Notes of IE4483

MoeKid101

1 Association Analysis

Problem Background: $I = \{i_1 \sim i_d\}$ be the set of items in a dataset $T = \{t_1 \sim t_N\}$ where $t_i \subseteq I$. Want to discover correlations between items.

Def Itemset: any subset $S \subseteq I$.

Def k-Itemset: an itemset contatining k items.

Def Contain: a transaction t_j contains an itemset X if $X \subseteq t_j$.

Def Support Count: the support count $\sigma(X)$ of itemset X is the number of transactions containing X.

Def Association Rule: an implication $X \to Y$ where $X, Y \subseteq I$ and $X \cap Y = \emptyset$.

Def Support: $\text{Supp}(X \to Y) = \mathbf{Pr}[X \cup Y] = \frac{\sigma(X \cup Y)}{|T|}$.

Def Confidence: $\text{Conf}(X \to Y) = \mathbf{Pr}[Y|X] = \frac{\sigma(X \cup Y)}{\sigma(X)}$.

Def Maximal Frequent Itemset: V is maximal frequent itemset if $\sigma(V) \geq m_s$ and all children of V is infrequent.

Def Closed Frequent Itemset: V is a closed frequent itemset if $\sigma(V) \geq m_s$ and $\sigma(V) \neq \sigma(X)$ for any X be the children of V.

Problem Definition: (*Association Rule Mining*) given I and T, find all the rules with $\text{Supp} \geq m_s$ and $\text{Conf} \geq m_c$. Brute-force is too costly, so we deal with two constraints separately.

(1) Generate frequent itemsets (satisfying $\text{Supp}(\cdot) \geq m_s$).

(2) Rule generationg (extract $\text{Conf}(\cdot) \geq m_c$ from frequent itemsets).

Problem Solution **(to generating frequent itemsets)**:

(1) Construct a level graph where the k-th level contain C_d^k nodes representing the set of k-itemsets. u_k (belonging to level k) is connected to v_{k+1} iff. $u_k \subset v_{k+1}$.

(2) Start from level 0 and always derive level $k+1$ from level k, thus obtaining all frequent itemsets.

Lemma : A set V is frequent only if all ascendants are frequent.

Thm : The set of frequent itemsets equals the set of maximal frequent itemsets and their ascendants.

Corollary : The frequent itemsets is recorded by maximal frequent itemsets, and the support information is stored only for closed frequent itemsets.

Problem Solution **(to generating association rules)**:

Lemma : If rule $\text{Conf}(X \to Y - X) \leq m_c$, then $\forall X' \subset X$, $\text{Conf}(X' \to Y - X') \leq m_c$.

Problem lies in that association is not enough, because a strong rule $X \to Y$ may have X and Y negatively-correlated. We need correlation using interestingness measures like $\text{Lift}(X \to Y) = \frac{\mathbf{Pr}[Y|X]}{\mathbf{Pr}[Y]}$.

2 Decision Tree

First consider classification problem $y = f(\boldsymbol{x})$. Two steps: training (to construct a model based on training set) and classification (to predict class labels of test set).

Def Data Sample: A data sample $\boldsymbol{x}$ is an n-dimensional attribute vector $\boldsymbol{x} = [x_1, x_2, ..., x_n]^\top$.

Categories of samples:

(1) Nominal attributes: attributes like gender or color, no mathemtaical operations.

(2) Ordinal attributes: also categories but with predefined order, *e.g.* education levels.

(3) Interval attributes: Defined order and meaningful intervals between values, *e.g.* dates and temperature.

(4) Ratio attributes: Defined order, meaningful intervals and a meaningful "zero", *e.g.* height, weight. With zero predefined, ratios between samples are meaningful.

Def Decision Tree: Each node is a test, each branch representing outcome of test and each leaf node holds a class label.

Selection Measures:

(1) Information Gain: (used by ID3)

Def Entropy: For random variable $X \sim p(x)$, $H(X) = \mathbf{E}_p[-\log p(x)] = -\int_{\mathbb{R}} p(x) \log p(x) \mathrm{d}x$. (equivalently denoted by $\text{Info}(p)$)

Def Conditional Entropy: For random variable X and Y, $H(Y|X) = -\int_{\mathbb{R}^2} p(x,y) \log p(y|x) \mathrm{d}y\mathrm{d}x$. For classification of dataset D into subsets according to attribute A, $\text{Info}_A(D) = \sum_j p(A = a_j)\text{Info}(D_j)$.

Def Information Gain: $\text{Gain}(A) = \text{Info}(D) - \text{Info}_A(D)$ or $\text{Gain}(X) = H(Y) - H(Y|X)$.

(2) Gain Ratio: (used by C4.5) Introduced because information gain favor attributes with many classes.

Def Split Information: Measures how much information does the split contain, $\text{SplitInfo}_A(D) = -\sum_j p(A = a_j) \log p(A = a_j)$ or $H(X)$.

Def Gain Ratio: $\text{GainRatio}(X) = \frac{\text{Gain}(X)}{H(X)}$ or $\text{GainRatio}(A) = \frac{\text{Gain}(A)}{\text{SplitInfo}_A(D)}$.

(3) Gini Index: (used in CART) $\text{Gini}(D) = 1 - \sum_i p_i^2$. Splitting through attribute A leads to $\text{Gini}_A(D) = \sum_j \frac{|D_j|}{|D|}\text{Gini}(D_j)$.

Using this we have $\Delta\text{Gini}(A) = \text{Gini}(D) - \text{Gini}_A(D)$.

Tree Pruning: either before (early stopping) or after building the tree.

Advantages: Computationally efficient, easy to interpret, robust to noise, avoid overfitting, don't require priors of distribution.

Evaluation Metrics: Sensitivity ($\frac{\text{TP}}{\text{TP+FN}}$), specificity ($\frac{\text{TN}}{\text{TN+FP}}$), precision ($\frac{\text{TP}}{\text{TP+FP}}$), recall ($\frac{\text{TP}}{\text{TP+FN}}$), F_β score ($\frac{(1+\beta^2)\times\text{Prec}\times\text{Reca}}{\beta^2\times\text{Prec}+\text{Reca}}$), TPR ($\frac{\text{TP}}{\text{TP+FN}}$), FPR ($\frac{\text{FP}}{\text{TN+FP}}$), ROC curve (FPR-x, TPR-y), AUC.

Four aspects of evaluation: ***speed***, ***robustness*** (noise), ***scalability*** (construct classifier efficiently given large dataset) and ***interpretability***.

Evaluate classifier through: ***handout***, ***k-fold cross validation***, ***leave-one-out***.

3 k Nearest Neighbours

Similarity measure:

(1) minkowski distance $d(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{d} |x_i - y_i|^r\right)^{1/r}$ satisfying positivity, symmetricity and triangle inequality.

(2) cosine similarity $\cos(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x}^\top \boldsymbol{y}}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|}$ (doesn't satisfy the three properties).

4 Support Vector Machine

Def Geometric Margin: $\gamma = \min_{(\boldsymbol{x},y)\in\mathcal{D}} \frac{y}{\|\boldsymbol{w}\|}\left(\boldsymbol{w}^\top \boldsymbol{x} + b\right)$.

Optimization problem written as:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\left(\sum_{i=1}^{n} \xi_i\right)^k$$
$$\text{s.t. } \forall i, y^{(i)}\left(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b\right) \geq 1 - \xi_i \text{ and } \boldsymbol{\xi} \geq \mathbf{0}$$

Multi-class classifier: ***one-against-rest*** (use the result from the one with highest confidence), ***one-against-one*** (voting).

5 Symbolic AI

Defined by **problem states**, **state space** and **inference rules**. Neural networks are not symbolic, but sub-symbolic instead. A problem is described by a *state space graph*, which can be solved through searching.

Knowledge representation: an example is predicate calculus, a formal language for AI, describing problem states and inference rules.

6 Search

Def Parent and Child: Directed G, if $(n_j, n_k) \in E$, then n_j is parent of n_k; n_k is the child of n_j.

Def Siblings: Directed G, if $(n_j, n_k) \in E$ and $(n_j, n_i) \in E$, then n_i and n_k are siblings.

Def Rooted Graph: Directed G with a unique node n_s from which all paths originate.

Def Leaf: n_s in directed G with no children.

Def Ancestor / Descendant: On a path $(n_0, n_1, ..., n_l)$ in a rooted graph, n_i is the ancestor of $n_{i+1}, ..., n_l$ and the descendant of $n_0, ..., n_{i-1}$.

Def Incident: In $G = (V, E)$, the edges with an end node v are incident on v.

Def Parallel Arcs: If multiple $e \in E$ are incident on the same two nodes u, v then these es are parallel arcs.

Def Tree: A graph with a unique path between every pair of nodes.

Def Strategy:: Picking the order of node expansion. Evaluated by **completeness** (DFS isn't complete as it may stuck in infinite-depth dead end), **time complexity**, **space complexity** (measured in b as maximum branching factor, d as depth of least-cost solution, m as maximum depth of state space) and **optimality**.

Uninformed: only information in problem definition.

Informed: problem-specific knowledge to guide search.

6.1 Uninformed Search

State Space Approach: Nodes → states. Arcs → steps. There exists *goal conditions* as solutions. The problem is finding a path from start state to goal.

Def State Space:: four-tuple (N, A, S, GD) where N is the set of nodes, A is the set of arcs, $S \subset N$ is the start states, $GD \subset N$ is the goal states.

Data-driven (forward, fact→fact→goal) or *goal-driven* (backward, goal→subgoal→fact) or hybrid of these two.

Breath-First Search: Maintain two lists *open* (examined but children unexamined) and *closed* (all children examined). Then *open* is a queue (FIFO).

Depth-First Search: Two implementations. **(1)** *open* (stack, LIFO) and *closed*. Always pop the first element and add its children in the front of *open*. **(2)** Record path from root to current node (also stack) and whether each node has been visited.

Backtracking (with DFS implementation **(1)**): *open* is **New State List**. *closed* is **Dead Ends** and **State List**.

Iterative DFS: For increasing i do DFS with depth bound i. Reduce space complexity at the cost of higher time complexity.

Comparison between BFS and DFS:

(1) BFS is sure to find shortest path while DFS doesn't.

(2) BFS never explores blind alley.

(3) DFS requires less memory. Suppose the current depth is

n, B is branching factor, then DFS requires $O(Bn)$ while BFS requires $O(B^n)$.

6.2 Heuristic Search

Heuristic function $h : V \to \mathbb{R}$ is the estimated cost from v to goal state. **Admissible** $h(n) \leq$ real cost. Can have $g : V \to \mathbb{R}$ as the cost from initial state to v. Use $f(v) = g(v) + h(v)$.
Fallible because of limited information. Might not find solution and this limitation can't be eliminated by better heuristics or better searching algorithms.
Hill Climbing: Select the best child but not retain any information. Halted after a state better than its children is reached.

Might have infinite paths, stuck at local maxima.

Best-First Search: maintain *open* and *closed* where *open* is a priority queue. Always remove the node minimiznig $h(n)$ and add its children. ($g(v) = 0$)
A*: Requires an admissible heuristic and $g(v)$ is the accumulated cost from initial state to v. With the admissible property, A* is complete and optimal. Space complexity is high to keep all nodes in memory.
Thm : If $\forall v$, admissible $h_2(v) \geq h_1(v)$, then h_2 dominates h_1 and is better for search.

7 Bayes

Hypothesis h and training data D with prbabilities $P(h)$ and $P(D)$ respectively. We are interested in $P(h|D)$.
Thm Bayes Theorem: $P(h|D) \propto P(D|h)P(h)$ where $P(h)$ is the prior before observing D, $P(D|h)$ is the likelihood of observing D given h.
Def Maximum A Posteriori: $h_{\text{MAP}} = \arg\max P(D|h)P(h)$. With constant $P(h)$ for different h, MAP equivalent to MLE.
Naive Bayes Assumption Features are conditionally independent. $P(x_1, ..., x_d|h) = \prod_{i=1}^{d} P(x_i|h)$.

8 Clustering

Def Cluster: A set of data similar to each other. Clustering is make data in same group more similar to each other than to those in other groups.

K-Means: Minimize $\sum_{j=1}^{k} \sum_{i \in C_j} d\left(\boldsymbol{x}^{(i)}, \boldsymbol{\mu}_j\right)$ (distance metric may differ as long as non-negativity, symmetricity and triangle inequality holds).
(1) Pick k samples as initial centers $\boldsymbol{\mu}_k$.
(2) Assign $\boldsymbol{x}^{(i)}$ to its closest cluster center.
(3) Update $\boldsymbol{\mu}_j = \frac{1}{|C_j|} \sum_{i \in C_j} \boldsymbol{x}^{(i)}$. Return to **(2)**.

Effective and easy to implement. But k is manually determined and may stuck at poor local minimum (prone to initialization).

Hierarchical Agglomerative Algorithm (HAC): Always merge two clusters with minimum distance (minimum, maximum, average or center).

Deterministic, suits to any post-determined k. But more memory- and computationally-costly.

9 Principal Component Analysis

Applied to remove feature redundancy, model complexity and remove noise.
Defined by minimizing MSE of projection or maximizing the variance of projected data. $\max_{\|\boldsymbol{v}\|=1} \boldsymbol{v}^\top \boldsymbol{X}^\top \boldsymbol{X} \boldsymbol{v}$.