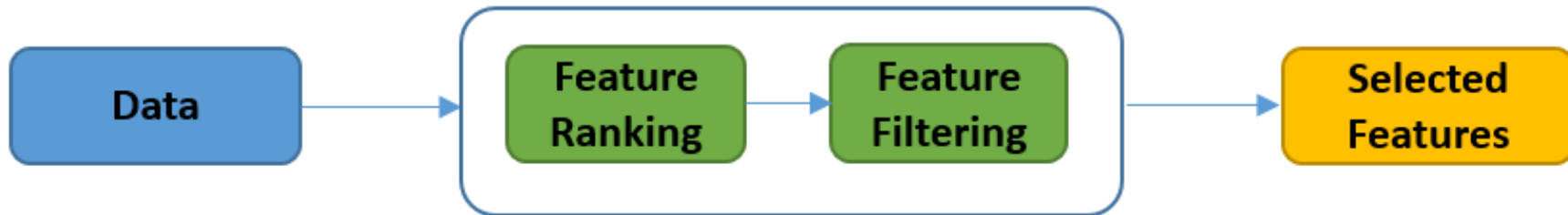
- Rely on the predictive performance of a predefined algorithm
- Repeat until some stopping criteria are satisfied
- Achieve high accuracy for a particular learning method
- Computational expensive (worst case search space is 2^d), some typical search strategies are sequential search, best-first search, ...



- Step 1: search for a subset of features
- Step 2: evaluate the selected features
- Repeat Step 1 and 2 until stopped

Filter methods

- Independent of any learning algorithms
- Relying on certain characteristics of data to assess feature importance (e.g., feature correlation, mutual information...)
- More efficient than wrapper methods
- The selected features may not be optimal for a particular learning algorithm



Embedded methods

- A trade-off between wrapper and filter methods by embedding feature selection into the model learning, e.g., decision tree

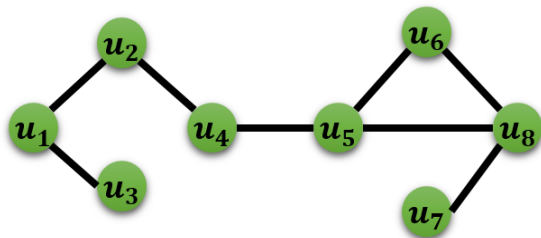


- Inherit the merits of wrapper and filter methods
 - Include the interactions with the learning algorithm
 - More efficient than wrapper methods

How to perform feature
selection on plain networks?

Scenario 1: w/o explicit node features

Input Network



Manual Feature Engineering

	f_1	f_2	...	f_d
u_1				
u_2				
u_3				
u_4				
u_5				
u_6				
u_7				
u_8				

Hand-Crafted Features

	c_1	c_2	c_3
u_1			
u_2			
u_3			
u_4			
u_5			
u_6			
u_7			
u_8			

Node Labels (optional)

Off-the-Shelf Learning Models

Conventional Feature Selection

u_1		
u_2		
u_3		
u_4		
u_5		
u_6		
u_7		
u_8		

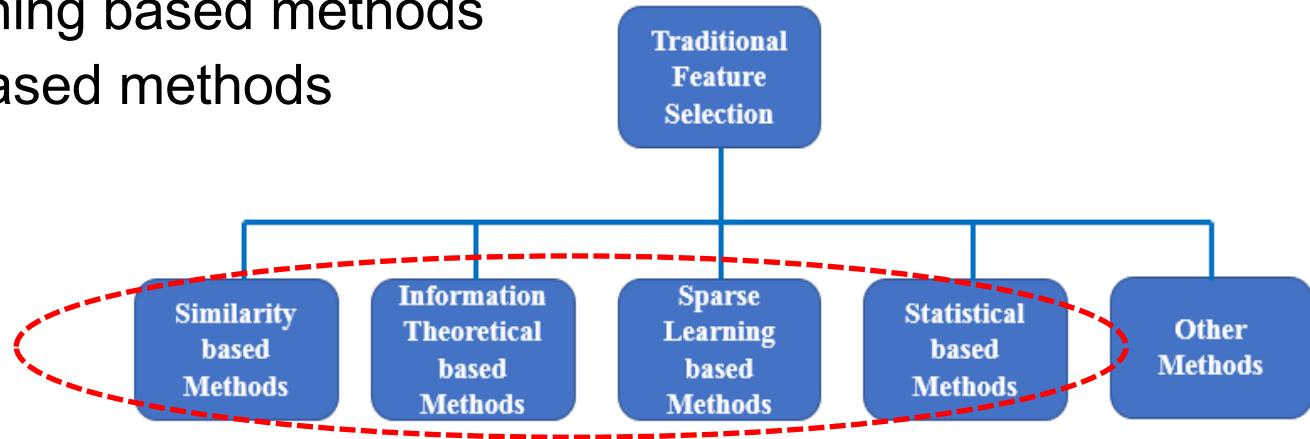
Selected features

- Classification
- Clustering
- Anomaly Detection
- Visualization

Conventional feature selection algorithms

We categorize feature selection methods mainly according to their adopted techniques

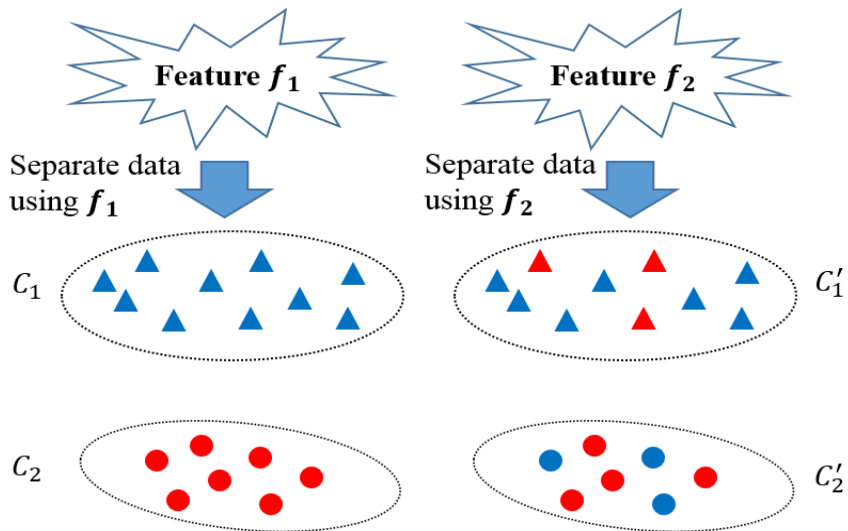
- Similarity based methods
- Information theoretical based methods
- Sparse learning based methods
- Statistical based methods



Similarity based feature selection

Similarity based methods assess the importance of features by their ability to preserve data similarity

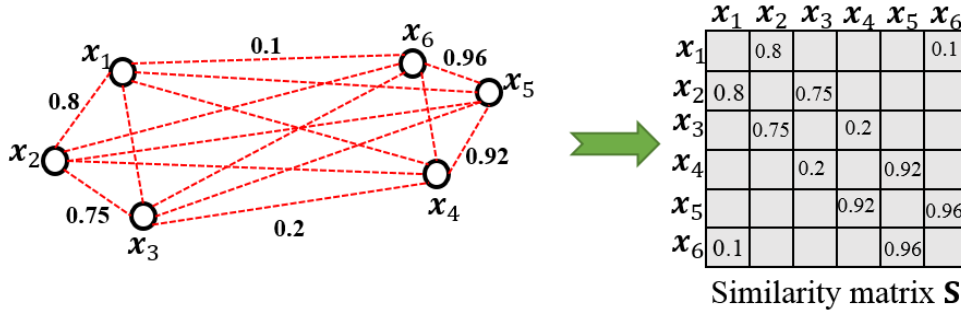
- A good feature should assign similar values to instances that are close to each other – (the “closeness” is obtained from data similarity matrix)



- Different **colors** denote different classes
- Different **shapes** denote different values assigned by a feature

Similarity based methods - similarity matrix

- Pairwise data similarity is often encoded in the data similarity matrix



- E.g., network data - explicitly given (adjacency matrix)

- E.g., w/o class labels - RBF kernel
$$S_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$$

- E.g., w/ class label - class labels
$$S_{ij} = \begin{cases} \frac{1}{n_l} & \text{if } y_i = y_j = l \\ 0 & \text{otherwise} \end{cases}$$

Similarity based methods - framework

- Suppose data similarity matrix is $\mathbf{S} \in \mathbb{R}^{n \times n}$, to find the k most relevant features \mathcal{S} , we need to maximize:

$$\max_{\mathcal{S}} U(\mathcal{S}) = \max_{\mathcal{S}} \sum_{f \in \mathcal{S}} U(f) = \max_{\mathcal{S}} \sum_{f \in \mathcal{S}} \hat{\mathbf{f}}^T \hat{\mathbf{S}} \hat{\mathbf{f}}$$

utility of feature set \mathcal{S}

utility of feature f

transformation of \mathbf{f}

transformation of \mathbf{S}

- It is often solved by greedily selecting the top k features that maximize their individual utility $U(f)$
- Different methods vary in the way how the vector \mathbf{f} and similarity matrix \mathbf{S} are transformed to $\hat{\mathbf{f}}$ and $\hat{\mathbf{S}}$

Fisher score [Duda et al., 2001]

- Given class labels, within class and between class data similarity matrix \mathbf{S}^w (local affinity) and \mathbf{S}^b (global affinity) are defined as

$$\mathbf{S}_{i,j}^w = \begin{cases} 1/n_l & \text{if } y_i = y_j = l \\ 0 & \text{otherwise} \end{cases} \quad \mathbf{S}_{i,j}^b = \begin{cases} 1/n - 1/n_l & \text{if } y_i = y_j = l \\ 1/n & \text{otherwise} \end{cases}$$

- A good feature make instances from different classes **far away** and make instances from the same class **close to each other**
- The score of the i -th feature f_i is

$$\text{score}(f_i) = \frac{\mathbf{f}'_i \mathbf{L}^b \mathbf{f}_i}{\mathbf{f}'_i \mathbf{L}^w \mathbf{f}_i}$$

Laplacian matrix obtained from \mathbf{S}^w and \mathbf{S}^b

Laplacian score [He et al., 2005]

- First, it builds the data similarity matrix \mathbf{S} , diagonal matrix \mathbf{D} and Laplacian matrix \mathbf{L} without using class labels
- Motivation: a good feature should (1) preserve data similarity structure; and (2) have high feature variance
- Then the Laplacian Score of feature f_i is:

Measure the consistency of features on the similarity matrix (smaller, the better)

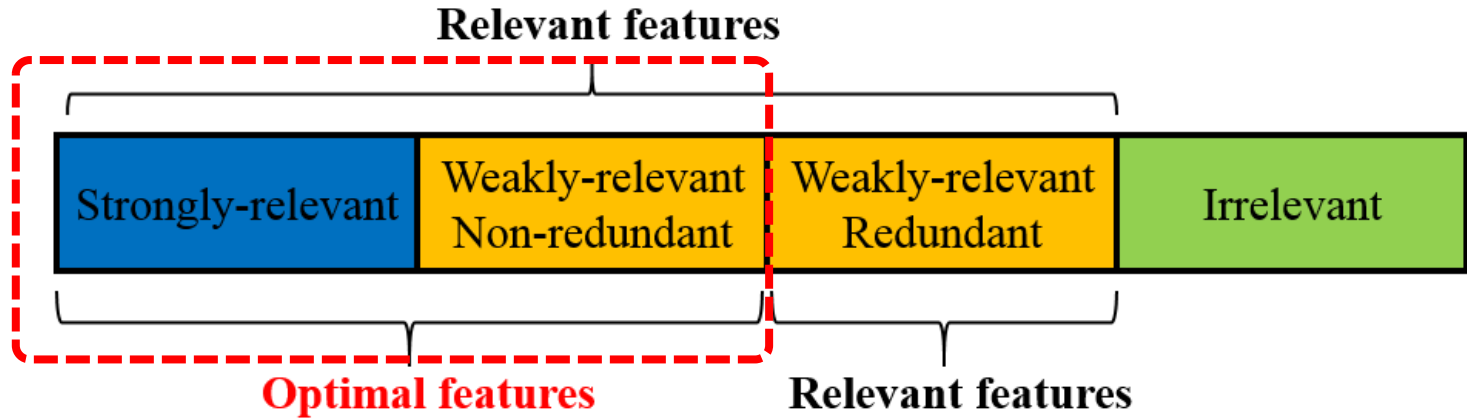
Feature variance (higher, the better)

$$score(f_i) = \frac{\tilde{\mathbf{f}}_i' \mathbf{L} \tilde{\mathbf{f}}_i}{\tilde{\mathbf{f}}_i' \mathbf{D} \tilde{\mathbf{f}}_i}, \text{ where } \tilde{\mathbf{f}}_i = \mathbf{f}_i - \frac{\mathbf{f}_i' \mathbf{D} \mathbf{1}}{\mathbf{1}' \mathbf{D} \mathbf{1}} \mathbf{1}$$

- Laplacian score is also equivalent to: $1 - \left(\frac{\tilde{\mathbf{f}}_i}{\|\mathbf{D}^{\frac{1}{2}} \tilde{\mathbf{f}}_i\|} \right)' \mathbf{S} \left(\frac{\tilde{\mathbf{f}}_i}{\|\mathbf{D}^{\frac{1}{2}} \tilde{\mathbf{f}}_i\|} \right)$

Information theoretical based methods

- Exploit different heuristic filter criteria to measure the importance of features



- Our target is to find these “optimal” features

Information theoretical based methods

- Searching for the best feature subset is NP-hard, most methods employ forward/backward sequential search heuristics
- E.g., for forward search, given selected features \mathcal{S} , we should do the following for the next selected feature f_i
 - Maximize its correlation with class labels Y :

$$I(f_i; Y) \longrightarrow \text{Information Gain}$$

- Minimize the redundancy w.r.t. selected features in \mathcal{S} :

$$\sum_{f_j \in \mathcal{S}} I(f_j; f_k) \longrightarrow \text{Information Gain}$$

- Maximize its complementary info w.r.t. selected features in \mathcal{S} :

$$\sum_{f_j \in \mathcal{S}} I(f_j; f_k | Y) \longrightarrow \text{Conditional Information Gain}$$

Information theoretic based methods - framework

- Given selected features \mathcal{S} , the feature score for the next selected feature f_i can be determined by

$$score(f_k) = I(f_k; Y) + \sum_{f_j \in \mathcal{S}} g[I(f_j; f_k), I(f_j; f_k|Y)]$$

$g(*)$: a function

- If $g(*)$ is a linear function, then it can be represented as

$$score(f_k) = I(f_k; Y) - \beta \sum_{f_j \in \mathcal{S}} I(f_j; f_k) + \lambda \sum_{f_j \in \mathcal{S}} I(f_j; f_k|Y)$$

Between 0 and 1

- But also, $g(*)$ can be a nonlinear function

Information gain [Lewis, 1992]

- Information gain only measures the feature importance by its correlation with class labels
- The information gain of a new unselected feature f_k

$$\text{score}(f_k) = I(f_k; Y)$$

- Selecting features independently
- It is a special case of the linear function by setting $\beta = \lambda = 0$

$$\text{score}(f_k) = I(f_k; Y) - \beta \sum_{f_j \in \mathcal{S}} I(f_j; f_k) + \lambda \sum_{f_j \in \mathcal{S}} I(f_j; f_k | Y)$$

Min. redundancy max. relevance [Peng et al., 2005]

- Intuitively, with more selected features, the effect of feature redundancy should gradually decrease
- The score of a new unselected feature f_k is

$$\text{score}(f_k) = I(f_k; Y) - \frac{1}{|\mathcal{S}|} \sum_{f_j \in \mathcal{S}} I(f_k; f_j)$$

reduced effect of feature redundancy

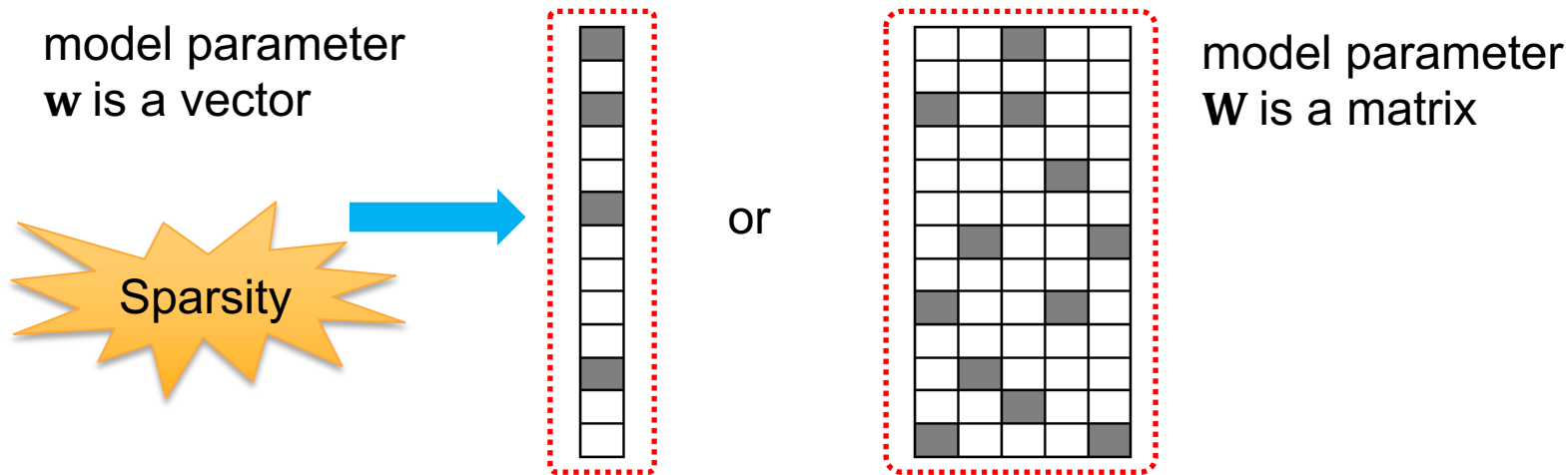
- MRMR is also a special case of the linear function by setting $\lambda = 0$ and adjusting β adaptively

Sparse learning based methods

- The selected features of aforementioned methods may not be optimal for a particular learning task
- Embedded methods embed feature selection into model construction – two phases complement each other
- Sparse learning based methods is a popular type of embedded methods with several advantages
 - With strong theoretical guarantee
 - Empirical success in many real-world applications
 - Flexible models for complex feature structures

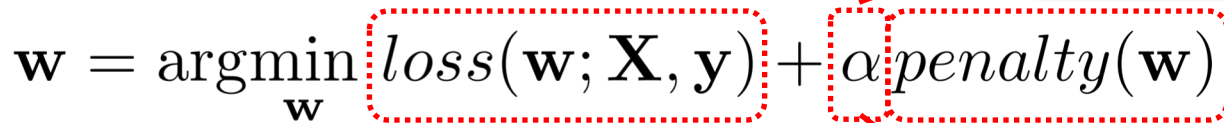
What is feature sparsity?

- The model parameters in many data mining tasks can be represented as a vector \mathbf{w} or a matrix \mathbf{W}
- Sparsity indicates that many elements in \mathbf{w} and \mathbf{W} are small or exactly zero



Sparse learning methods - framework

- Let us start from the binary classification or the univariate regression problem
- Let \mathbf{w} denote the model parameter (a.k.a. feature coefficient), it can be obtained by solving

$$\mathbf{w} = \underset{\mathbf{w}}{\operatorname{argmin}} \underbrace{\operatorname{loss}(\mathbf{w}; \mathbf{X}, \mathbf{y})}_{\text{Loss}} + \underbrace{\alpha \operatorname{penalty}(\mathbf{w})}_{\text{Penalty}}$$


Balance parameter

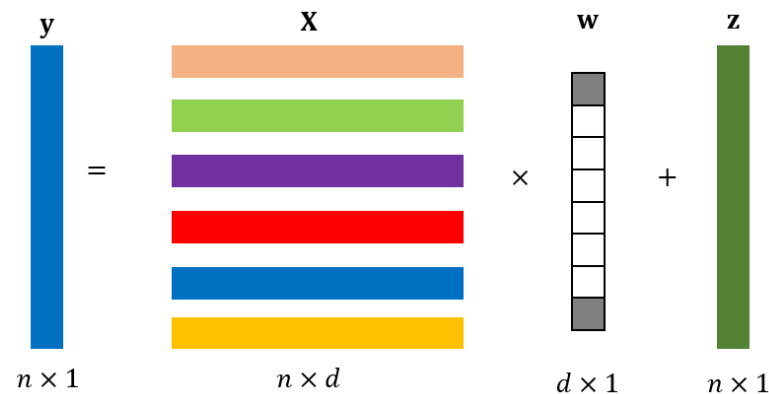
- Least squares loss
- Hinge loss
- Logistic loss...

- $\|\mathbf{w}\|_0$ seeks for optimal features
- However, it is not a valid norm, nonconvex and NP-hard
- It is often relaxed to $\|\mathbf{w}\|_1$ (Lasso), which is the tightest convex hull

Lasso [Tibshirani, 1996]

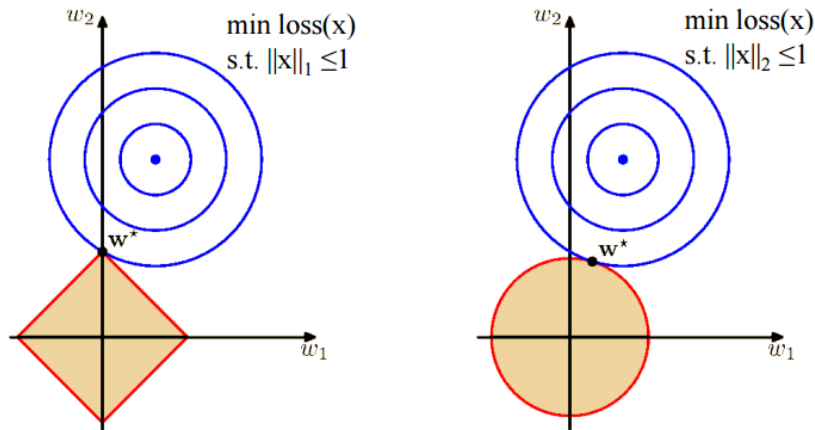
- Based on ℓ_1 -norm regularization on weight \mathbf{w}

$$\min_{\mathbf{w}} \text{loss}(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha \|\mathbf{w}\|_1 \quad \longrightarrow \quad \min_{\mathbf{w}} \text{loss}(\mathbf{w}; \mathbf{X}, \mathbf{y}) \text{ s.t. } \|\mathbf{w}\| \leq t$$



The feature score of the i -th feature is $|w_i|$; the higher the value, the more important the feature is

- Why ℓ_1 -norm induces sparsity?

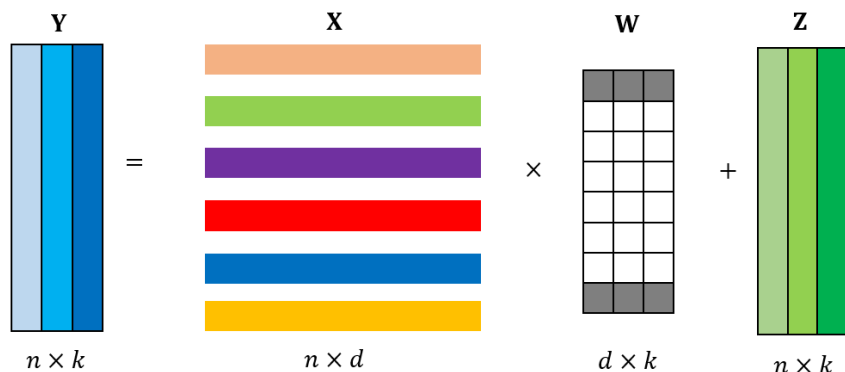


Extension to multi-class or multivariate problems

- Require feature selection results to be consistent across multiple targets in multi-class classification or multi-variate regression

$$\min_{\mathbf{W}} \text{loss}(\mathbf{W}; \mathbf{X}, \mathbf{Y}) + \alpha \|\mathbf{W}\|_{2,1}$$

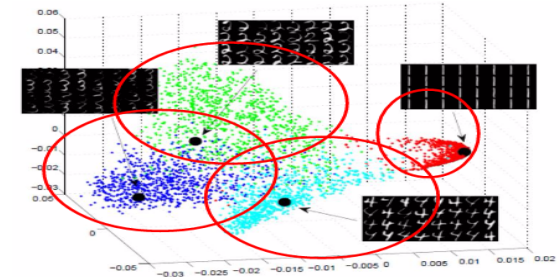
- $\|\mathbf{W}\|_{2,1}$ achieves joint feature sparsity across multiple targets



The feature score of the i -th feature is $\|\mathbf{W}_{i*}\|_2$ - the higher the value, the more important the feature is

Unsupervised sparse learning based methods

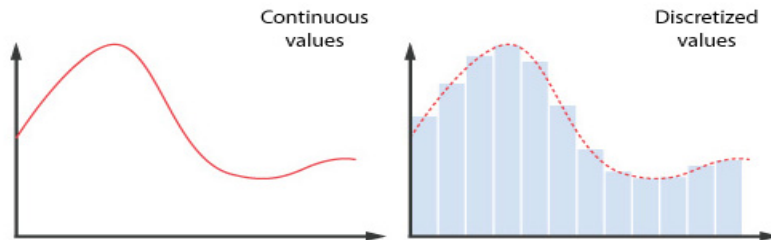
- Without class labels, we attempt to find discriminative features that can preserve data clustering structure
- There are two options
 - Obtain clusters and then perform FS (e.g., MCFS)
 - Embed FS into clustering (e.g., NDFS)
- The 2nd option is preferred as not all features are useful to find clustering structure



Type 1	Data → Clustering Structure → Learning Model	Typical methods: MCFS, MRFS, SPFS, FSSL...
Type 2	Data → Clustering Structure → Learning Model ↑ └──────────────────────────┘	Typical methods: NDFS, JELSR, RUFS, EUFS...

Statistical based methods

- This family of algorithms are based on different statistical measures to measure feature importance
- Most algorithms evaluate features individually, so the feature redundancy is inevitably ignored
- Most algorithms can only handle discrete data, the numerical features have to be discretized first



T-Score [Davis and Sampson, 1986]

- It is used for binary classification problems
- Assess whether the feature makes the means of samples from two classes statistically significant
- The t-score of each feature f_i is

$$t_score(f_i) = \frac{|\mu_1 - \mu_2|}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

Mean value of samples from the first class

Mean value of samples from the second class

Standard deviation value for samples from the first class

Standard deviation value for samples from the second class

- The higher the T-score, the more important the feature is

Chi-square score [Liu and Setiono, 1995]

- Utilize independence test to assess whether the feature is independent of class label
- Given a feature f_i with r values, its feature score is

$$Chi_square_score(f_i) = \sum_{j=1}^r \sum_{s=1}^c \frac{(n_{js} - \mu_{js})^2}{\mu_{js}}$$

instances with the j -th feature value and in class s

$$\mu_{js} = \frac{n_{*s} n_{j*}}{n}$$

instances with the j -th feature value

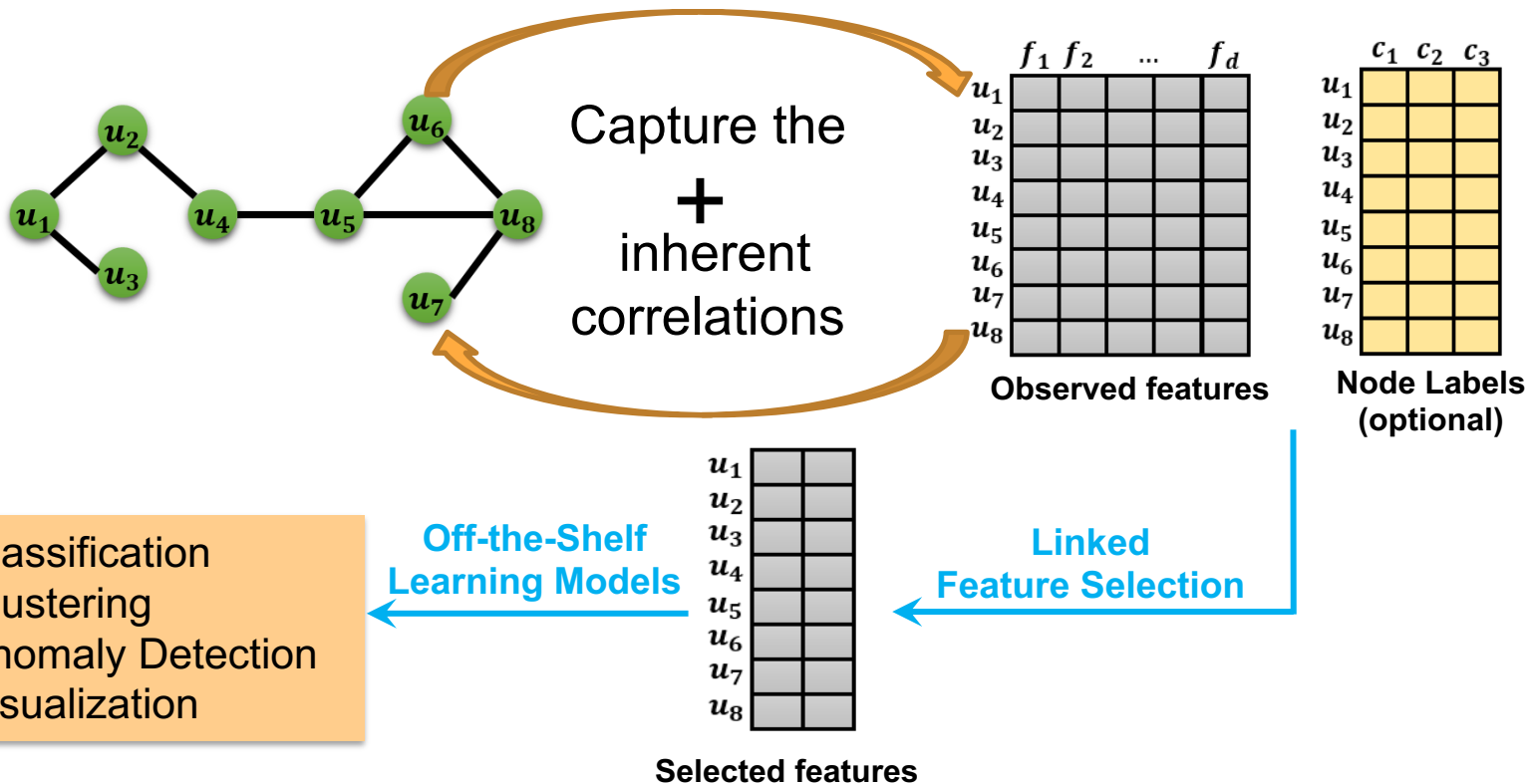
instances in class s

- Higher chi-square indicates that the feature is more important

How to perform feature
selection on attributed
networks?

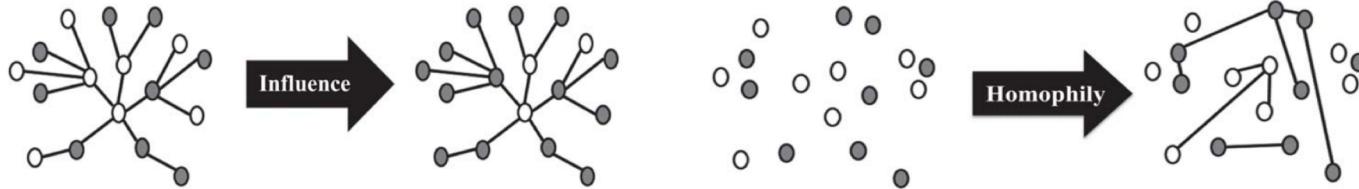
Scenario 2: w/ explicit node features

Directly perform feature selection on the observed node attributes

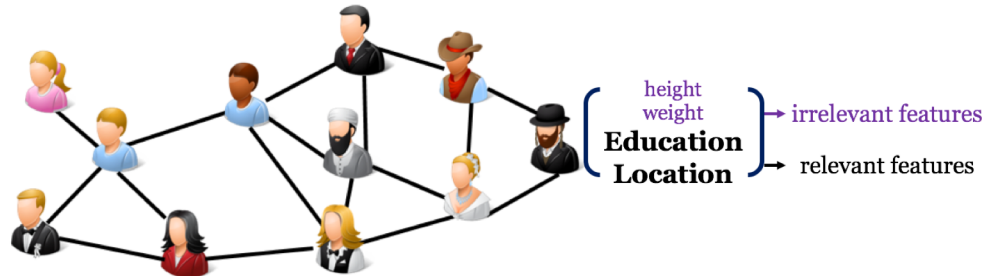


Why linked feature selection?

- Social Influence & Homophily: node features and network are inherently correlated – ***assortativity patterns***



- Many learning tasks are **enhanced** by modeling the correlation
 - Community detection
 - Anomaly detection
 - Node classification



- But not all features are highly correlated with network structure!

FSNet [Gu and Han, 2011]

- Use a linear classifier to capture the relationship between observed node features \mathbf{X} and class labels \mathbf{Y}

$$\min_{\mathbf{W}} \|\mathbf{XW} - \mathbf{Y}\|_F^2 + \alpha \|\mathbf{W}\|_{2,1} + \beta \|\mathbf{W}\|_F^2$$

Joint feature sparsity (points to $\|\mathbf{W}\|_{2,1}$)

Avoid overfitting (points to $\|\mathbf{W}\|_F^2$)

- Employ graph regularization to model links

$$tr(\mathbf{W}'\mathbf{X}'\mathbf{L}\mathbf{X}\mathbf{W})$$

undirected network: $\mathbf{L} = \mathbf{D} - \mathbf{A}$

directed network: $\mathbf{L} = \mathbf{\Pi} - \frac{1}{2}(\mathbf{\Pi}\mathbf{P} + \mathbf{P}'\mathbf{\Pi})$

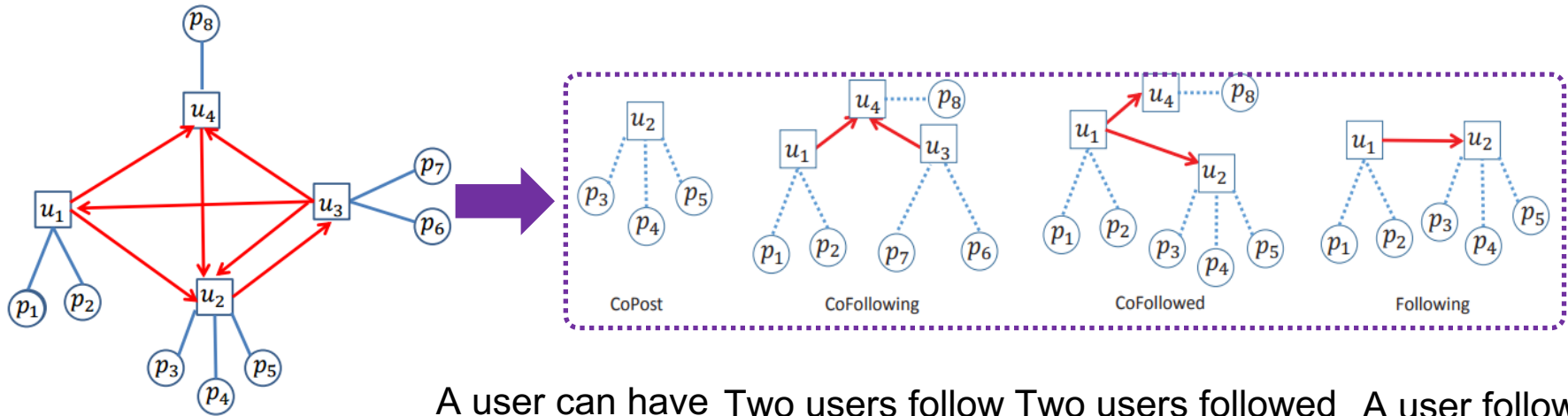
\mathbf{P} : transition matrix of random walk
 $\mathbf{\Pi}$: stationary distribution

- Objective function of FSNet

$$\min_{\mathbf{W}} \|\mathbf{XW} - \mathbf{Y}\|_F^2 + \alpha \|\mathbf{W}\|_{2,1} + \beta \|\mathbf{W}\|_F^2 + \gamma tr(\mathbf{W}'\mathbf{X}'\mathbf{L}\mathbf{X}\mathbf{W})$$

LinkedFS [Tang and Liu, 2012]

- Investigate feature selection on social media data with various types of social relations: four basic types



u_i : user p_i : post

A user can have multiple posts Two users follow a third user Two users followed by a third user A user follows another user

- These social relations are supported by the assortativity patterns

LinkedFS [Tang and Liu, 2012]

- For CoPost hypothesis
 - Posts by the same user are likely to be of similar topics
- Feature selection with the CoPost hypothesis

$$\min_{\mathbf{W}} \|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_F^2 + \alpha \|\mathbf{W}\|_{2,1} + \beta \sum_{u \in \mathbf{u}} \sum_{\{p_i, p_j\} \in \mathbf{P}_u} \|\mathbf{X}(i, :)\mathbf{W} - \mathbf{X}(j, :)\mathbf{W}\|_2^2$$

CoPost hypothesis

CoPost relations

- Observations

	BlogCatalog	Digg
# Posts	7,877	9,934
# Original Features	84,233	12,596
# Features after <i>TFIDF</i>	13,050	6,544
# Classes	14	15
# Users	2,242	2,561
# Following Relations	55,356	41,544
Ave # Posts	3.5134	3.8790
Max # Followers	820	472
Min # Followers	1	1
Network Density	0.0110	0.0063
Clustering Coefficient	0.3288	0.2461



Improvement in Digg(%)				
Datasets	CP	CFI	CFE	FI
\mathcal{T}_5	+14.54	+7.01	+4.69	+15.25
\mathcal{T}_{25}	+4.59	+1.59	0	+4.02
\mathcal{T}_{50}	+7.19	+3.92	+1.05	+8.48
\mathcal{T}_{100}	+4.90	+3.15	+1.63	+4.64
Improvement in BlogCatalog(%)				
Datasets	CP	CFI	CFE	FI
\mathcal{T}_5	+10.71	+7.89	+7.89	+12.62
\mathcal{T}_{25}	+10.04	+5.00	+5.00	+9.50
\mathcal{T}_{50}	+9.70	+2.16	+2.16	+7.34
\mathcal{T}_{100}	+7.18	+0.46	+0.46	+7.67

Personalized FS [Li et al., 2017]

- Features of nodes are highly idiosyncratic



The price comes down! #apple

8:14 PM - 15 Jun 2019



student who wants to buy apple product

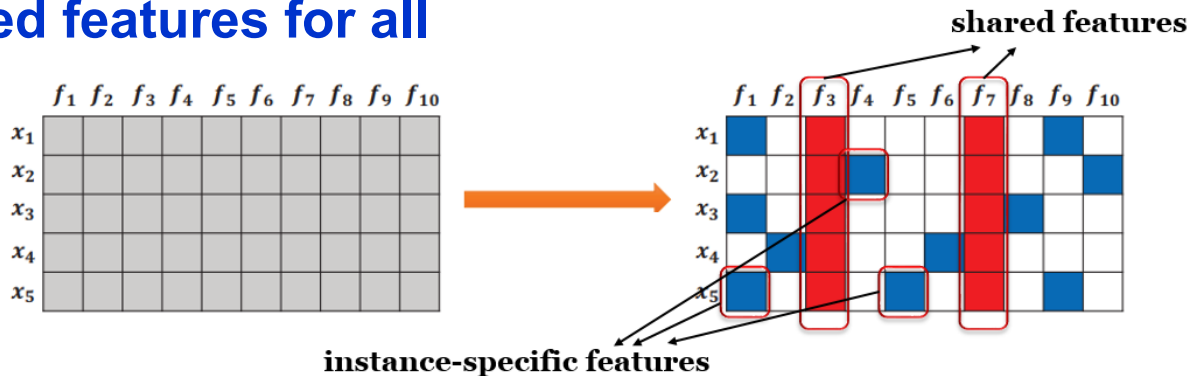


businessman who holds a lot of apple stock



[Wu et al. 2016]

- How to address the idiosyncrasy nature of node attributes?
- Solution: find a subset of **personalized features for each individual** and **shared features for all**



Personalized feature selection [Li et al., 2017]

- To find personalized features, we attempt to achieve feature sparsity within each local feature weight

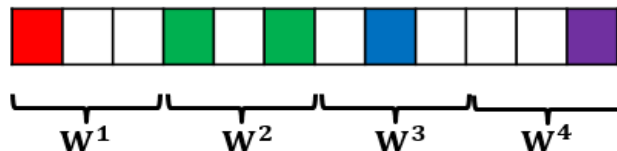
$$\min_{\tilde{\mathbf{W}}, \mathbf{W}^i} \sum_{i=1}^n \|\mathbf{x}_i(\tilde{\mathbf{W}} + \mathbf{W}^i) - \mathbf{y}_i\|_2^2 + \beta \sum_{i=1}^n \|\mathbf{W}^i\|_{2,1}^2 + \gamma \|\tilde{\mathbf{W}}\|_{2,1}$$

global feature weight for all nodes

local feature weight for the i -th nodes

exclusive group lasso for personalized features

group lasso for shared features



- To reduce overfitting, we impose penalty on the local feature weight

$$\min_{\tilde{\mathbf{W}}, \mathbf{W}^i} \sum_{i=1}^n \|\mathbf{x}_i(\tilde{\mathbf{W}} + \mathbf{W}^i) - \mathbf{y}_i\|_2^2 + \beta \sum_{i=1}^n \|\mathbf{W}^i\|_{2,1}^2 + \gamma \|\tilde{\mathbf{W}}\|_{2,1} + \alpha \sum_{i,j=1}^n \mathbf{A}_{ij} \|\mathbf{W}^i - \mathbf{W}^j\|_F$$

Incentivize local feature weight difference to be exactly zero

Comparison with global feature selection methods

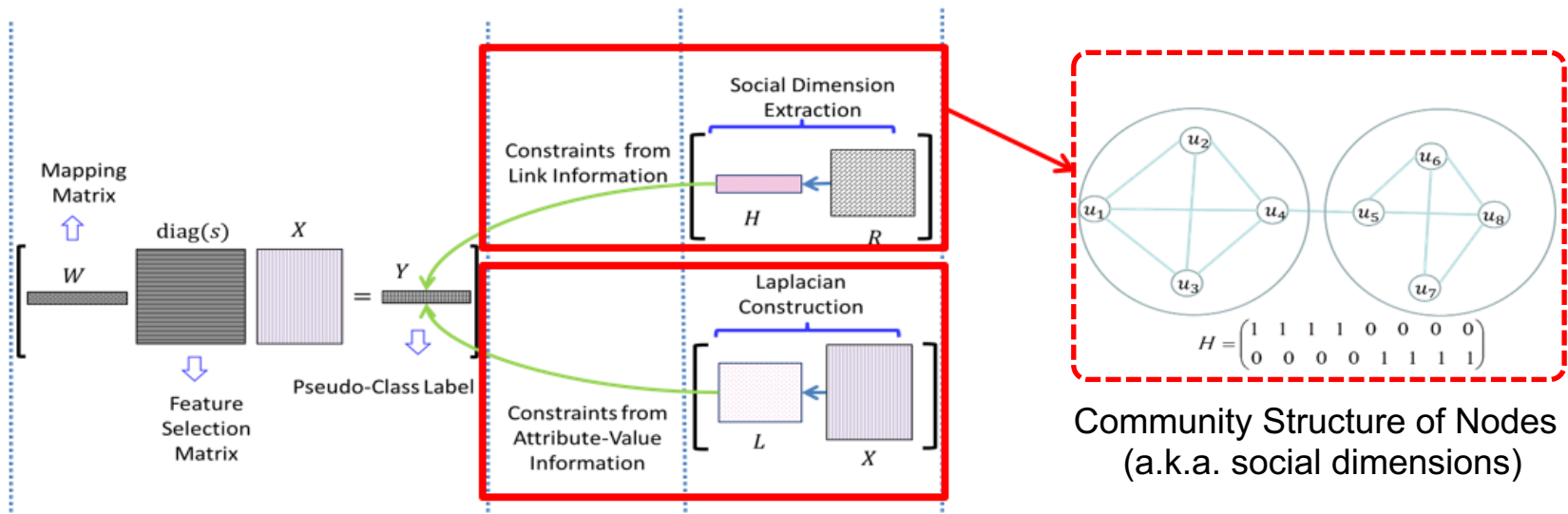
Node classification on Cora

Training Ratio		1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1	NMF	0.5342	0.5938	0.6235	0.6531	0.6570	0.6617	0.6739	0.6787	0.6854	0.6938
	wvRN	0.2516	0.3181	0.3511	0.3928	0.4204	0.4372	0.4598	0.4727	0.4849	0.5026
	SocDim	0.3697	0.4419	0.4741	0.5078	0.5340	0.5502	0.5680	0.5831	0.5947	0.5952
	GNMF	0.5632	0.6063	0.6504	0.6587	0.6634	0.6743	0.6771	0.6873	0.6880	0.6927
	FsNet	0.5363	0.6096	0.6240	0.6308	0.6467	0.6359	0.6422	0.6444	0.6408	0.6433
	PRL	0.6009	0.6127	0.6341	0.6622	0.6767	0.6939	0.7117	0.7184	0.7235	0.7365
Macro-F1	NMF	0.5279	0.5856	0.6184	0.6479	0.6529	0.6579	0.6693	0.6748	0.6804	0.6885
	wvRN	0.2276	0.3043	0.3416	0.3836	0.4123	0.4299	0.4495	0.4607	0.4722	0.4902
	SocDim	0.3651	0.4372	0.4690	0.5023	0.5293	0.5429	0.5599	0.5754	0.5863	0.5869
	GNMF	0.5533	0.6006	0.6236	0.6544	0.6571	0.6689	0.6733	0.6819	0.6925	0.6963
	FsNet	0.5189	0.6010	0.6175	0.6306	0.6452	0.6338	0.6417	0.6436	0.6398	0.6426
	PRL	0.5720	0.6153	0.6447	0.6697	0.6661	0.6923	0.7143	0.7153	0.7228	0.7335

PRL outperforms FsNet, showing the effectiveness of finding personalized features

LUFS [Tang and Liu, 2012]

- Nodes are often unlabeled on networks
- No explicit signal for feature relevance
- Fortunately, links provide additional constraints



LUFS [Tang and Liu, 2012]

- Obtain within, between, and total social dimension scatter matrices

$$\mathbf{S}_w = \mathbf{Y}'\mathbf{Y} - \mathbf{Y}'\mathbf{F}\mathbf{F}'\mathbf{Y}, \quad \mathbf{S}_b = \mathbf{Y}'\mathbf{F}\mathbf{F}'\mathbf{Y}, \quad \mathbf{S}_t = \mathbf{Y}'\mathbf{Y} \quad \mathbf{F} = \mathbf{H}(\mathbf{H}'\mathbf{H})^{-\frac{1}{2}}$$

Y' → Pseudo labels Weighted social dimension matrix ← H

- Instances are similar within social dimensions while dissimilar between social dimensions

$$\max_{\mathbf{W}} \text{tr}((\mathbf{S}_t)^{-1}\mathbf{S}_b)$$

- Attribute similarity should be consistent of pseudo label similarity

$$\min \text{tr}(\mathbf{Y}'\mathbf{L}\mathbf{Y}) \rightarrow \text{similarity matrix using RBF}$$

- Objective function of LUFS

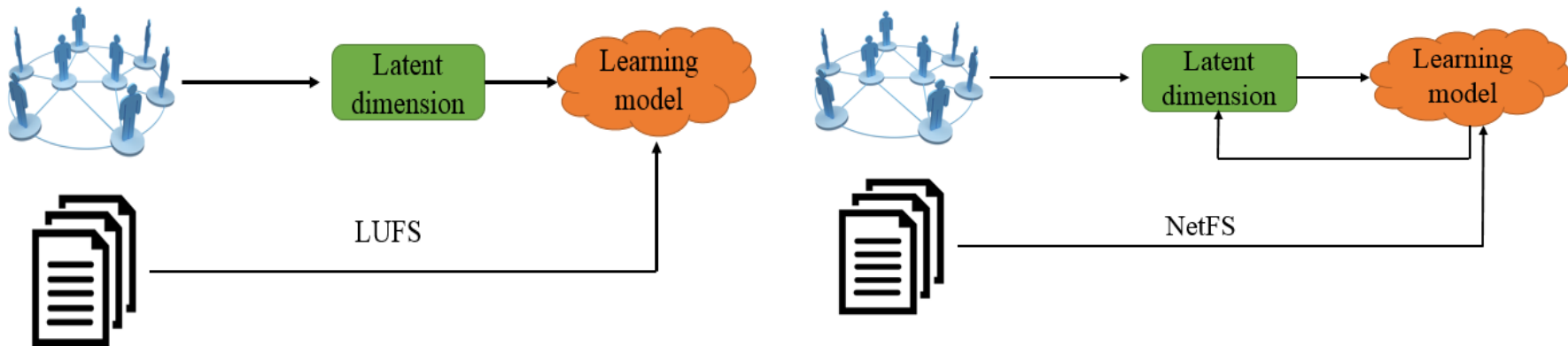
$$\min_{\mathbf{W}, \mathbf{s}} \text{tr}(\mathbf{Y}\mathbf{L}\mathbf{Y}') - \alpha \text{tr}((\mathbf{S}_t)^{-1}\mathbf{S}_b)$$

$$\mathbf{Y} = \mathbf{W}'\text{dig}(\mathbf{s})\mathbf{X} \quad \text{s.t. } \mathbf{s} \in \{0, 1\}^d, \mathbf{s}'\mathbf{1} = k,$$

$$\|\mathbf{Y}(:, i)\|_0 = 1, 1 \leq i \leq n.$$

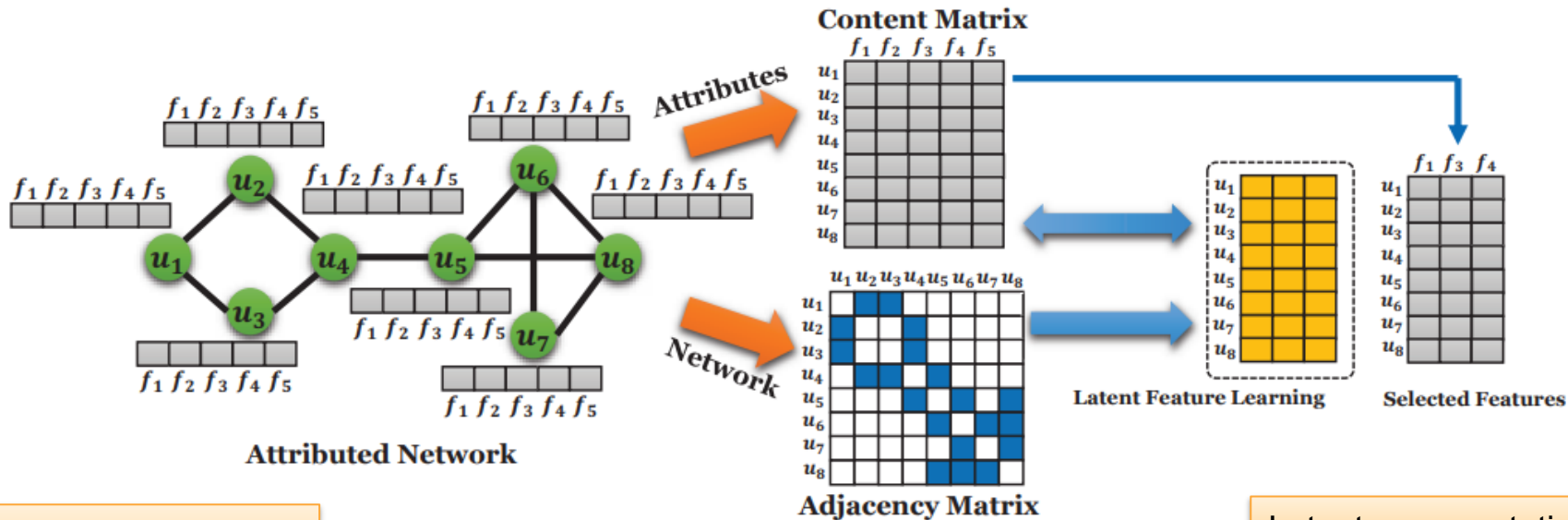
NetFS [Li et al., 2016]

- LUFS performs network structure modeling and feature selection separately
- NetFS embeds latent representation modeling into feature selection and is more robust to noise links



Difference between LUFS and NetFS

NetFS [Li et al., 2016]



Latent representation of nodes as constraints

Latent representation

$$\min_{\mathbf{U} \geq 0, \mathbf{W}} \|\mathbf{XW} - \mathbf{U}\|_F^2 + \alpha \|\mathbf{W}\|_{2,1} + \frac{\beta}{2} \|\mathbf{A} - \mathbf{UU}'\|_F^2$$

Joint feature sparsity

Comparison between NetFS vs. LUFS

- Perform unsupervised feature selection then apply K-means
- Evaluate feature quality by clustering results

BlogCatalog					
# features	200	400	600	800	1000
LapScore	26.75	28.95	26.35	24.52	32.91
SPEC	17.90	18.01	18.90	19.55	20.52
NDFS	24.61	32.35	33.43	31.89	34.85
LUFS	21.52	21.70	31.72	31.79	32.39
NetFS	50.89	42.38	42.96	42.73	43.17

Flickr					
# features	200	400	600	800	1000
LapScore	12.30	12.37	12.42	13.21	13.28
SPEC	11.87	12.48	13.15	13.89	14.32
NDFS	15.52	17.23	27.21	29.94	35.70
LUFS	13.25	18.19	20.40	23.59	22.53
NetFS	22.18	30.39	35.49	36.51	35.52



Conventional feature selection with features



Separate the network structure modeling and feature selection



The proposed NetFS performs better!

Summary

- Curse of dimensionality and dimensionality reduction
- Feature selection and its categorization
- Conventional feature selection w/o explicit node features
- Linked feature selection w/ explicit node features