The Chinese University of Hong Kong, Shenzhen

DDA 4210

Advanced Machine Learning

Report for Project

Author	Student ID
黎俊乐 Li, Junle	118010142
厉馨阳 Li, Xinyang	120090345
魏诗云 Wei, Shiyun	120090564
曾悦清 Zeng, Yueqing	120090398

Code links:

`https://github.com/Linnore/GNN-Cora-CUHKSZAG`

`https://github.com/Linnore/CUHKSZ_AcademicGraph`

May 1, 2023

1 Introduction and Motivation

Graph neural networks (GNNs) have emerged as a powerful tool for analyzing graph-structured data. They have been applied to a wide range of tasks, including node classification and link prediction. In this project, we are particularly interested in the task of link prediction, for it could be a way to provide effective recommendations for potential node affinities. Inspired by the academic tool *Connected Paper*, we constructed an academic graph consisting of the publications of all professors and lecturers from the School of Data Science at the Chinese University of Hongkong, Shenzhen (CUHKSZ-AG dataset). With CUHKSZ-AG, we implement GCN and GraphSAGE models to do node classifications and link prediction. The latter allows us to make a recommendation system for within-SDS academic publications. Students of SDS can explore their research interests with the aid of our model, and find potential academic advisors given the with-SDS academic publication's recommendations from our model.

2 Data Setting and Processing

Data Setting The academic graph can be described as $G(V, E)$. where V is the set of academic publications (nodes), and E is the set of citation relationships that connect two publications (edges). Each $v \in V$ has its own feature vector $X_v \in \mathbb{R}^D$, where D is the dimension of node features.

Datasets We used two datasets in our project: the CUHKSZ-AG dataset we created and the benchmark dataset Cora. 1. The CUHKSZ-AG dataset. It consists of 6614 papers, and each paper is labelled into 8 categories according to the field of studies. A 768-dimension feature vector that embeds the paper's content is also attached to each paper (node). The embedding vectors are obtained through *Semantic Scholar*'s API and constructed by the Natural Language Processing model Specter which aims to extract content-based embedding of scientific papers. 2. The Cora dataset, which consists of 2,708 scientific publications and 10556 citation linkages among them. Each publication has a 1433-dimension bag-of-keywords 01 feature vector.

CUHKSZ-AG Dataset Construction To construct the academic graph of SDS's professors and lecturers, we first applied web crawling on SDS's offical website with the help of *BeautifulSoup* python library. After obtaining a table of scholars' names and with their publications, we use APIs of *Semantic Scholar* to get their unique ID in *Semantic Scholar*'s database. To distinguish the exact scholar of our SDS from scholars with the same name around the world, publications' titles returned by *Semantic Scholar*'s API are used to match the ones crawled on SDS's website. Then, publications of SDS's scholars are requested through *Semantic Scholar*'s API and we managed to obtained titles, reference list, content-based embedding vectors, and other information we need.

After tedious data cleaning, e.g., deleting the isolated papers (no citation relationship within the dataset), our CUHKSZ-AG dataset is now published on `https://github.com/Linnore/CUHKSZ_AcademicGraph`. Following the instructions of this repo, one can easily import CUHKSZ-AG dataset in the standard format of *torch_geometric.data.Dataset* with two lines of python codes.

3 Task description

Though the ultimate purpose of this project is to implement a method for paper recommendation given the academic graph, we accomplish firstly the paper classification task to practice our skills for GCN/GraphSAGE model implementations, and to show that our dataset CUHKSZ-AG is informative and helpful for practicing GNNs implementations.

Node Classification Given a graph $G(V, E)$ where each $v \in V$ has a feature vector $x_v \in \mathbb{R}^D$, node classification is to predict the label y_v of a node $v \in V$ given x_v for $v \in V$ and the edge set E.

Link Prediction Given a graph $G(V, E)$ where each $v \in V$ has a feature vector $x_v \in \mathbb{R}^D$, link prediction is to predict the probability of the existence of each edge $e = (u, v)$ where $u, v \in V$.

4 Methodology

GNN Model Design for Node Classification Given a graph $G(V, E)$ where each $v \in V$ has a feature vector $x_v \in \mathbb{R}^D$, define $h_{i,v}$ as the node embedding for v after the i^{th} GNN layer. Denote the label of each node as $y_v \in [0, 1, ..., C-1]$, where C is the number of categories. Then, the forward path of a node classification GNN model for each $v \in V$ can be shown as

$$\mathbf{x_v} \in \mathbb{R}^D \underbrace{\longrightarrow \mathbf{h}_{1,v} \in \mathbb{R}^{H_1} \longrightarrow ... \longrightarrow}_{\text{k GCN/GraphSAGE layers}} \mathbf{h}_{k,v} \in \mathbb{R}^{H_k} \xrightarrow{\text{Softmax}} \mathbf{f}_v \in \mathbb{R}^C \tag{1}$$

where the output is the node embedding e_v that contains the probability of the node's label in $\{0, 1, ..., C-1\}$, and $H_i, i = 1, ..., k$ is the size of the hidden node embedding vector. The loss for each node v is the negative log-likelihood: $nll\_loss(\mathbf{f}_v, y_v)$, and the model aims to $\min \sum_{v \in V} nll\_loss(\mathbf{f}_v, y_v)$.

GNN Model Design for Recommendation System To design GNN models for link prediction, we need to expand the set of notations as follows. Given a graph $G(V, E)$, we call it the positive graph $G^+(V, E^+)$ since the edges in this graph exist. Define the so-called negative graph $G^-(V, E^-)$ as the edge complement of G^+, that is the graph that consists of all non-existing edges. Further, define the edge label of $e \in E^+ \cup E^-$, $y_e = \mathbb{I}(e \in E^+)$, where $\mathbb{I}(\cdot)$ is the indicator function. The forward path of a GNN model for link prediction is very similar to in (1). For each $v \in V$,

$$\mathbf{x_v} \in \mathbb{R}^D \quad \underbrace{\longrightarrow \mathbf{h}_{1,v} \in \mathbb{R}^{H_1} \longrightarrow ... \longrightarrow}_{\text{k GCN/GraphSAGE layers using } G^+(V,E^+)} \quad \mathbf{h}_{k,v} \triangleq \mathbf{f}_v \in \mathbb{R}^{H_k}. \tag{2}$$

Note that the node embedding $\mathbf{f}_v$ is computed using only $G^+(V, E^+)$.

To design the loss function, we define the score of edge $e = (u, v)$ as $s_e = sigmoid(f_u^T f_v)$, where $u, v \in V$. The higher the score is, the higher the probability that the corresponding edge exists. To train the model's ability for link prediction, we expect that the model can predict $\hat{y_e} = 1$ for $e \in E^+$ and $\hat{y_e} = 0$ for $e \in E^-$. Therefore, the binary cross entropy can be used as the loss for each edge, and the model aims to $\min \sum_{e \in E^+ \cup E^-} binary\_cross\_entropy(s_e, y_e)$.

Notations for Layer Design

- The notation for the k-layer GCN model's layer design hereinafter will be

$$GCN[\text{dropout\_rate}|H^1, H^2, ..., H^k] \tag{3}$$

  where the first parameter denotes the dropout rate before conducting graph convolution, and H^i is the size of the hidden node embedding vector after i^{th} GCN layer.

- The notation for the k-layer GraphSAGE model's layer design hereinafter will be

$$GraphSAGE[\text{aggregator\_type}|H^1, H^2, ..., H^k] \tag{4}$$

  where the first parameter denotes the aggregator type used in all GraphSAGE layers, and H^i is the size of the hidden node embedding vector after i^{th} GraphSAGE layer.

Data Split for Node Classification Notice that the academic graphs of Cora and CUHKSZ-AG are sparse graphs with numbers of nodes less than 10000; therefore, we can feed the model with the entire graph without mini-batching techniques by sampling sub-graphs. The data split we refer to here is therefore for loss computation only. We apply data split by randomly splitting V into $V_{train}, V_{val}, V_{test}$. The training, validation, and testing loss is then defined as $\sum_{v \in V} nll\_loss(\mathbf{f}_v, y_v)$ for $V_{train}, V_{val}, V_{test}$ respectively. By such design, only the embedding of $v \in V_train$ participates in the backward propagation, yet the embedding of all $v \in V$ is simultaneously learned.

Mini-batching Training of GNN Models for Link Prediction In the task of link prediction, we need to gather loss from edges in the dense graph G^- where $|E^-| \approx |V|^2$. Therefore, it is impossible to feed the model with the entire G^- for loss computation. To solve this problem, we adopt mini-batching techniques by constructing sub-graphs of G^- through neighborhood sampling (see Algorithm 2 in the paper the presents GraghSAGE Inductive Representation Learning on Large Graphs). From now on, we denote the mini-batch i of G^- as $G^-(V, E_i^-)$.

Data Split for Link Prediction In this project, we adopt two ways of data split strategies to train GraphSAGE models for link prediction.

- Random Edge Split for Transductive Learning: Given a graph $G(V, E)$, we randomly select a subset of E as E_{train}, then define $G_{train}(V, E_{train})$ as the training graph. G^+_{train} and G^-train are both defined according to the above random link split. Then the training loss of the mini-batch i is defined as $\sum_{e \in E^+_{train} \cup E^-_{i,train}} binary\_cross\_entropy(s_e, y_e)$. Note that all the edges in G^+_{train} are used and the mini-batching is only for the dense G^-. The validation loss and testing loss of each mini-batch are defined similarly.

  This random link split strategy should best cooperate with transductive learning since the model would learn and update all nodes' embedding though the backward propagation only involves the training edges.

- Graph Separation for Inductive Learning: Given a graph $G(V, E)$, we first apply random node split and obtain V_{train}, V_{val}, and $Vtest$. Then we define

$$\widetilde{E}_{train} = \{e = (u, v) \in E : u \in V_{train} \ and \ v \in V_{train}\} \tag{5}$$

$$\widetilde{E}_{val} = \{e = (u, v) \in E : u \in V_{val} \ or \ v \in V_{val}\} \tag{6}$$

$$\widetilde{E}_{test} = \{e = (u, v) \in E : u \in V_{test} \ or \ v \in V_{test}\} \tag{7}$$

  and hence $\widetilde{G}_{train}$, $\widetilde{G}_{val}$, and $\widetilde{G}_{test}$. The corresponding loss definitions follow the same convention, for instance, the training loss at mini-batch i is $\sum_{e \in \widetilde{E}^+_{train} \cup \widetilde{E}^-_{i,train}} binary\_cross\_entropy(s_e, y_e)$.

  This strategy separates a training graph from the original graph by a cut; therefore, it is suitable for inductive learning in which the model would not learn the fixed embedding of any node not presented in the training graph but yet learn the way to compute its embedding.

Recommendation for Potential Links For a node of interest, say u, the essential idea to recommend another node v to u is to detect $e = (u, v) \in E^-$ with high score $s_e = sigmoid(f_u^T f_v)$, where f_u and f_v are the embedding of node u and v. The top-k recommendation is, therefore, the k v_i's such that edge $(u, v_1), ..., (u, v_k)$ have the highest scores among $(u, v) \in E^-$.

- For Transductive Model Trained by Random Edge Split: Input G^+_{train} to the model to get f_v for all $v \in V$. Then follow the procedures above. Note that the node of interest, say u, must satisfy $u \in V$, where V is the set of nodes already provided to train the model.
- For Inductive Model Trained by Graph Separation: For a node of interest, say u, prepare a graph $G'(V', E')$ that contains u. Input G' to the model to get f_v for all $v \in V'$. Then follow the procedures above. Note that if u is already presented in the graph $G+$ we feed the model, then input $G+$ already suffices. If not, G' can be the augmented graph by union G with u and u's edges.

Miscellaneous

- Optimizer: We use Pytorch's Adam optimizer with tuned learning rates for all the models.
- Activation Function: We use *ReLU* as the activation function between GCN/GraphSAGE layers.
- Early Stopping: we monitor the validation loss with patience = 10 epochs to avoid overfitting.

5 Numerical Results and Discussion

5.1 Node Classification

Dataset	Best Model	Test Accuracy
CUHKSZ-AG	$GCN[0.3\|16, 8]$	75.66%
Cora	$GCN[0.5\|16, 8]$	86.6%
CUHKSZ-AG	$GraphSAGE[max\|112, 16]$	76.11%
Cora	$GraphSAGE[mean\|112, 16]$	86.5%

5.2 Recommendation Results

Example test accuracy of our models on CUHKSZ-AG datasets.

Dataset	Model	Method	Test Accuracy
CUHKSZ-AG	$GCN[0\|112, 16]$	transductive	97.38%
CUHKSZ-AG	$GraphSAGE[mean\|112, 16]$	transductive	98.69%
CUHKSZ-AG	$GraphSAGE[mean\|112, 16]$	inductive	97.19%

Example recommendation results of selected papers are in Appendix. One can also follow the notebooks at the root of our repo to get recommendation results of any paper in our dataset.

Discussion

- **Accuracy: AUC Score(area below ROC curve)** The test accuracy of our model is larger than 0.97, which makes sense for the feasibility of the recommendation system. And for the GraphSAGE model, transductive accuracy is higher than the inductive one for the recommendation of a node existing in the training graph (e.g. the paper of Prof. Milzarek), which is a trade-off between accuracy and inductiveness.
- **Recommendation Result** We use a paper by Prof. Jicong Fan and a paper by Prof. Andre Milzarek as example results of our models' recommendation. See (table 1,2,3, 4, 5, 6). From the titles of the recommended paper, it seems that the recommended results are highly related to the original paper. Such as *A Semi-smooth Newton Stochastic Proximal Point Algorithm with Variance Reduction*, all recommended papers are related to *optimization*.

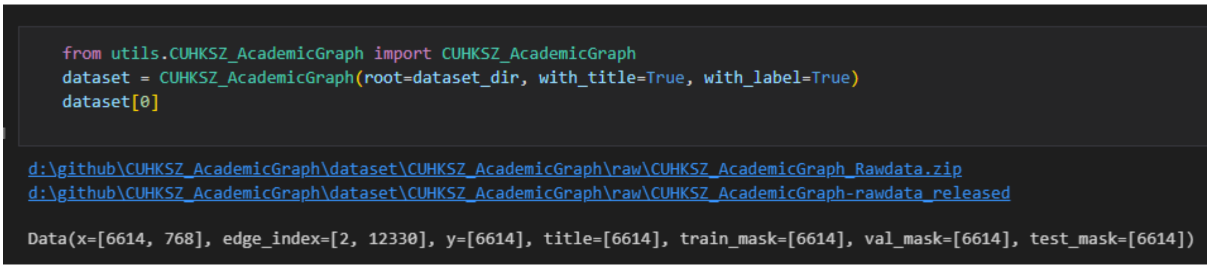
  Meanwhile, the three models coincide and recommend some common results, which makes our recommendation results more convincing.
- **Limitations** However, for a specific classification under a large theme, for example, the second-order method focused by Prof. Milzarek's paper (see in table 4, 5, 6)), our model failed to recommend papers that are also related to second-order methods though the recommendation results are related to the broader theme–optimization. Possible reasons for this limitation are: 1) the node features failed to capture specific key content related to the sub-topic; 2) our dataset is too small and fails to diverse these sub-topics under a large theme.

6 Significance and Novelty

- We constructs a public academic graph dataset CUHKSZ-AG using the publications of the professors and lecturers from the School of Data Science in CUHKSZ, which can be applied to the tasks of node classification and linkage prediction. This dataset can be helpful for GNN beginners in CUHKSZ.
- Our results provide empirical support for implementing GNN-based recommendation system based on academic graph.
- (Engineering perspective) Our pipelines implement GCN and GraphSAGE with the newest Pytorch 2.0, Pytorch Lightning 2.0, and other useful tools like tensorborad. There are very limited materials online for these newly updated libraries to implement GNN, especially for Pytorch Lightning. We believe publishing our pipelines in the Pytorch Lightning community can be useful for beginners of GNN and the Pytorch Lightning framework.

7 Appendix

To load our dataset with 2 lines of codes:



```
from utils.CUHKSZ_AcademicGraph import CUHKSZ_AcademicGraph
dataset = CUHKSZ_AcademicGraph(root=dataset_dir, with_title=True, with_label=True)
dataset[0]
```

```
d:\github\CUHKSZ_AcademicGraph\dataset\CUHKSZ_AcademicGraph\raw\CUHKSZ_AcademicGraph_Rawdata.zip
d:\github\CUHKSZ_AcademicGraph\dataset\CUHKSZ_AcademicGraph\raw\CUHKSZ_AcademicGraph-rawdata_released

Data(x=[6614, 768], edge_index=[2, 12330], y=[6614], title=[6614], train_mask=[6614], val_mask=[6614], test_mask=[6614])
```

Figure 1: To load CUHKSZ-AG dataset

To visualize the training and validation loss during the training process of recommendation system:

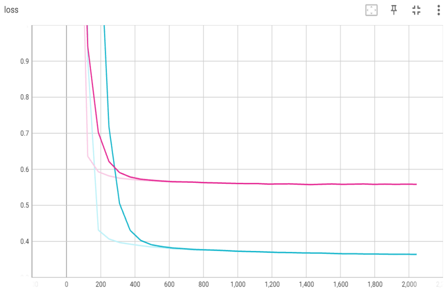


Figure 2: Loss curve of GCN model for our Academic Graph

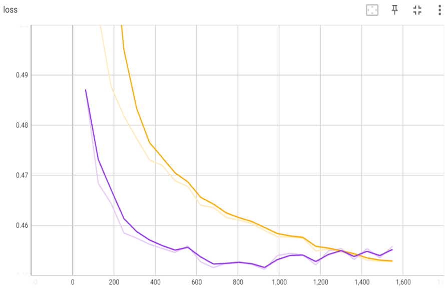


Figure 3: Loss curve of GraphSAGE model for our Academic Graph with inductive method

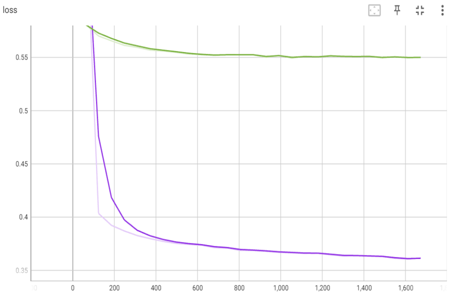


Figure 4: Loss curve of GraphSAGE model for our Academic Graph with transductive method

Examples for recommendations:

Table 1: Recommendation result for Prof. Jicong Fan's paper: *Non-linear matrix completion* by GCN using transductive method

Scores	Title
0.999998	*R1-PCA: rotational invariant L1-norm principal component analysis for robust subspace factorization*
0.999976	*Two-dimensional PCA: a new approach to appearance-based face representation and recognition*
0.999972	*Convex and Semi-Nonnegative Matrix Factorizations*
0.999921	*Efficient and Robust Feature Selection via Joint 2, 1-Norms Minimization*
0.999827	*Factor Group-Sparse Regularization for Efficient Low-Rank Matrix Recovery*
0.999821	*Orthogonal nonnegative matrix t-factorizations for clustering*
0.999815	*On the Equivalence of Nonnegative Matrix Factorization and Spectral Clustering*
0.999636	*Robust nonnegative matrix factorization using L21-norm*
0.999580	*Matrix Completion via Sparse Factorization Solved by Accelerated Proximal Alternating Linearized Minimization*
0.999403	*RSP-Based Analysis for Sparsest and Least ℓ_1-Norm Solutions to Underdetermined Linear Systems*

Table 2: Recommendation result for Prof. Jicong Fan's paper: *Non-linear matrix completion* by GraphSAGE using inductive method

Scores	Title
0.999294	*Simultaneous tensor subspace selection and clustering: the equivalence of high order svd and k-means clustering*
0.997687	*R1-PCA: rotational invariant L1-norm principal component analysis for robust subspace factorization*
0.997050	*Two-dimensional PCA: a new approach to appearance-based face representation and recognition*
0.996697	*Exactly Robust Kernel Principal Component Analysis*
0.996125	*K-means clustering via principal component analysis*
0.995987	*Linearized cluster assignment via spectral ordering*
0.994392	*Polynomial Matrix Completion for Missing Data Imputation and Transductive Learning*
0.994122	*Matrix completion by deep matrix factorization*
0.993743	*Graph-Laplacian PCA: Closed-Form Solution and Robustness*
0.993690	*Robust Matrix Completion via Joint Schatten p-Norm and lp-Norm Minimization*

Table 3: Recommendation result for Prof. Jicong Fan's paper: *Non-linear matrix completion* by GraphSAGE using transductive method

Scores	Title
0.998610	*R1-PCA: rotational invariant L1-norm principal component analysis for robust subspace factorization*
0.997893	*Continuum Isomap for manifold learnings*
0.997892	*Theory of Compressive Sensing via 1-Minimization: a Non-RIP Analysis and Extensions*
0.997549	*Improved RIP-Based Bounds for Guaranteed Performance of Several Compressed Sensing Algorithms*
0.997527	*Matrix Completion via Sparse Factorization Solved by Accelerated Proximal Alternating Linearized Minimization*
0.997456	*Exactly Robust Kernel Principal Component Analysis*
0.997211	*Graph-Laplacian PCA: Closed-Form Solution and Robustness*
0.997092	*Exact Low-Rank Matrix Completion from Sparsely Corrupted Entries Via Adaptive Outlier Pursuit*
0.997052	*Isometric Embedding and Continuum ISOMAP*
0.996688	*Matrix completion by deep matrix factorization*

Table 4: Recommendation result for Prof. Andre Milzarek's paper: *A Semismooth Newton Stochastic Proximal Point Algorithm with Variance Reduction* by GCN using transductive method

Scores	Title
0.999697	*Error Bound and Convergence Analysis of Matrix Splitting Algorithms for the Affine Variational Inequality Problem*
0.999641	*On the convergence of the coordinate descent method for convex differentiable minimization*
0.999583	*A Proximal Alternating Direction Method of Multiplier for Linearly Constrained Nonconvex Minimization*
0.999417	*On the linear convergence of descent methods for convex essentially smooth minimization*
0.998926	*Quasi-maximum-likelihood multiuser detection using semi-definite relaxation with application to synchronous CDMA*
0.998608	*Dynamic Spectrum Management: Complexity and Duality*
0.998319	*Approximation Bounds for Quadratic Optimization with Homogeneous Quadratic Constraints*
0.998145	*Error bounds and convergence analysis of feasible descent methods: a general approach*
0.994802	*On the linear convergence of the alternating direction method of multipliers*
0.998085	*Transmit beamforming for physical-layer multicasting*

Table 5: Recommendation result for Prof. Andre Milzarek's paper: *A Semismooth Newton Stochastic Proximal Point Algorithm with Variance Reduction* by GraphSAGE using inductive method

Scores	Title
0.996411	*Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems*
0.996287	*On the Convergence Rate of Dual Ascent Methods for Linearly Constrained Convex Minimization*
0.996171	*Convergence to good non-optimal critical points in the training of neural networks: Gradient descent optimization with one random initialization overcomes all bad non-global local minima with high probability*
0.995749	*On the Superlinear Convergence of Interior-Point Algorithms for a General Class of Problems*
0.995666	*On Iteration Complexity of a First-Order Primal-Dual Method for Nonlinear Convex Cone Programming*
0.995354	*On the convergence of the iteration sequence in primal-dual interior-point methods*
0.995296	*Error Bound and Convergence Analysis of Matrix Splitting Algorithms for the Affine Variational Inequality Problem*
0.995210	*Auxiliary Problem Principle of augmented Lagrangian with Varying Core Functions for Large-Scale Structured Convex Problems*
0.994802	*A proof of convergence for gradient descent in the training of artificial neural networks for constant target functions*
0.994586	*First-Order Primal-Dual Augmented Lagrangian Method for Nonlinear Cone Constrained Composite Convex Optimization*

Table 6: Recommendation result for Prof. Andre Milzarek's paper: *A Semismooth Newton Stochastic Proximal Point Algorithm with Variance Reduction* by GraphSAGE using transductive method

Scores	Title
0.995166	*Approximation Algorithms for Quadratic Programming*
0.994754	*Error Bounds for Quadratic Systems*
0.994748	*A Proximal Alternating Direction Method of Multiplier for Linearly Constrained Nonconvex Minimization*
0.994640	*A Unified Convergence Analysis of Block Successive Minimization Methods for Nonsmooth Optimization*
0.994478	*Comparison of two binaural beamforming approaches for hearing aids*
0.994225	*On the Convergence Rate of Dual Ascent Methods for Linearly Constrained Convex Minimization*
0.994027	*Level-set Subdifferential Error Bounds and Linear Convergence of Variable Bregman Proximal Gradient Method*
0.993912	*A block coordinate descent method of multipliers: Convergence analysis and applications*
0.993116	*Error bounds and convergence analysis of feasible descent methods: a general approach*
0.992559	*A Robust Gradient Tracking Method for Distributed Optimization over Directed Networks*