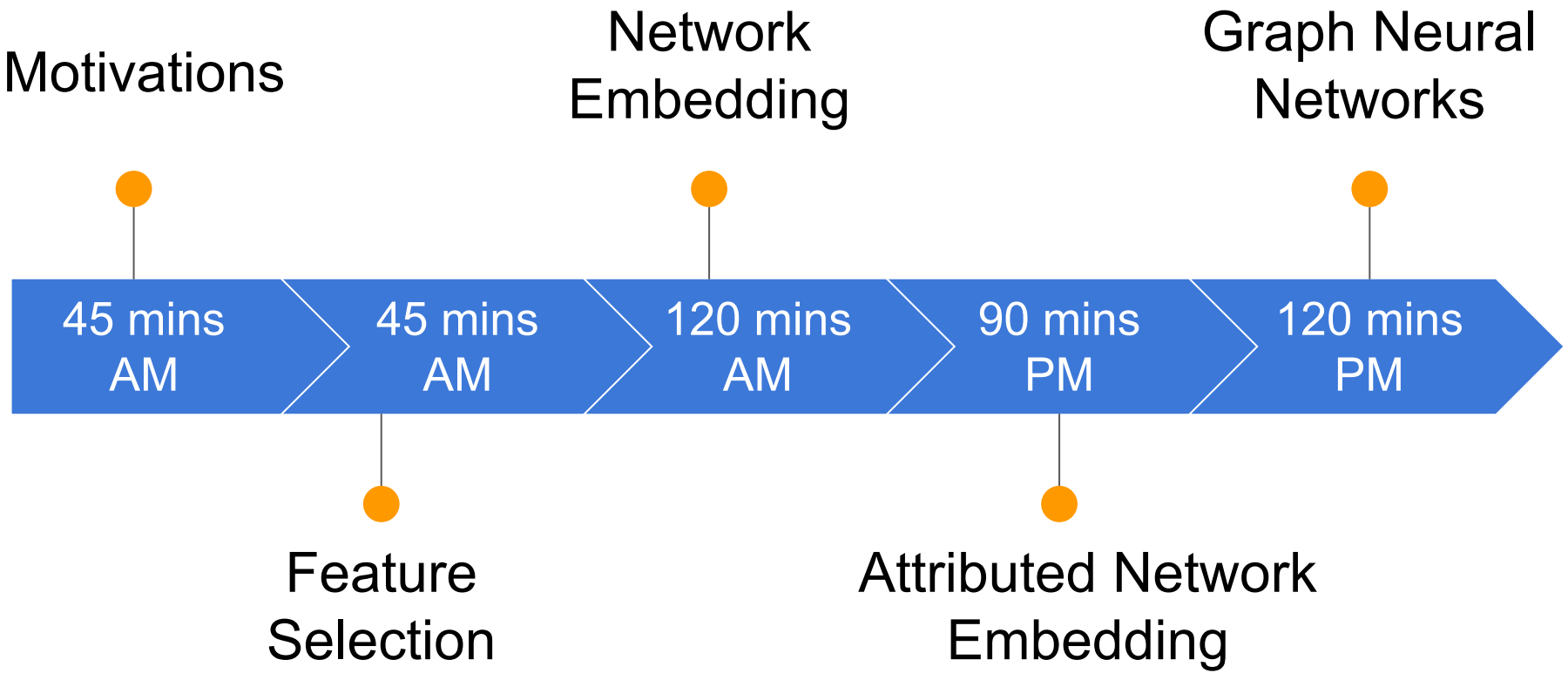
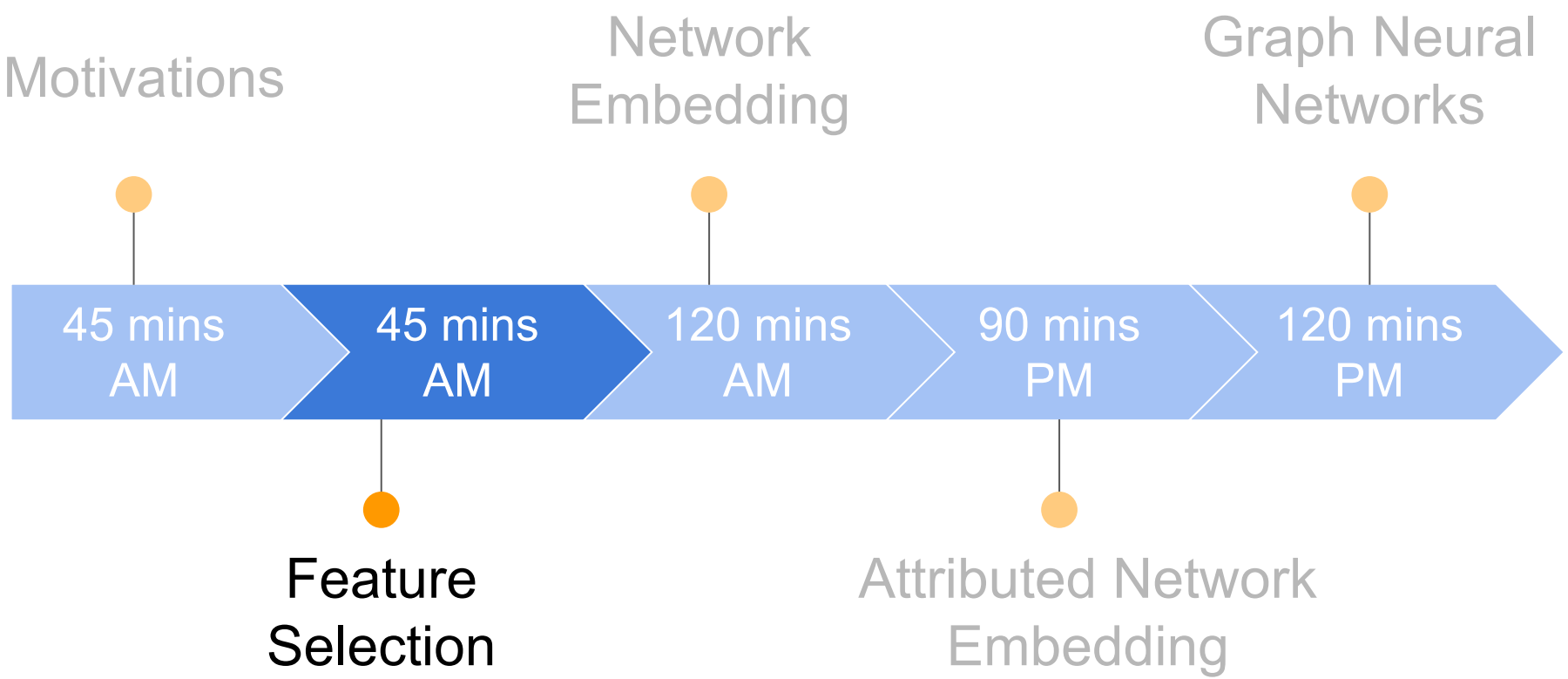
Learning From Networks

——*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; jundongl@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
45 mins AM
45 mins AM
120 mins AM
90 mins PM
120 mins PM
Feature Selection
Attributed Network Embedding


Motivations
Network Embedding
Graph Neural Networks
45 mins AM
45 mins AM
120 mins AM
90 mins PM
120 mins PM
Feature Selection
Attributed Network Embedding

Features from Networks

To facilitate various graph mining algorithms, the first step is to obtain the feature representations of nodes

- 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., profile information 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 could generate **a large number** of features
  - Not clear what features could be useful for learning on graphs

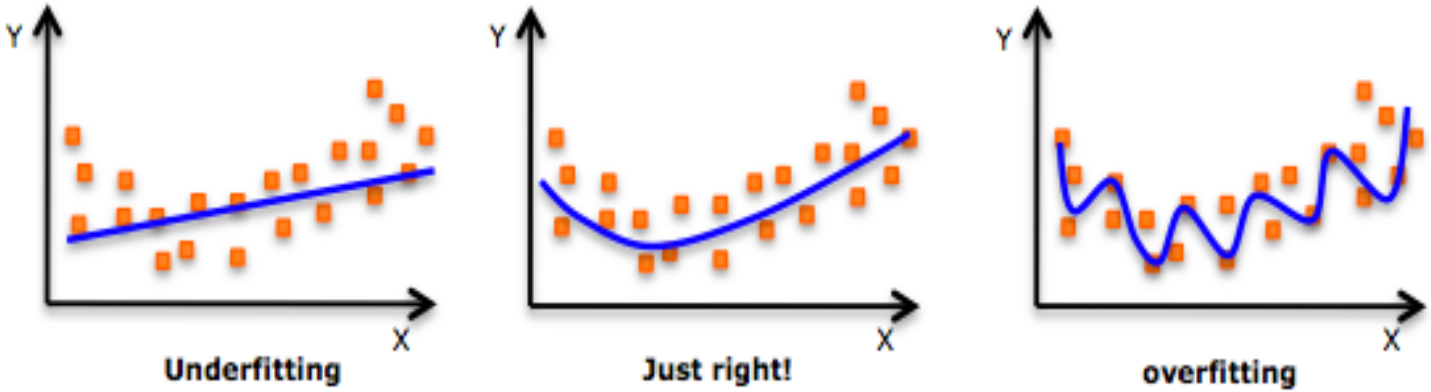
- Scenario 2: w/ explicit node features
  - Observed node features are very high-dimensional, noisy, and sparse
  - The intrinsic dimension of data may be small, e.g., the number of genes responsible for a certain disease

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

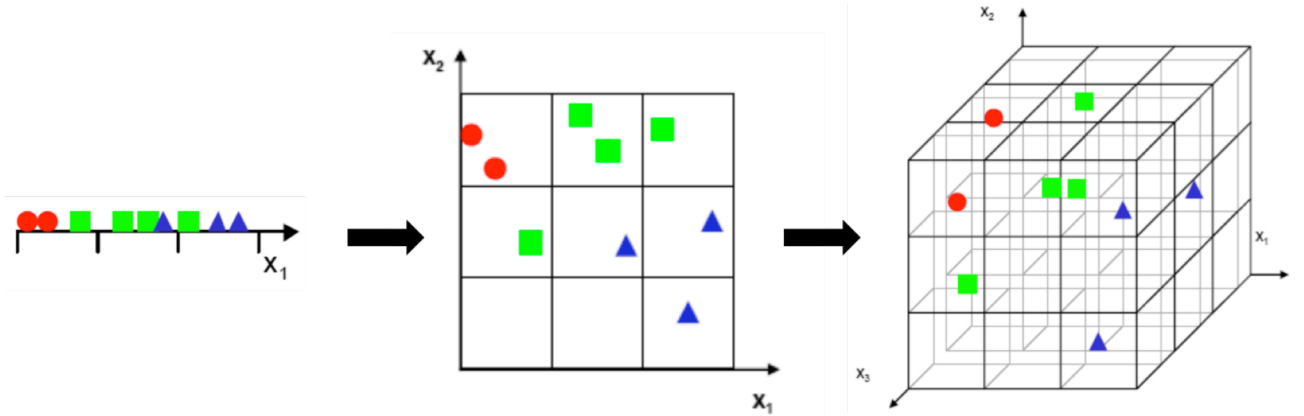
- For instance, to estimate the covariance matrix with d^2 parameters, we need $n > d^2$; otherwise we have less than one node (on average) per parameter



Curse of Dimensionality - Required Samples

- Data sparsity becomes exponentially worse as the feature dimension increases

- Conventional distance metrics become ineffective

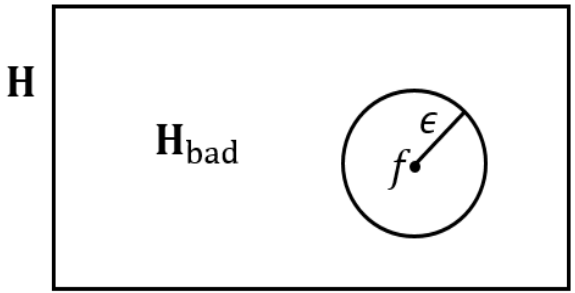


3 samples per region **1 sample per region** **1/3 sample per region**

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

How Many Data Samples Do We Need?

- How do we know that the hypothesis h is close to the target function f if we don't know what f is? – PAC Learning Theory
- $\mathbf{X} \in \mathbb{R}^{N \times d}$: data samples; $\mathbf{H}$: the set of possible hypotheses
- A hypothesis h is called approximately correct if $\text{error}(h) \leq \epsilon$, and $\text{error}(h_b) \geq \epsilon$ (where $h_b \in \mathbf{H}_{bad}$)
- The probability that it agrees with a given example is at most $1 - \epsilon$



$$P(h_b \text{ aggrees with N samples}) \leq (1-\epsilon)^N$$

How Many Data Samples Do We Need?

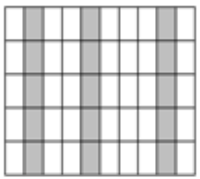
- The probability contains at least one consistent hypothesis

$$P(\mathbf{H}_{bad} \text{ contains a consistent hypothesis}) \leq |\mathbf{H}_{bad}|(1-\epsilon)^N \leq |\mathbf{H}|(1-\epsilon)^N$$

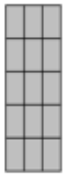
- Reduce the probability of this event below a small number δ

$$|\mathbf{H}|(1-\epsilon)^N \leq \delta \quad \Rightarrow \quad N \geq \frac{1}{\epsilon}(ln\frac{1}{\delta} + ln|\mathbf{H}|)$$

- If $\mathbf{H}$ is the set of all Boolean functions on d features, then we have $|\mathbf{H}| = 2^{2^d}$, and the sample complexity grows as 2^d
- If we reduce d, we can make data "***bigger***"
  - Before reducing d, $5/2^{10}$=$5/2^{10}$
  - After reducing d, $5/2^3$=$5/8$



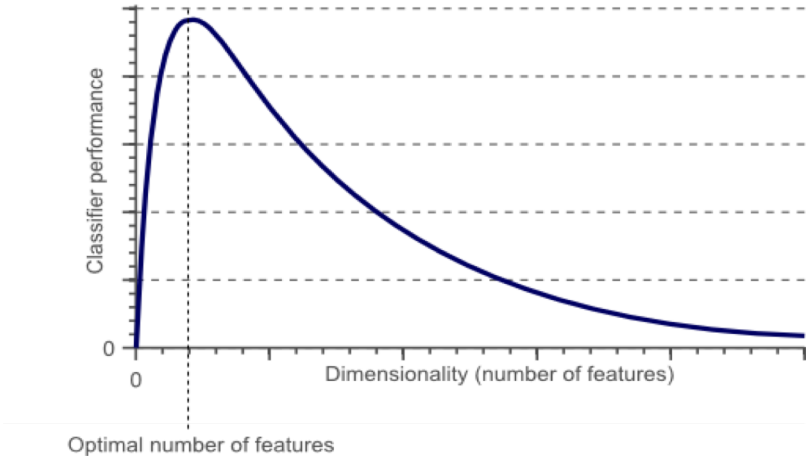
$\mathbf{X} \in \mathbb{R}^{5\times 10}$



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

Curse of Dimensionality - Summary

- In practice, the curve of learning performance w.r.t. the feature dimension looks like this

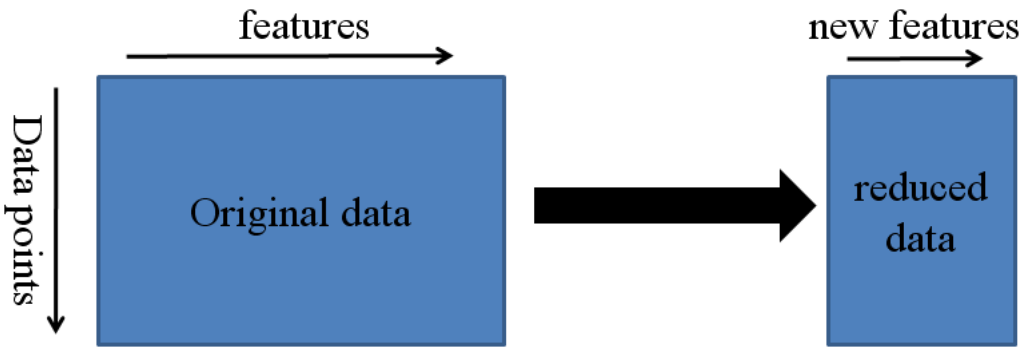


http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

- For a fixed sample size n, it is often the case that there is an optimal number of features to use

Dimensionality Reduction

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



Feature Extraction

- Project the original high-dimensional data into a new feature space of low dimensionality

- Given a set of n data instances $\{x_1, x_2, \ldots, x_n\}$ with d features, obtain the low-dimensional representations:

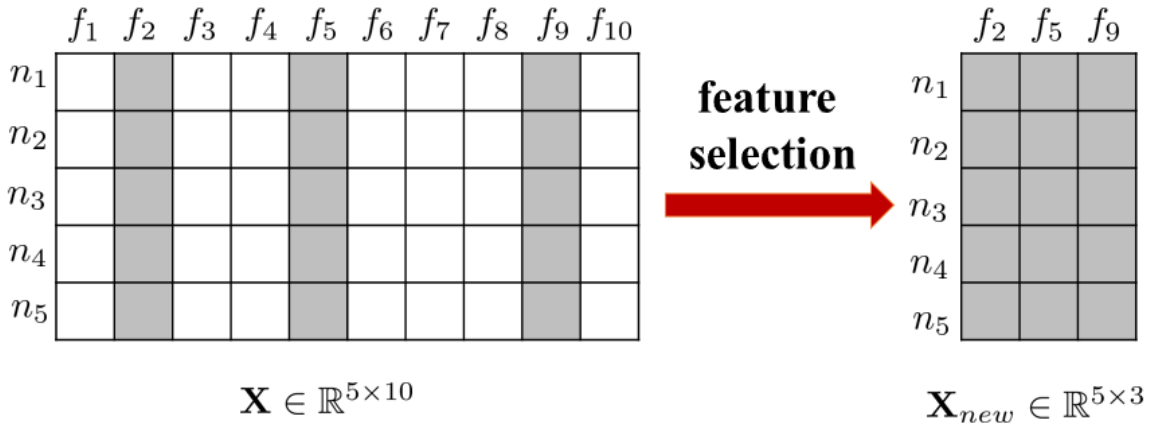
$$\mathbf{x}_i \in \mathbb{R}^d \rightarrow \mathbf{y}_i \in \mathbb{R}^p \ (p \ll d)$$

- 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

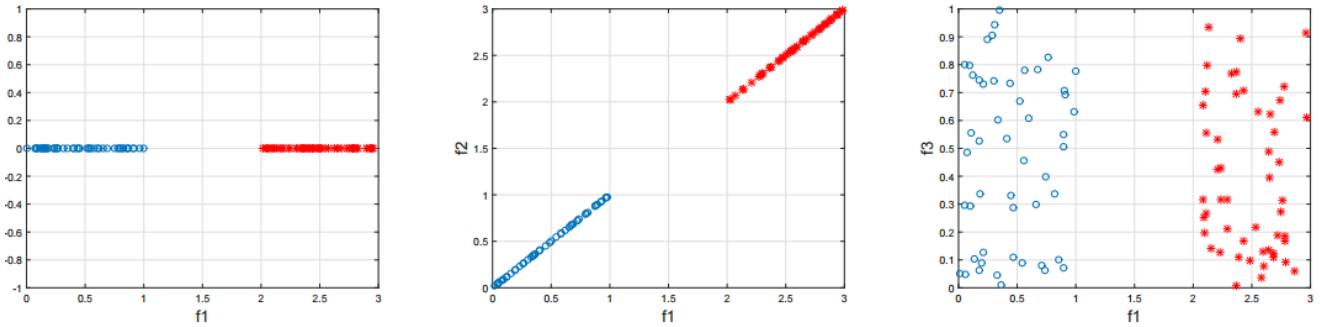
- Feature selection selects an “optimal” subset of features from the original high-dimensional feature set with a ***certain criterion***



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

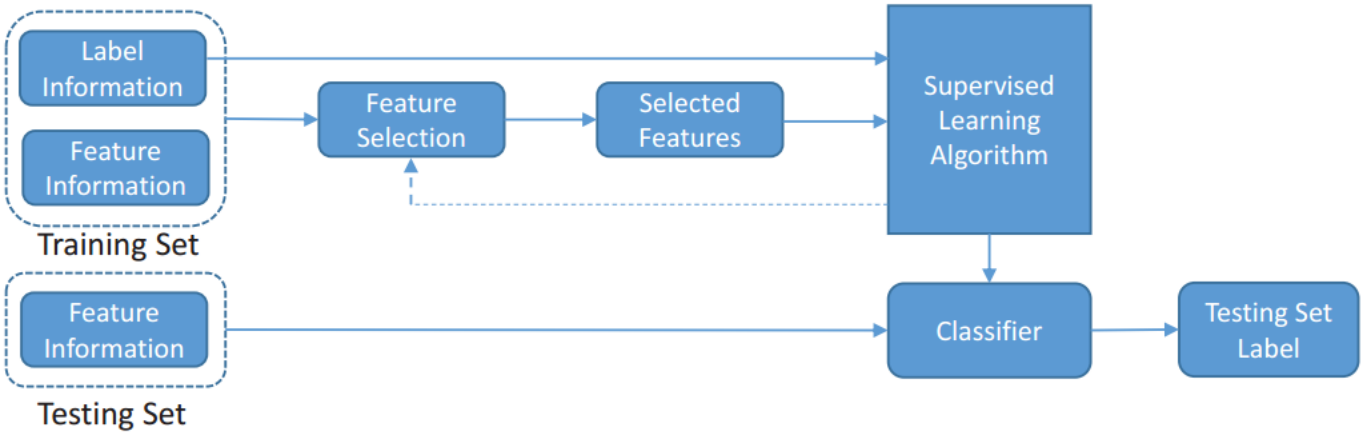
- From the label perspective:
  - Supervised
  - Unsupervised
  - Semi-Supervised

- From the selection strategy perspective:
  - Wrapper methods
  - Filter methods
  - Embedded methods

Supervised Feature Selection

- Supervised feature selection is often for classification or regression

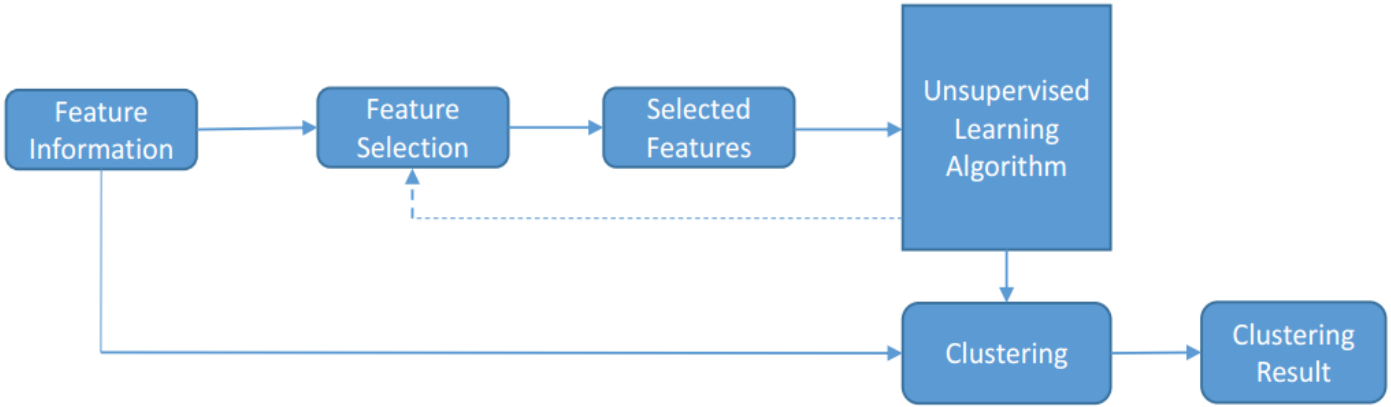
- Find discriminative features that separate samples from different classes (classification) or approximate target variables (regression)



Unsupervised Feature Selection

- Label information is expensive to obtain which requires both time and efforts

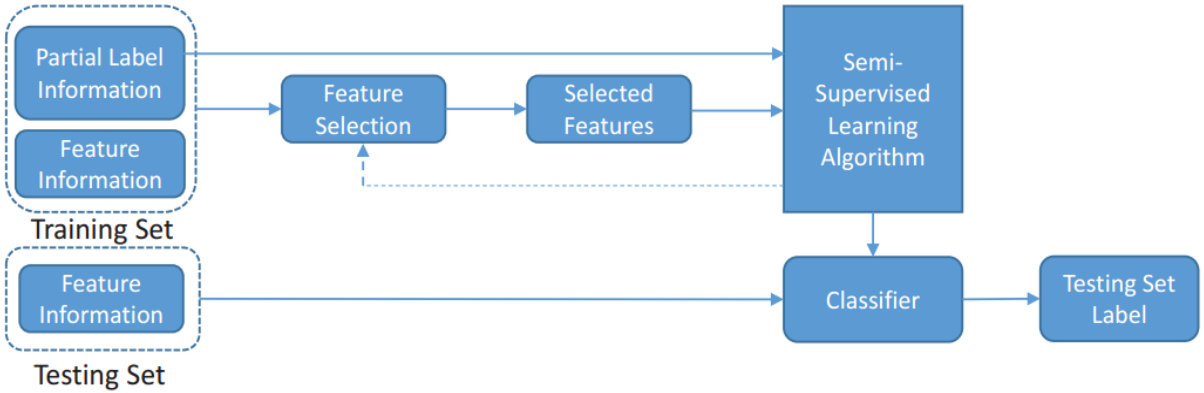
- Unsupervised methods seek alternative criteria to define the relevance of features



Semi-Supervised Feature Selection

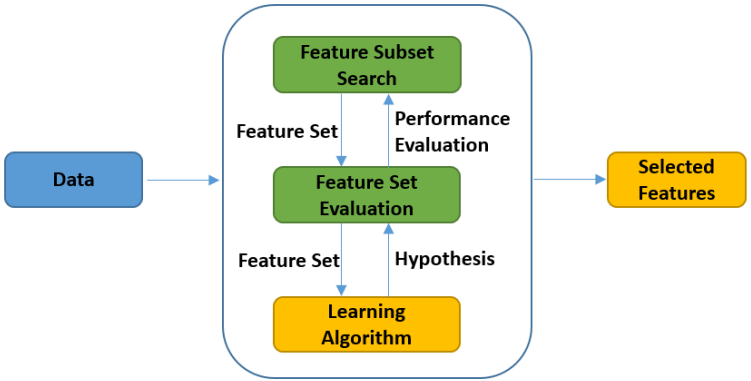
- We often have a small amount of labeled data and a large amount of unlabeled data

- Semi-supervised methods exploit both labeled and unlabeled data to find relevant features



Wrapper Methods

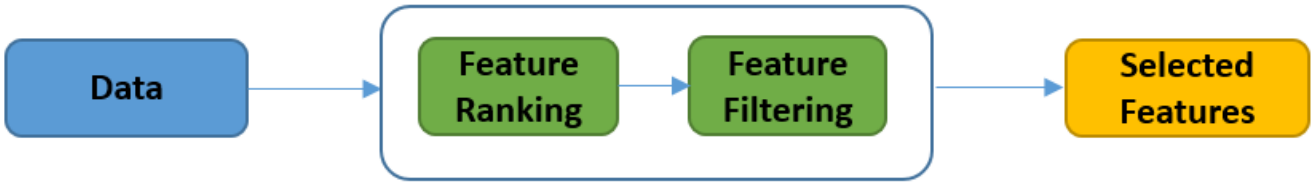
- Rely on the predictive performance of a predefined learning algorithm to assess features
- Repeat until some stopping criteria are satisfied
- Achieve high accuracy for a particular learning method
- Computational expensive (worst case search space is 2^d), some typical search strategies are sequential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



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

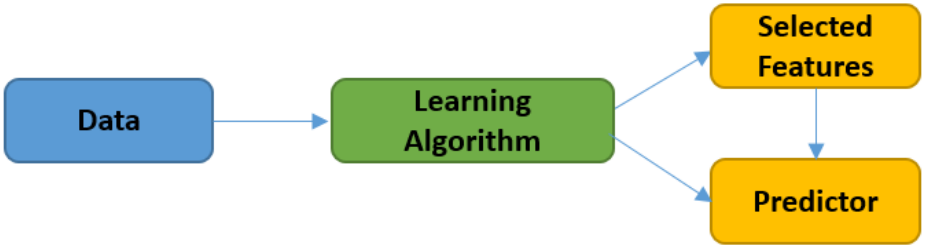
Filter Methods

- 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

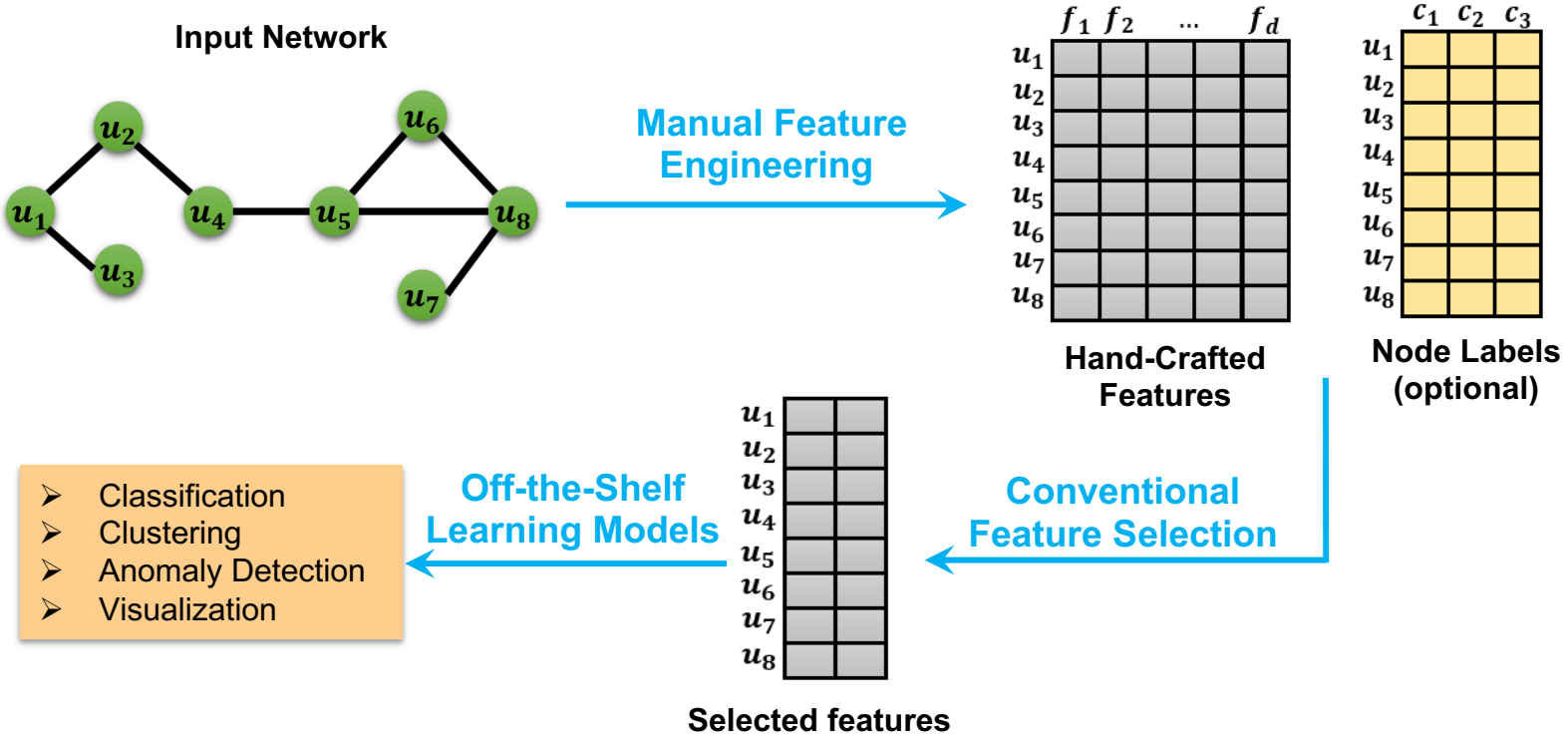
- A trade-off between wrapper and filter methods by embedding feature selection into the model learning, e.g., ID3



- Inherit the merits of wrapper and filter methods
  - Include the interactions with the learning algorithm
  - More efficient than wrapper methods
- Like wrapper methods, they are biased to the underlying learning algorithms

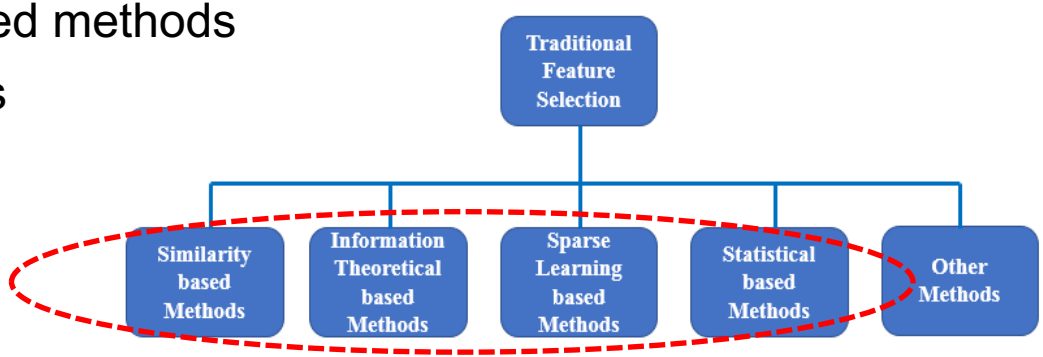
How to perform feature selection on plain networks?

Scenario 1: w/o Explicit Node Features



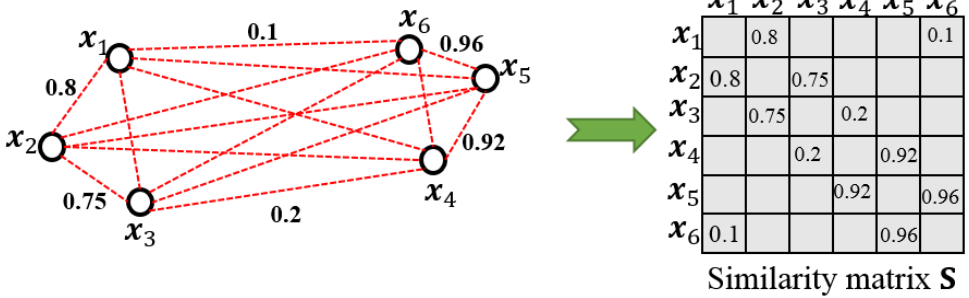
Conventional Feature Selection Algorithms

- One way to group conventional feature selection algorithms is according to the adopted techniques
  - Similarity based methods
  - Information theoretical based methods
  - Sparse learning based methods
  - Statistical based methods
  - Other methods



Similarity based Methods - Similarity Matrix

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



	x_1	x_2	x_3	x_4	x_5	x_6
x_1		0.8				0.1
x_2	0.8		0.75			
x_3		0.75		0.2		
x_4			0.2		0.92	
x_5				0.92		0.96
x_6	0.1				0.96	

Similarity matrix **S**

- E.g., w/o class labels, it can be defined by the RBF kernel

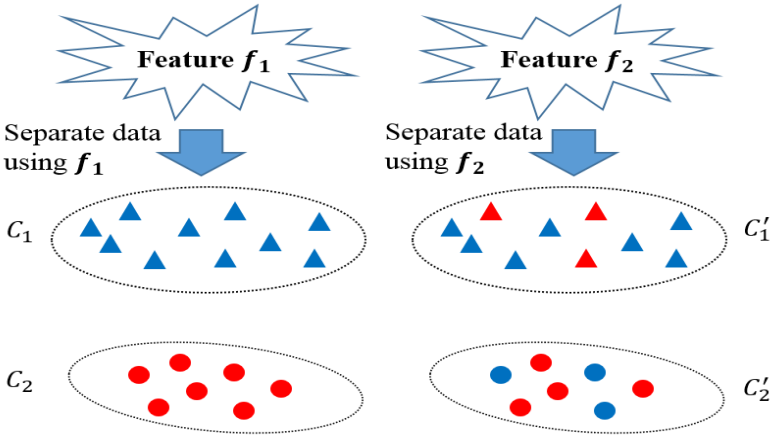
$$\mathbf{S}_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$$

- E.g., w/ class labels, it can be obtained as

$$\mathbf{S}_{ij} = \begin{cases} \frac{1}{n_l} & \text{if } y_i = y_j = l \\ 0 & \text{otherwise} \end{cases}$$

Similarity based Feature Selection

- Similarity based methods assess the importance of features by their ability to preserve data similarity
  - A good feature should ***not randomly*** assign values to data instances
  - 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 - 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:

Utility function $U(.)$: how well the feature set preserves the data similarity structure

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

utility of feature set $\mathcal{S}$

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 $\hat{\mathbf{S}}$

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: $1 - \left(\frac{\tilde{\mathbf{f}}_i}{||\mathbf{D}^{\frac{1}{2}} \tilde{\mathbf{f}}_i||} \right)' \mathbf{S} \left(\frac{\tilde{\mathbf{f}}_i}{||\mathbf{D}^{\frac{1}{2}} \tilde{\mathbf{f}}_i||} \right)$

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

$$\mathbf{S}^w_{i,j} = \begin{cases} 1/n_l & \text{if } y_i = y_j = l \\ 0 & \text{otherwise} \end{cases} \qquad \mathbf{S}^b_{i,j} = \begin{cases} 1/n - 1/n_l & \text{if } y_i = y_j = l \\ 1/n & \text{otherwise} \end{cases}$$

- 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

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

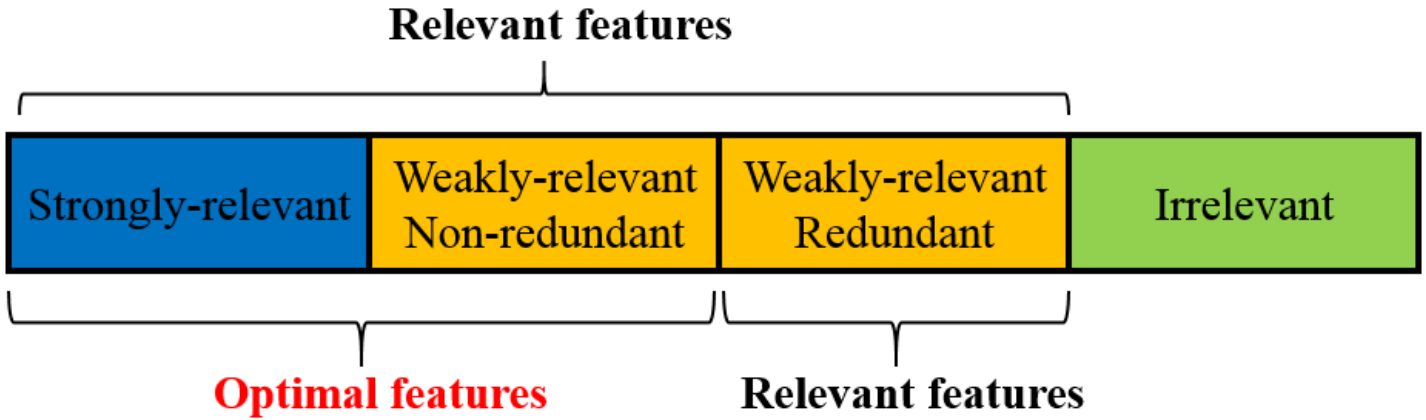
Laplacian matrix obtained from $\mathbf{S}^w$ and $\mathbf{S}^b$

- Fisher Score can be calculated from Laplacian Score

$$fisher\_score(f_i) = 1 - \frac{1}{laplacian\_score(f_i)}$$

Information Theoretical based Methods

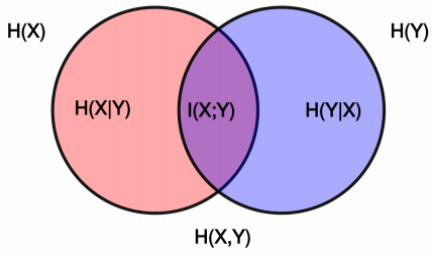
- Exploit different heuristic filter criteria to measure the importance of features



- Our target is to find these “optimal” features

Information Theoretical Measures

- Entropy of a discrete variable X $$H(X) = -\sum_{x_i \in X} P(x_i) log(P(x_i))$$
- Conditional entropy of X given Y $$H(X|Y) = -\sum_{y_j \in Y} P(y_j) \sum_{x_i \in X} P(x_i|y_j) log(P(x_i|y_j))$$
- Information gain between X and Y $$I(X;Y) = H(X) - H(X|Y)$$ $$= \sum_{x_i \in X} \sum_{y_j \in Y} P(x_i, y_j) log \frac{P(x_i, y_j)}{P(x_i) P(y_j)}$$



- Conditional information gain $$I(X;Y|Z) = H(X|Z) - H(X|Y,Z)$$ $$= \sum_{z_k \in Z} P(z_k) \sum_{x_i \in X} \sum_{y_j \in Y} P(x_i, y_j|z_k) log \frac{P(x_i, y_j|z_k)}{P(x_i|z_k) P(y_j|z_k)}$$

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:
    $$I(f_i; Y)$$
  - Minimize the redundancy w.r.t. selected features in $\mathcal{S}$:
    $$\sum_{f_j \in \mathcal{S}} I(f_j; f_k)$$
  - Maximize its complementary info w.r.t. selected features in $\mathcal{S}$:
    $$\sum_{f_j \in \mathcal{S}} I(f_j; f_k | Y)$$

Information Theoretic based Methods - Framework

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

$g(*)$: a function

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

- 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

$$score(f_k) = I(f_k; Y)$$

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

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

Min. Redundancy Max. Relevance [Peng et al., 2005]

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

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

reduced effect of feature redundancy

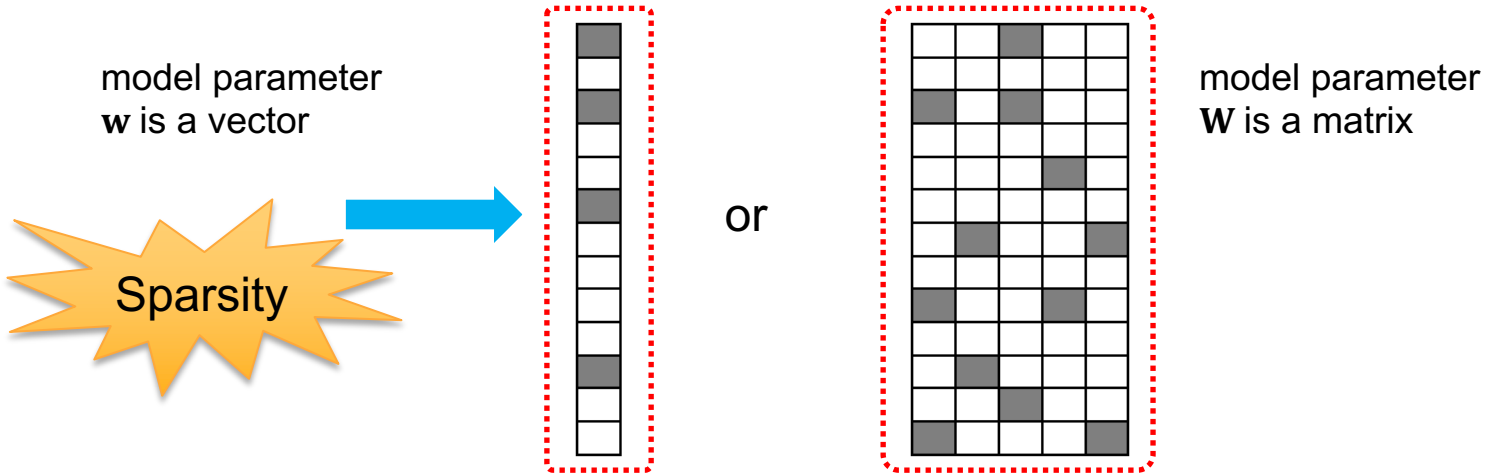
- MRMR is also a special case of the linear function by setting $\lambda = 0$ and adjusting β adaptively

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?

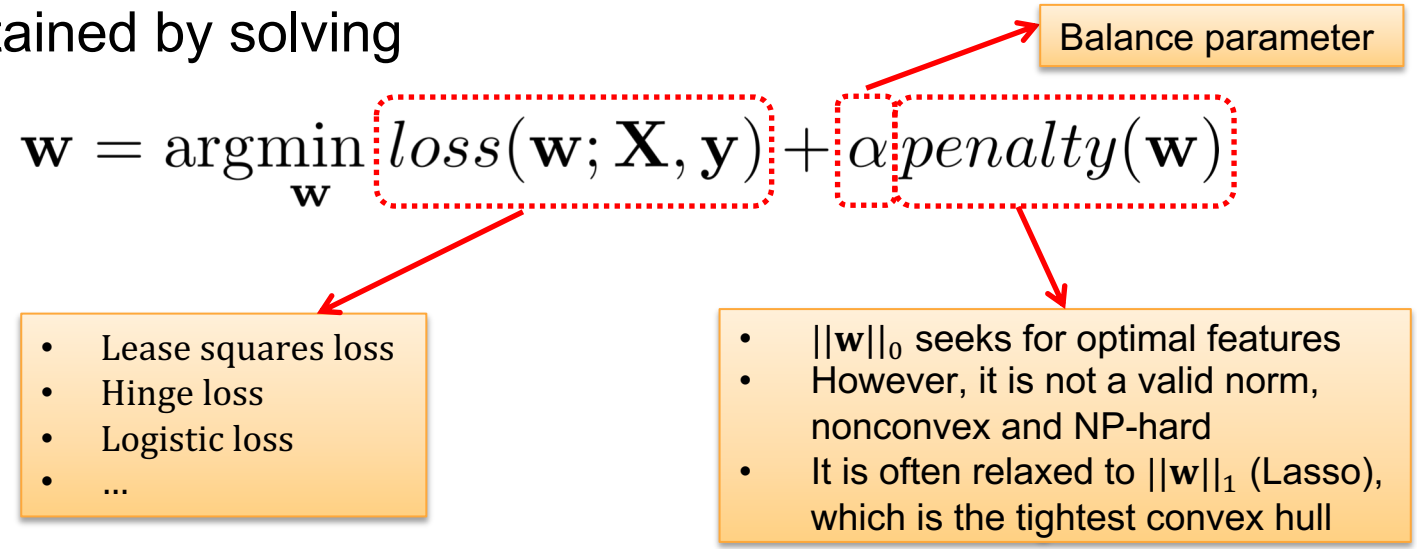
- The model parameters in many data mining tasks can be represented as a vector $\mathbf{w}$ or a matrix $\mathbf{W}$
- Sparsity indicates that many elements in $\mathbf{w}$ and $\mathbf{W}$ are small or exactly zero



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

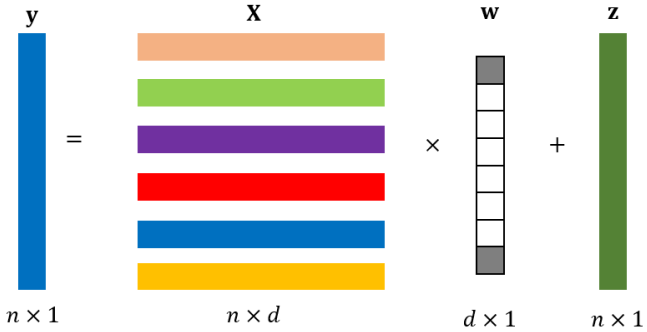


Lasso [Tibshirani, 1996]

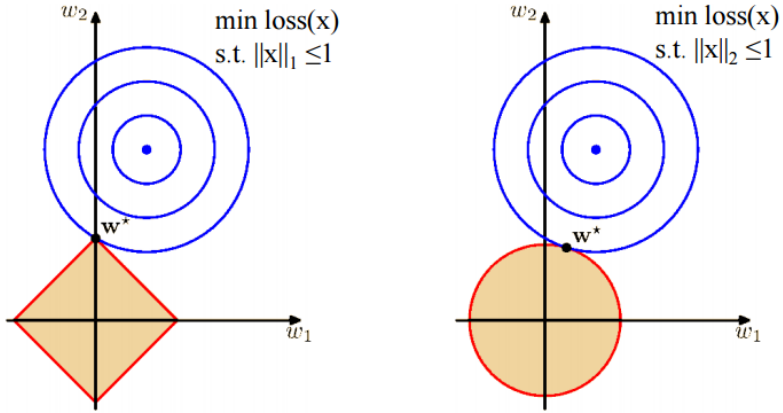
- Based on ℓ_1-norm regularization on weight $\mathbf{w}$

$$\min_{\mathbf{w}} loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha \|\mathbf{w}\|_1$$

- In the case of least square loss with offset value, it looks like this …



The feature score of the i-th feature is $|\mathbf{w}_i|$; the higher the value, the more important the feature is



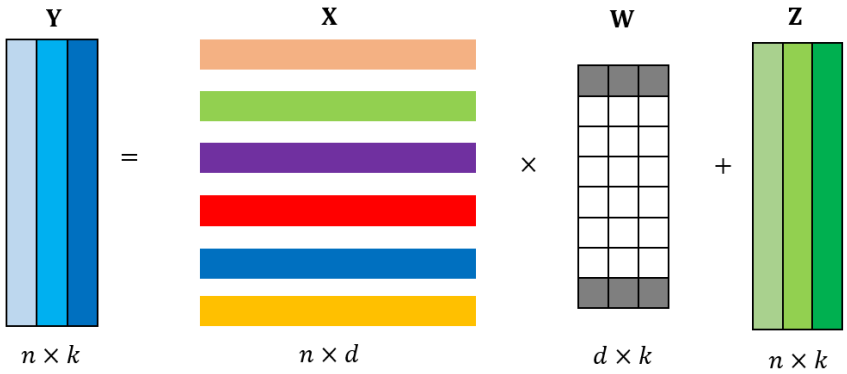
- Why ℓ_1-norm Induces Sparsity?

Extension to Multi-Class or Multi-Variate Problems

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

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

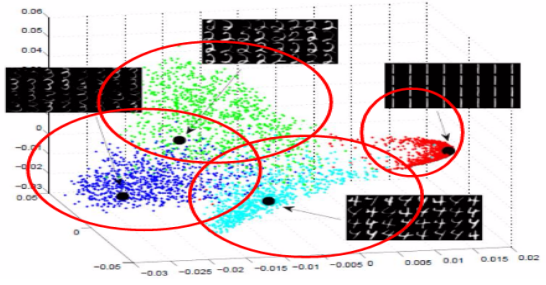
- $||\mathbf{W}||_{2,1}$ achieves joint feature sparsity across multiple targets
- In the case of least square loss with offset, it looks like this …



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



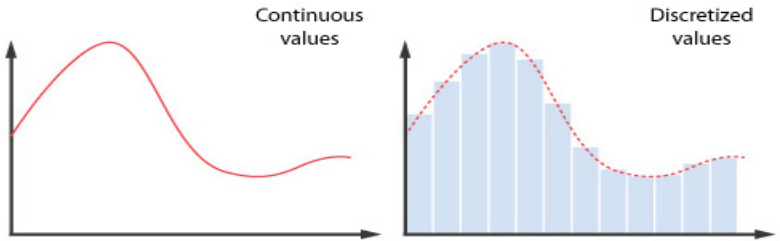
- There are two options
  - Obtain clusters and then perform FS (e.g., MCFS)
  - Embed FS into clustering (e.g., NDFS)

- 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

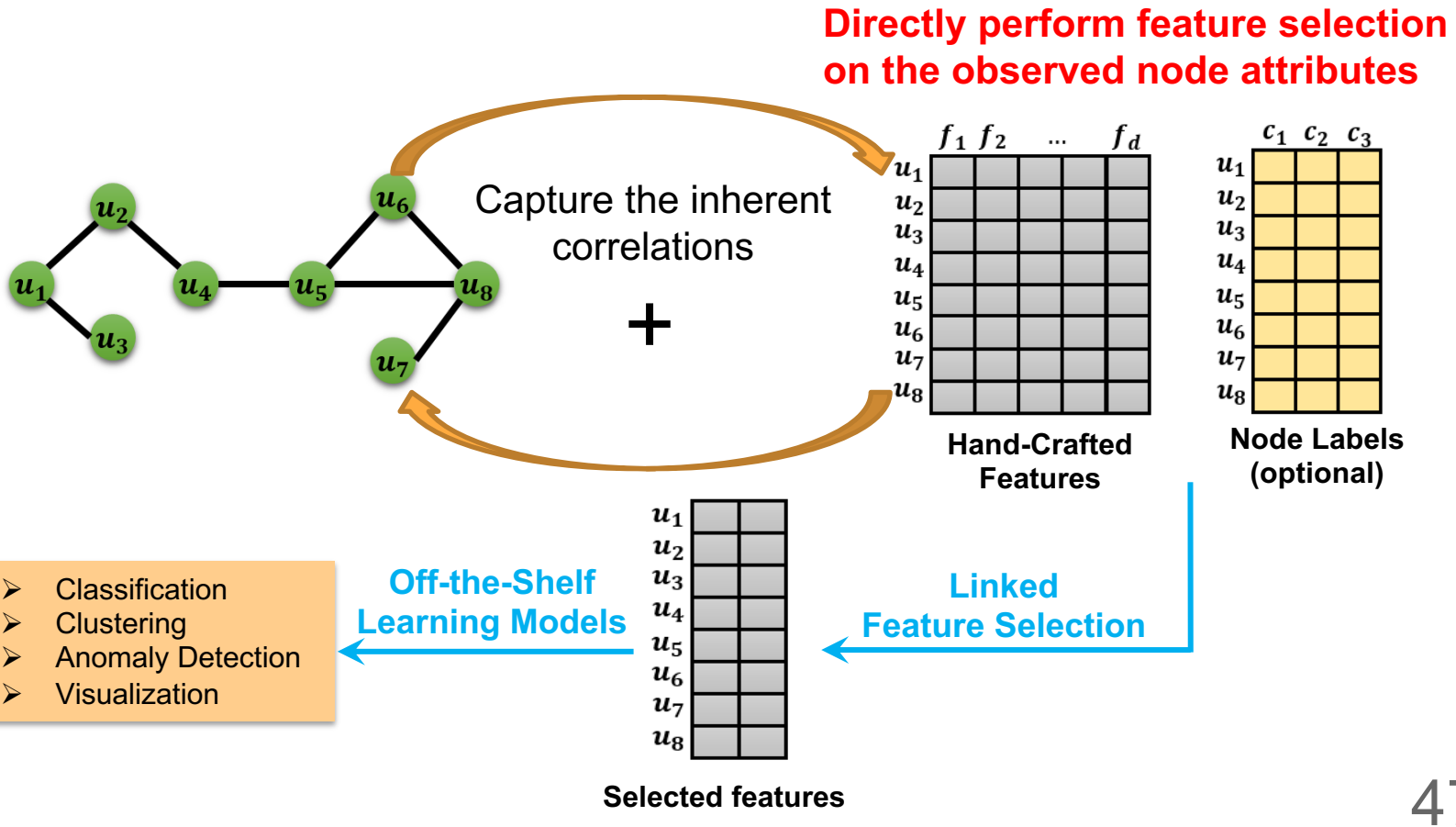
Other Types of Methods

- Reconstruction based Feature Selection
  - Minimize reconstruction error of data with selected features
  - Reconstruction function can be both linear and nonlinear

- Hybrid Feature Selection
  - Construct a set of different feature selection results
  - Aggregate different outputs into a consensus result

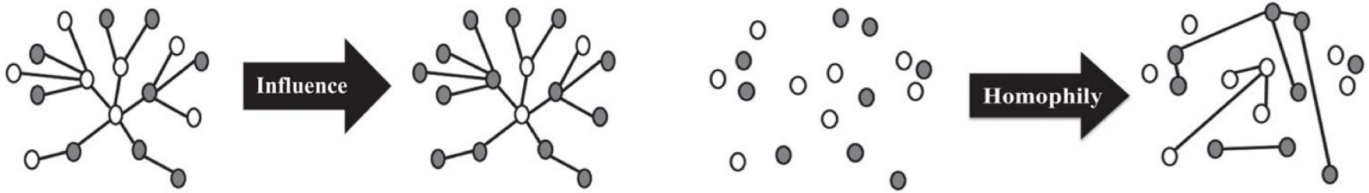
How to perform feature selection on attributed networks?

Scenario 2: w/ Explicit Node Features

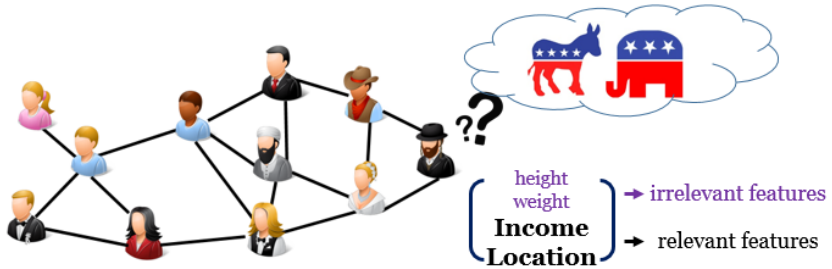


Why Linked Feature Selection?

- 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
  - Collective 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 $\mathbf{Y}$

Joint feature sparsity

Avoid overfitting

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

- Employ graph regularization to model links

undirected network

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

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

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

directed network

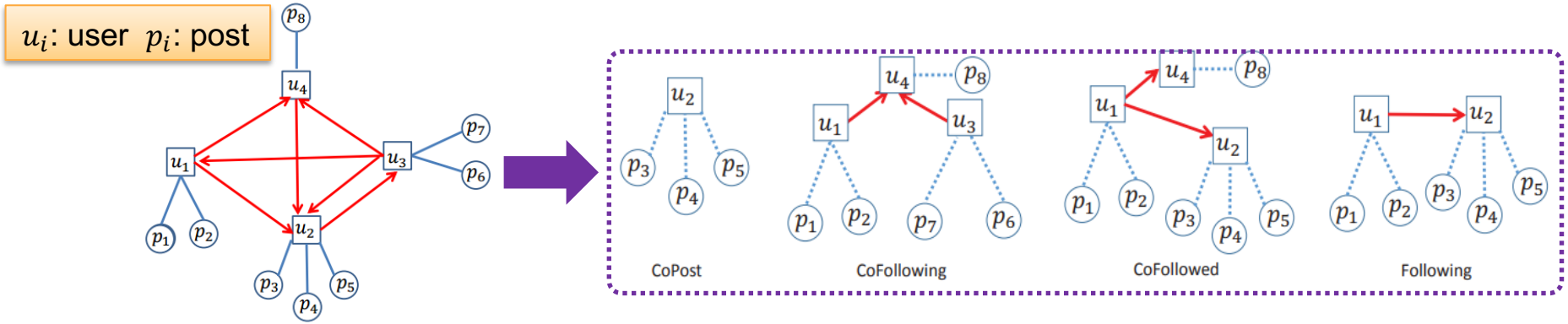
P: transition matrix of random walk
Π: stationary distribution

- 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



- These social relations are supported by social theories (Homophily and Social Influence)

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

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

CoPost hypothesis

CoPost relations

- Observations

Table 1: Statistics of the Datasets

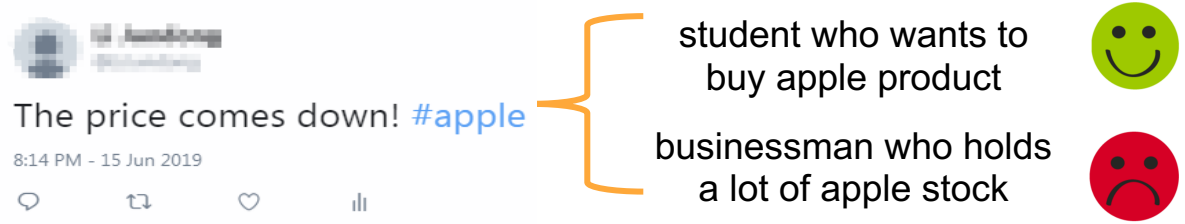
	BlogCatalog	Digg
# Posts	7,877	9,934
# Original Features	84,233	12,596
# Features after *TFIDF*	13,050	6,544
# Classes	14	15
# Users	2,242	2,561
# Following Relations	55,356	41,544
Ave # Posts	3.5134	3.8790
Max # Followers	820	472
Min # Followers	1	1
Network Density	0.0110	0.0063
Clustering Coefficient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

Table 5: Classification Accuracy Improvement of the Proposed Methods

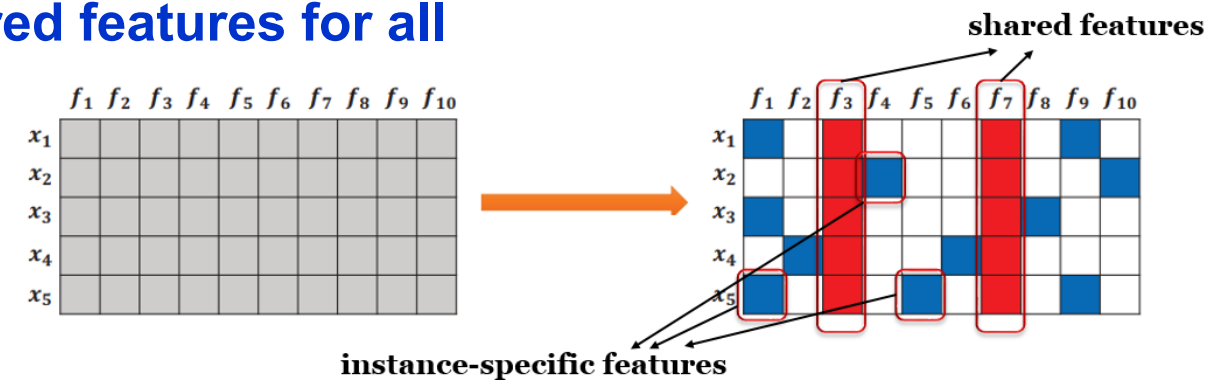
Improvement in Digg(%)				
Datasets	CP	CFI	CFE	FI
$\mathcal{T}_5$	+14.54	+7.01	+4.69	+15.25
$\mathcal{T}_{25}$	+4.59	+1.59	0	+4.02
$\mathcal{T}_{50}$	+7.19	+3.92	+1.05	+8.48
$\mathcal{T}_{100}$	+4.90	+3.15	+1.63	+4.64
Improvement in BlogCatalog(%)				
Datasets	CP	CFI	CFE	FI
$\mathcal{T}_5$	+10.71	+7.89	+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]

- Content information are highly idiosyncratic



- 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 FS [Li et al., 2017]

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

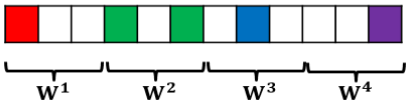
$$\min_{\tilde{\mathbf{W}},\mathbf{W}^i} \sum_{i=1}^{n} \|\mathbf{x}_i(\tilde{\mathbf{W}}+\mathbf{W}^i)-\mathbf{y}_i\|_2^2+\beta\sum_{i=1}^{n}\|\mathbf{W}^i\|_{2,1}^2+\gamma\|\tilde{\mathbf{W}}\|_{2,1}$$

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 a network lasso penalty term 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 FS Methods

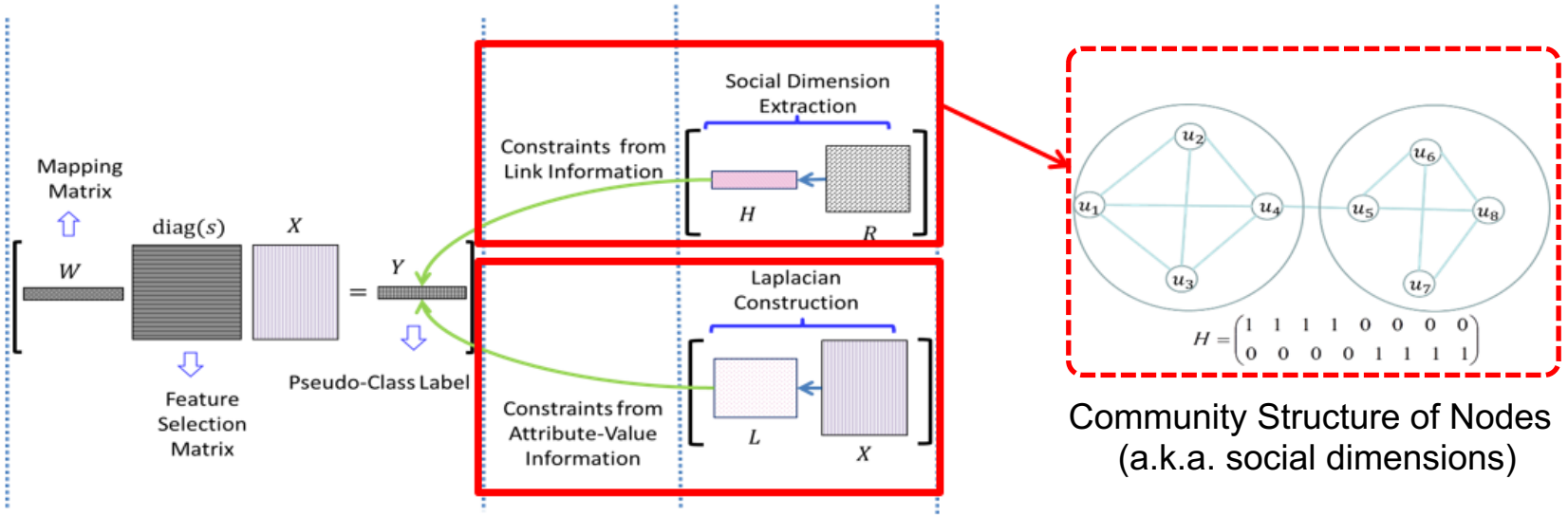
- Classification on Cora

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



Community Structure of Nodes (a.k.a. social dimensions)

LUFS [Tang and Liu, 2012]

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

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

Pseudo labels

Weighted social dimension matrix

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

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

- Attribute similarity should be consistent of pseudo label similarity

$$\min tr(\mathbf{Y}'\mathbf{L}\mathbf{Y})$$

similarity matrix using RBF

- Objective function of LUFS

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

$$\text{s.t.}\ \mathbf{s} \in \{0,1\}^d,\ \mathbf{s}'\mathbf{1} = k,$$

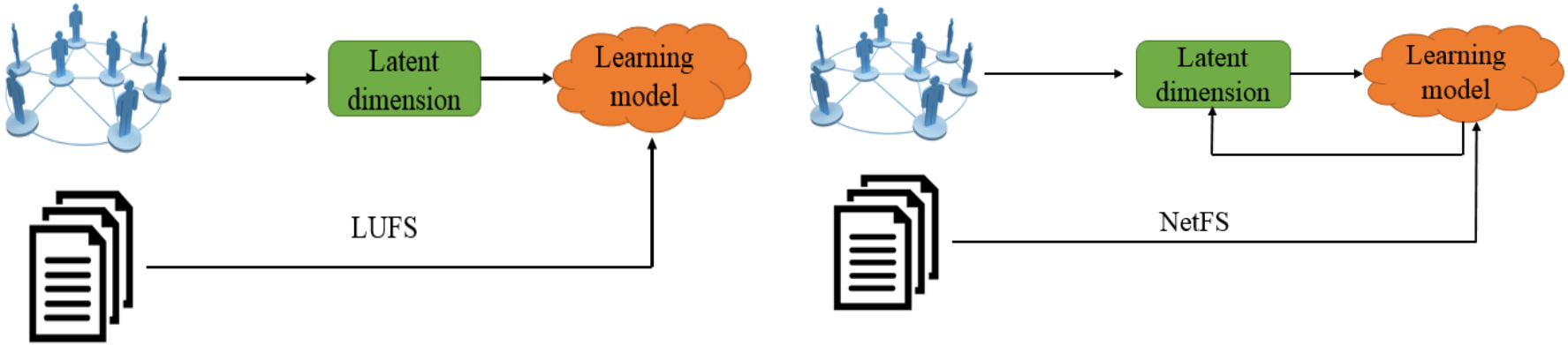
$$\mathbf{Y} = \mathbf{W}'\text{dig}(\mathbf{s})\mathbf{X}$$

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

NetFS [Li et al., 2016]

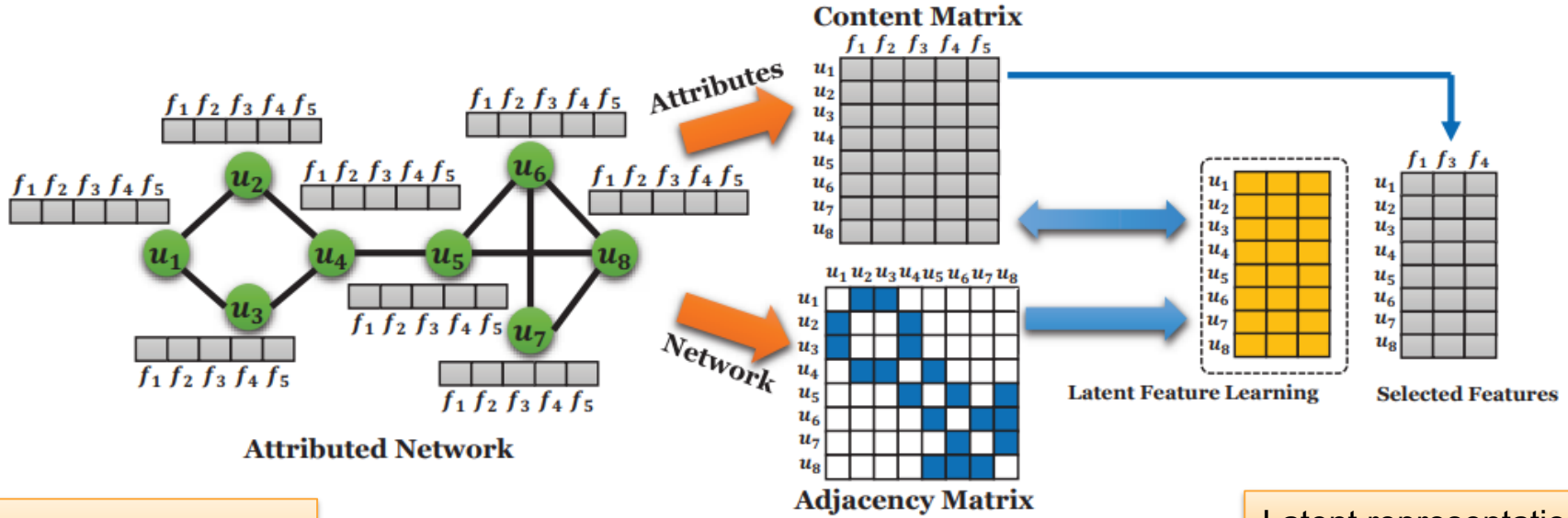
- LUFS performs network structure modeling and feature selection separately

- 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

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

Joint feature sparsity

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!

High-dimensional Features

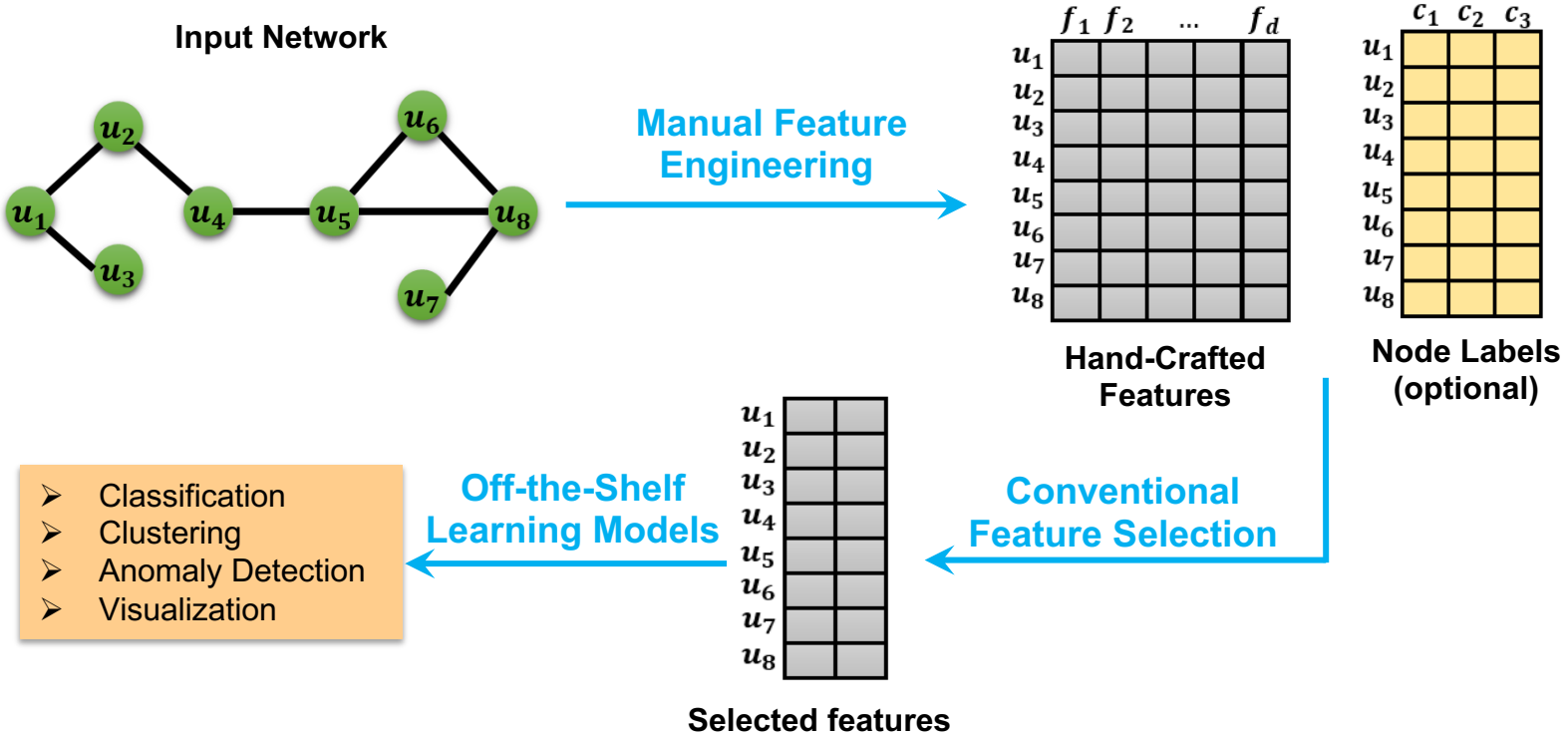
The features of nodes are often in a high-dimensional feature space

- Scenario 1: w/o explicit node features
  - Manual feature engineering could generate **a large number** of features
  - Not clear what features could be useful for learning on graphs

- Scenario 2: w/ explicit node features
  - Observed node 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***

How to Perform FS w/o Explicit Node Features?



How to Perform FS w/ Explicit Node Features?