Data Mining (CS 145) Midterm Cheat Sheet by Patricia Xiao

Models

Model	Data Type	Task Type
Linear Regression	Vector	Prediction
Logistic Regression	Vector	Classification
Decision Tree	Vector	Classification
SVM	Vector	Classification
NN	Vector	Classification
KNN	Vector	Classification
K-means	Vector	Clustering
hierarchical clustering	Vector	Clustering
DBSCAN	Vector	Clustering
Mixture Models	Vector	Clustering

Basic Concepts

- **Dispersion:** Quartiles& Inter Range (Q_1 25%, Q_3 75%, IQR $= Q_3 - Q_1$, Outlier 1.5 IQR away $Q_{1/3}$), 5 n Summary: min, Q_1, median, Q_3, max
- Bias: $E(\hat{f}(x)) - f(x)$, Variance: $Var(\hat{f}(x)) = E[(\hat{f}(x) - E(\hat{f}(x)))^2]$, $E[(\hat{f}(x) - f(x) - \epsilon] = bias^2 + variance + noise$; $E(\epsilon) = 0$, $Var(\epsilon) = \sigma^2$; bias→underfit; variance→overfit
- Model Evaluation and Selection: K-way cross validation, AIC ($2k - 2\ln(\hat{L})$) & BIC ($k\ln(n) - 2\ln(\hat{L})$) (k params, n objs), Stepwise feature selection (forward: add, backward: from full model)
- Generalized Linear Model (GLM): exponential family, $p(y;\eta) = b(y)exp(\eta^T T(y) - a(\eta))$; **linear decision boundary**
- **Bagging:** Bootstrap Aggregating (multi-datasets → multiple classifiers → combine classifiers)
- **Kernel:** $K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$
- Chain rule: $\partial J/\partial x = (\partial J/\partial y)(\partial y/\partial x)$
- Minkowski distance (l_h): $d(x,y) = \sqrt[h]{\sum_i^d (x_i - y_i)^h}$, l_1 Manhattan, l_2 Euclidean, l_∞ supremum; triangle inequality applies ($d(i,j) \leq d(i,k) + d(k,j)$).
- **Confusion Matrix:** True / False for correctness, Positive / Negative for result
- Multi-class classification: All-vs-all (AVA) is better than One-vs-All (OVA)

Formula

1. $\sigma^2 = E[(X - E(X))^2] = E(X^2) - E^2(X)$
2. $\|\alpha\|^2 = \alpha^T \alpha$ where α is a vector.
3. $(AB)^T = B^T A^T$, $(AB)^{-1} = B^{-1}A^{-1}$
4. $\frac{\partial(Ax)}{\partial x} = A$, $\frac{\partial(AX)}{\partial X} = A^T$
5. $\frac{\partial(x^T A x)}{\partial x} = x^T(A + A^T)$, $\frac{\partial(X^T A^T A X)}{\partial X} = 2A^T A X$
6. $X \sim N(\mu, \sigma^2) \Rightarrow f(X = x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
7. $\sigma'(x) = \sigma(x)(1 - \sigma(x))$
8. $\log(a^b) = b\log a$, $\log(ab) = \log(a) + \log(b)$
9. Classifiers $f_i(x)$: $var(\frac{\sum_i f_i(x)}{t}) = var(f_i(x))/t$
10. $a \cdot b = \sum a_i b_i = \|a\|\|b\|\cos(a,b)$
11. normal $\mathbf{n}$, any vector in plane, $\mathbf{x}$, $\mathbf{n} \cdot \mathbf{x} = 0$
12. covariance: $\sigma(X_1, X_2) = E(X_1 - \mu_1)(X_2 - \mu_2)$

Tools

1. **mean - mode = 3 × (mean - median)** mode(peak)∼median∼mean(∼ tail)
2. Z-score (normalization): $Z = \frac{x-\mu}{\delta}$ (robust: mean absolute deviation, $z_{jf} = \frac{x_{if} - mean_f}{sum_f}$) (nominal: dummy variable(s), ordinal: $(r-1)/(M-1)$) where r, M start from 1.
3. **Logistic / Sigmoid** Function: $\sigma(x) = \frac{1}{1+e^{-x}}$
4. Entropy: $H(Y) = -\sum_{i=1}^m p_i \log(p_i)$; Conditional Entropy: $H(Y|X) = \sum_x p(x) H(Y|X = x)$
5. Cross Entropy Loss: $H(q,p) = -\sum_k q_k \log(p_k)$
6. Lagrange multiplier α is used to solve Quadratic Programming (e.g. SVM)
7. Soft margin (allow moving at a cost): minimizing $\Phi(w) = 1/2 w^T w \Rightarrow \Phi(w) = 1/2 w^T w + C\sum \zeta_i$, limitation $y(w^T x_i + b) \geq 1 \Rightarrow y(w^T x_i + b) \geq 1 - \zeta_i$ ($\zeta_i \geq 0$); doesn't affect the solution of SVM.
8. ROC (Receiver Operating Characteristics): TP rate (y-axis) - FP rate (x-axis), score = area below curve
9. **Dendrogram**: the hierarchical, cut to clusters.

Linear Regression

$y = x^T\beta$ where bias term $x_{i0} = 1$, x: $(n \times (p+1))$ matrix, y: $(n \times 1)$ vector, β: $((p+1) \times 1)$ vector. **Continuous** $y = x\beta^T$. (**OLS**, Ordinary Least Square) $J(\beta) = \frac{1}{2n}(X\beta - y)^T(X\beta - y) = \frac{1}{2n}(\beta^T X^T X\beta - y^T X\beta - \beta^T X^T y + y^T y)$.
Closed form solution: $\frac{\partial J}{\partial \beta} = 0$, $\hat{\beta} = (X^T X)^{-1} X^T y$
Gradient Descent: $\beta^{(t+1)} := \beta^{(t)} - \eta\Delta$
Batch GD: (converge) $\Delta = \frac{\partial J}{\partial \beta} = \sum_i x_i(x_i^T \beta - y_i)/n$
Stochastic GD: (n times) $\Delta = -(y_i - x_i^T \beta^{(t)}) x_i$
LR with Probabilistic Interpretation: (using **MLE**, *Maximum Livelihood Estimation*) $L(\beta) = \prod_i p(y_i|x_i, \beta) = \prod_i p(N(x_i^T\beta, \sigma^2)) = \prod_i \frac{1}{\sqrt{2\pi\sigma^2}} exp\{-\frac{(t_i - x_i^T\beta)^2}{2\sigma^2}\}$
Invertible $X^T X$: add $\lambda\sum_{j=1}^p \beta_j^2$ to $\sum_i (y_i - x_i^T\beta)^2$ (**Ridge Regression**, or linear regression with l_2 norm)
Non-linear Correlation: create new terms e.g. x^2

Logistic Regression

Generalized **linear** model (GLM).
$P(Y = 1|X, \beta) = \sigma(X^T\beta) = \frac{e^{X^T\beta}}{1+e^{X^T\beta}}$
$P(Y = 0|X, \beta) = 1 - \sigma(X^T\beta) = \frac{1}{1+e^{X^T\beta}}$
$Y|X, \beta \sim Bernoulli(\sigma(X^T\beta))$
MLE: $L = \prod_i p_i^{y_i}(1 - p_i)^{1-y_i}$, p_i is $P(Y = 1|X, \beta)$
Eq to max log likelihood $L = \sum_i (y_i x_i \beta - \log(1 + e^{x_i^T\beta}))$
Gradient ascent $\beta^{new} = \beta^{old} + \eta\frac{\partial L(\beta)}{\partial \beta}$
Newton-Raphson update $\beta^{new} = \beta^{old} - (\frac{\partial^2 L(\beta)}{\partial \beta})^{-1} \frac{\partial L(\beta)}{\partial\beta\partial\beta^T}$
Cross Entropy Loss (p for prediction, q for ground truth, $(q_0, q_1)|_{y=0} = (1,0)$, $(q_0, q_1)|_{y=1} = (0,1)$, $(p_0, p_1) = (P(Y = 0), P(Y = 1))$: $H(p,q) = -yx^T\beta + log(1 + e^{x^T\beta})$)

EM Algorithm

A framework to approach maximum likelihood.
$p(x_i, z_i = C_j) = w_j f_j(x_i)$, $p(x_i) = \sum_j w_j f_j(x_i)$
$p(D) = \prod_i p(x_i) = \prod_i \sum_j w_j f_j(x_i)$
$\log(p(D)) = \sum_i \log(\sum_j w_j f_j(x_i))$
E(expectation)-step assigns objects to clusters.

$$w_{ij}^{t+1} = p(z_i = j|\theta_j^t, x_i) \propto p(x_i|z_i = j, \theta_j^t) p(z_i = j) = f_j(x_i) w_j$$

M(maximization)-step finds the new clustering w.r.t. conditional distribution $p(z_i = j|\theta_j^t, x_i)$.

$$\theta^{t+1} = \underset{\theta}{\text{argmax}} \sum_i \sum_j w_{ij}^{t+1} \log L(x_i, z_i = j|\theta)$$

Decision Tree

m for $|y|$ in D, v for $|A|$
Expected Information needed to classify a tuple in D: $Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$
Info after split A: $Info_A(D) = \sum_{j=1}^{v} \frac{D_j}{D} \times Info(D_j)$
Info Gain (ID3): $Gain(A) = Info(D) - Info_A(D)$
Info gain biases towards multivalued attributes.
$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{D_j}{D} \times \log_2(\frac{D_j}{D})$
GR (C4.5): $GainRatio(A) = Gain(A)/SplitInfo(A)$
GR biases towards unbalanced splits.
$Gini(D) = 1 - \sum_{j=1}^{m} p_j^2$ for impurity
$Gini_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} Gini(D_j)$
Gini (CART): $\Delta Gini(A) = Gini(D) - Gini_A(D)$
Gini index also biases towards multivalued attributes.
STOP: same class; last attr; no sample (maj. vot.)
Avoid Over Fitting: Pre/Post-pruning, random forest
Classification → Prediction: Maj. Vote → e.g. Avg for leaf node.
turn to regression tree, $Var(D_j) = \sum_{y \in D_j} (y - \overline{y})^2/|D_j|$, look for the lowest **weighted average variance** $Var_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Var(D_j)$
A different view: leaf = box in the plane
Random forest is a set of trees, ensemble, bagging, good at **classification**, handles large & missing data, not good at **predictions**, lack interpretation.

SVM

$y = sign(\mathbf{W} \cdot \mathbf{X} + b)$, separating hyperplane $y = 0$
SVM searches for **M**aximum **M**arginal **H**yperplane
To Maximize Margin $\rho = \frac{2}{\|\mathbf{w}\|}$, w. Lagrange multiplier α, $L(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i (y_i (w^T x_i + b) - 1)$.
$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{N} \alpha_i y_i x_i = 0$, $\frac{\partial L}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i = 0$
Solution: $w = \sum \alpha_i y_i x_i$, $b = y_k - w^T x_k$
$f(x) = w^T x + b = \sum \alpha_i y_i x_i^T x + b$ default threshold 0
Linear v.s. Non-linear SVM: **Kernel**
Non-linear Decision Boundary: $f(x) = w^T \Phi(x) + b = \sum \alpha_i y_i K(x_i, x) + b$
Scalability: CF-Tree, Hierarchical Micro-cluster, selective declustering (decluster the clusters who could be support cluster; support cluster: centroid on support vector)

Perceptron (Single Unit)

$$x_i \xrightarrow{w_i} \sum(+b) \xrightarrow{f} o$$

Input vector x, Weight vector w, Bias b, weighted sum, going through activation function f, reach output o.

Backpropagation (BP)

Stochastic GD + Chain Rule
Special case: Sigmoid + Square loss, 2 layers
Assume: i, j, k are input, hidden, output layers' denotion, and O for output, T for true value.
$Err_k = O_k(1 - O_k)(T_k - O_k)$, $Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$, $w_{ij} = w_{ij} + \eta Err_j O_i$ and $w_{jk} = w_{jk} + \eta Err_k O_j$, $\theta_j = \theta_j + \eta Err_j$ and $\theta_k = \theta_k + \eta Err_k$.
$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial O_k} \frac{\partial O_k}{\partial O_j} \frac{\partial O_j}{\partial w_{ij}} = -\sum_k [(T_k - O_k)][O_k(1 - O_k) w_{jk}][O_j(1 - O_j) O_i]$

Neural Network (NN)

$n_{layers} = n_{hidden} + n_{output}(1)$
Feed-forward, Non-linear regression, capable of any continuous function.
Backpropagation is used for learning.

k - Nearest Neighbors (kNN)

Lazy learning (instead of eager), instance-based
Consider k nearest neighbors; maj. voting or average. (Could be distance-weighted.)
Curse of dimensionality: influence of noise
Get rid of irrelevant features; select proper k.
Proximity refers to similarity or **dissimilarity**. Always applies to binary values. If nominal, could do simple matching, or use a series of binary to represent a non-binary; ordinal: rank, normalize $z_{if} = \frac{r_{if} - 1}{M_f - 1}$.
Proximity could be measured by $\frac{|(0,1)| + |(1,0)|}{all}$ for symmetric variables, $\frac{|(0,1)| + |(1,0)|}{all - |(0,0)|}$ or Jaccard coefficient (similarity) $\frac{|(1,1)|}{all - |(0,0)|}$ for asymmetric.
Mixed type attributes: weighted combine.
Another method: cosine similarity $cos(d_1, d_2)$

Evaluation: Classification

Holdout method; Cross-validation (k-fold) LOO.
Confusion Matrix: True / False Positive / Negative
Accuracy = (TP + TN) / All
Error Rate = (FP + FN) / All
Sensitivity = TP / P (P = TP + FN)
Specificity = TN / N (N = FP + TN)
Precision = TP / P' (P' = TP + FP)
Recall = TP / P = Sensitivity
F_1 / F-score = $\frac{2 \times Precision \times Recall}{Precision + Recall}$
$F_\beta = \frac{(1+\beta^2) \times Precision \times Recall}{\beta^2 \times Precision + Recall}$ (R: P = $\beta : 1$)
ROC curve: TP rate (y) - FP rate (x). (area under)
TPR = TP / P, FPR = FP / N

Clustering

K-means: $J = \sum_{j=1}^{k} \sum_i w_{ij} \|x_i - c_j\|^2$
Assign $w_{ij} = 1$ to each x_i closest c_j; assign the center to be new centroid; stop when no change. O(tkn). For continuous, convex-shaped data, sensitive to noise.
K-modes: *mean* → *mode*, for categorical data
K-medoids: representative objects, e.g. PAM (s)
Hierarchical: bottom-up **Agglomerative Nesting (AGNES)** merges two closest clusters until end up in 1; top-down **DIANA (Divisive Analysis)**. $O(n^2)$.
Cluster Distance: Single link for min element-wise dist; Complete link for max; average for avg element pairs dist; centroid, medoid (center obj).
DBSCAN: Set Eps ϵ and $MinPts$. Neighborhood defined as $N_\epsilon(q) : \{p \in D | dist(p, q) \le \epsilon\}$. Core point $|N_\epsilon(q)| \ge MinPts$. p is **directly density-reachable** from q if q is core point and $p \in N_\epsilon(q)$; **density-reachable** if $q \to p_2 \to \cdots \to p$; **density-connected** if $o \to \cdots \to p \bigwedge o \to \cdots \to q$. Cluster: max set density-connected points. Individual points are **noise**. DFS $O(n \log n)$ w. spacial index, else $O(n^2)$.
Mixture Model: soft clustering ($w_{ij} \in [0, 1]$ rather than $w_{ij} \in \{0, 1\}$), joint prob of object i and cluster C_j: $p(x_i, z_i = C_j) = w_j f_j(x_i)$, using EM algorithm.
Gaussian Mixture Model (GMM): ⊃ **k-means**
Generative model, for each object, pick cluster Z, from $X|Z \sim N(\mu_Z, \sigma_Z^2)$ sample value; Overall likelihood function $L(D|\theta) = \prod_i \sum_j w_j p(x_i | \mu_j, \sigma_j^2)$;
E $w_{ij}^{t+1} = (w_j^t p(x_i | \mu_j^t, (\sigma_j^2)^t)) / (\sum_k w_k^t p(x_i | \mu_k^t, (\sigma_k^2)^t))$,
M $\mu_j^{t+1} = (\sum_i w_{ij}^{t+1} x_i) / (\sum_i w_{ij}^{t+1})$, $(\sigma_j^2)^{t+1} = (\sum_i w_{ij}^{t+1} (x_i - \mu_j^{t+1})^2 / (\sum_i w_{ij}^{t+1})$, $w_j^{t+1} = \sum_i w_{ij}^{t+1} / n$ (in 1-d case)
Why EM works? E-Step find **tight** lower bound L of ℓ at θ_{old}, M-Step find θ_{new} to maximize the lower bound. $\ell(\theta_{new}) \ge L(\theta_{new}) \ge L(\theta_{old}) = \ell(\theta_{old})$

Evaluation: Clustering

extrinsic (supervised) vs. intrinsic (unsupervised)
$purity(C, \Omega) = \frac{1}{N} \sum_K \max_j |c_k \cap \omega_j|$ (C out, Ω truth)
Normalized Mutual Information:
$NMI(C, \Omega) = \frac{I(C, \Omega)}{\sqrt{H(C) H(\Omega)}}$
$I(C, \Omega) = \sum_k \sum_j P(c_k \cap \omega_j) \log \frac{P(c_k \cap \omega_j)}{P(c_k) P(\omega_j)} = \sum_k \sum_j \frac{|c_k \cap \omega_j|}{N} \log \frac{N |c_k \cap \omega_j|}{|c_k||\omega_j|}$
$H(\Omega) = -\sum_j P(\omega_j) \log P(\omega_j) = -\sum_j \frac{|\omega_j|}{N} \log \frac{|\omega_j|}{N}$
Precision and Recall: same / different class / cluster
Select k: plot square loss - k, larger k smaller cost, find **knee** points; BIC penaltize; Cross validation

Midterm Reviews

- $\sigma = \sqrt{\frac{\sum(x_i - \overline{x})^2}{n}}$ where $|x| = n$
- parallel line $d = \frac{|c_2| - c_1|}{\sqrt{a^2+b^2}}$ where $ax + by + c = 0$
- $\ln(e^x) = x$, $e = 2.718281828459 \cdots \approx 2.7183$
- Likelihood is product of density (/ probability), log likelihood $logL(\theta) = \ell(\theta) = \sum_{allx} \log(P(X = x))$, find the max means $\ell'(\theta) = 0$
- $A \times B = C$ then $c_{ij} = a_{i*} \cdot b_{*j}$
- Newton-Raphson update converges fast
- Lasso: l_1 norm's another name
- For binary class, the entropy $H = \sum p \log(p)$, $H(p) = p\log(p) + (1-p)\log(1-p)$, $H(1-p) = H(p)$, H is the maximum when $p = 0.5$. If $\log_2$ is chosen, $H(0.5) = 1$
- Supervised clustering pairs up the data points C_n^2, same-same = TP, diff-diff = TN. ($n2$) written in vertex means C_n^2.
- NN is sometimes written in the form of number on edge and number on node, then the number on edge means weight, the number on node means bias. Don't forget bias. In a way bias could be regarded as a threshold. Input layer $O_i = x_i$
- p features (p-d input nodes), $3 + 4$ hidden nodes, 2 hidden layers, k output nodes, then $3p + 3 * 4 + 4k$ weights are needed, $3 + 4 + k$ biases are needed.
- In KNN, larger K causes under-fitting and smaller K causes over-fitting.
- Method 1 to calculate A^{-1}: do row-wise options to $[A|E]$, change it into $[E|A^{-1}]$
- Method 2 to calculate A^{-1}: $\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$
- Hessian matrix: second order partial deriv.
- $H(p,q) = -yx^T\beta + \log(1 + \exp(x^T\beta))$ ($y = 0/1$, LR)

After Midterm

Model	Data Type	Task Type
Apriori	Set	Frequent Pattern Mining
FP Growth	Set	Frequent Pattern Mining
GSP	Sequence	Frequent Pattern Mining
PrefixSpan	Sequence	Frequent Pattern Mining
DTW	Sequence	Similarity Search
Naive Bayes for text	Text	Classification
pLSA	Text	Clustering

Frequent Pattern Mining Basis

Each data point is also called "transaction".
pattern = itemsets + association rules
motivation: find inherent regularities on data
K-itemset: a set of K items
absolute support (support count): frequency / occur
relative support: probability / fraction
Frequent: if an itemset's support > threshold
rule $X \to Y$: $support = P(XY)$, $confidence = P(Y|X)$
closed patterns X: X is frequent, $\forall Y \supset X$, $support(Y) \neq support(X)$
maximum patterns X: X is frequent, $\forall Y \supset X$, $support(Y) < threshold$, Y is not frequent
Closed patterns is a **lossless compression** of frequency patterns. (reduce # pattern & rules!)
Scalable mining methods: Apriori, FP-Growth, Eclat
*ECLAT: Frequent Pattern Mining with Vertical Data

Apriori

A candidate generation-and-test approach
The **Apriori property** of frequent patterns: any nonempty subset of a frequent itemset is also frequent.
Apriori **pruning**: having infrequent subset = not frequent (not to be generated / tested)
Method work-flow: **initially**, scan DB once for frequent 1-itemset (L_1) (and have all items in every transaction **ordered** in decreasing frequency order); **recursively** generate k candidate itemsets (C_k) from k-1 frequent itemsets (L_{k-1}) (join + **prune**) (join = self-joining L_{k-1}, join l_1, l_2 only when $l_1[: k-2] == l_2[: k-2]$ and $l_1[k-1] < l_2[k-1]$); test the candidates against DB; **terminate** when no frequent or candidate set can be generated.
$TDB \to C_1 \to L_1 \to C_2 \to C_2(pruned) \to L_2 \ldots$, finally return all L_k (**k** DB scans) ($\text{argmax}_k |L_k| \approx 2$)

FP Growth

A frequent pattern-growth approach.
Apriori limit: BFS → multi-scan; C_k workload
Improving Apriori by reduce passes of TDB scans, shrink n_candidates, reduce workload of support counting of candidates thus facilitate it.
Partition: scan TDB twice. Partition, find local frequent (relative support), any frequent set in TDB must be frequent in at least one partition.
Hash-based technique: based on hash-map, focus on C_2 sets map to the same index. Count supports on a hash-tree. (TID - T - $\{C_2\}$)
Sampling: select a sample of TDB, do Apriori, back to TDB 1-scan verify boarders (abcd not ab) of closure frequent patterns, scan TDB again find missed.
FP-Tree: compressed from the DB. Scan DB once, sort frequent 1-itemset descending order to be $f-list$ and scan it again. Header Table (columns: Item, frequency, head-pointer) + prefix tree, with counts at nodes, all nodes linked with a chain in order from the head-pointer. Root node empty {}, other node are like $a:3$.
FP-Growth: DFS, avoid explicit candidate generation. Grow long patterns from short ones using local frequent items only. Recursively mine FP-Tree by *conditional pattern base* → $conditional\ FP-Tree$ **until the tree has a single path or empty**.
Projection (DB | *itemset*): all transactions having the *itemset*. If d is freq in $DB|abc$, $abcd$ freq.
Form p's **conditional pattern base**: accumulate all **transformed prefix paths** of the item p. e.g. If there's a path $\{\} - f : 4 - c : 3 - a : 3 - m : 3 - p : 2$, the cond. base of p contains: $fcam : 2$
For each pattern-base accumulate the count for each item and construct the conditional FP-tree.
The answer of frequent patterns add back the base. e.g. $\{\} - a : 3 - b : 2$ for DB—c, then the frequent patterns are ac, bc, abc.
Project on each one **except** the most frequent item, do recursively, each time finds frequent x in $DB|s$ to be added to s so that sx be new frequent pattern.
Single Prefix-path: a special case, solution is to divide and concatenate
Scaling: parallel / partition projection (of the frequent items) (parallel: all at once, partition: in frequency order)
Runtime grows **slowly** with the decreasing of threshold, **divide-and-conquer**, compressed DB with no candidate generation and test, scan entire DB once

Eclat

Mining by exploring vertical data format, similar with inverted index. Having a t-id list that stores the list of transaction ids where a itemset appears, t(A). t(X) = t(Y) means $P(XY)$ is high; t(X) $\subset$ t(Y) means $P(Y|X)$ is high. diffset is used to accelerate mining (keep track of differences of tids).

Association Rules

$confidence(A \Rightarrow B) = P(B|A) = \frac{P(AB)}{P(A)}$

rule is from a frequent pattern l and all its non-empty subsets.

$Lift(AB) = \frac{P(AB)}{P(A)P(B)}$ = 1 independent, > 1 positively correlated, < 1 negatively correlated

$\chi^2 = \sum \frac{(Observed-Expected)^2}{Expected}$ has a table to check $p-value = P(\chi^2 > *)$, if $p-value$ is small enough, it rejects the null hypothesis, so A and B are dependent

$all\_confidence = \min\{P(A|B), P(B|A)\}$

$max\_confidence = \max\{P(A|B), P(B|A)\}$

$Kulczynski = \frac{1}{2}(P(A|B) + P(B|A))$

Cosine: $\cos(A, B) = \sqrt{P(A|B) \times P(B|A)}$

Lift and χ^2 are affected by null-transaction, that is the "not A and not B"s.

Imbalance Ratio (IR): $IR(A, B) = \frac{|sup(A)-sup(B)|}{sup(A)+sup(B)-sup(AB)}$ where sup refers to supports.

GSP

element / event is a non-empty **unordered** set of items, sequence is an **ordered** list of events, **length is the number of instances of items included.**

Always written like $\langle a(bc)de(fgh)\rangle$.

A is B's **subsequence** means: any elements in A is a subset of a corresponding element in B, those elements in B are in same order they appear in A.

Start from the same L_1, the major difference is **join**. In this case, s_1 and s_2 can be joined only if s_1 with 1^{st} item dropped and s_2 has the last item dropped are the same. Joined together: $s_1[0], s_{mid}, s_2[-1]$. Note that all the items in any element are "**sorted**" by f-list.

SPADE

$DB : \{\langle SID, EID, Items\rangle\} \Rightarrow Item(SubSeq) : \{\langle SID, EID\rangle\}$, and then join by growing the subsequences one at a time by Apriori (joining two of those $\{\langle SID, EID\rangle\}$ tables for Items / Sub-Sequences, e.g. $a, b \Rightarrow ab, ba \Rightarrow aba, bab$).

Similar **limitations** with GSP: costly generation & multiple scans by BFS & long patterns

Prefix Span

_: **blank space** used when the last *item* from *prefix* is from the first *element* of *suffix*.

Prefix-based **projection** (α'): a projection of α w.r.t. prefix β is the maximum subsequence of α with prefix β. e.g. $\alpha = \langle a(abc)(ac)d(cf)\rangle$, $\beta = \langle ad\rangle$, then $\alpha' = \langle ad(cf)\rangle$

Start from L_1, project the database into $|L_1|$ projected database accordingly, mine each subset recursively via corresponding projected databases. (e.g. a-proj $\Rightarrow$ ab-proj)

Note that a and $\_a$ are **different** in counting frequencies. With suffix last element s_1, $\_a$ only when $\_a$ appears at the front of the suffix, or see (s_1a*).

No candidate needed, major cost is projection, projected DB keeps shrinking and could be improved by **pseudo-projection** (using pointers to point to the division point of the prefix and suffix to save time and space, work well unless DB is too big for main memory, disk-access is slow).

Dynamic Time Warping (DTW)

Time series $Y = \{Y_t : t \in T\}$, time-index T.

An observation of time series with length N could be represented as $Y = \{y_1, y_2, \ldots y_N\}$.

Euclidean distance: $d(C, Q) = (\sum |c_i - q_i|^p)^{\frac{1}{p}}$ (l_p)

L_p norm cannot deal with offset and scaling. (sol: normalization $c_i' = \frac{c_i - \mu(C)}{\sigma(C)}$)

Warp time axis? Even with different length.

$X = \{x_1, \ldots x_N\}$, $Y = \{y_1, \ldots y_M\}$, find alignment between s.t. overall cost is minimized. Local distance (cost) between x_n, y_m: $c(x_n, y_m)$. We could have an $N \times M$ matrix of costs between all pairs.

Our goal: find an (N, M)-warping path $p = (p_1, p_2, \ldots, p_L)$ with $p_l = (n_l, m_l)$, conditions: (1) boundary, $p_1 = (1, 1), p_L = (N, M)$; (2) monotonicity, n_l, m_l non-decreasing with l; (3) step size, 1, $p_{l+1} - p_l \in \{(0, 1), (1, 0), (1, 1)\}$

Solving by DP: $D(n, m) = \min\{D(n-1, m), D(n, m-1), D(n-1, m-1)\} + c(x_n, y_m)$, where $D(n, m)$ denotes the DTW distance between $X(1, \ldots n)$ and $Y(1, \ldots m)$. $D(N, M) = DTW(X, Y)$, $D(n, 1) = \sum_{k=1}^{n} c(x_k, y_1)$, $D(1, m) = \sum_{k=1}^{m} c(x_1, y_k)$.

$O(NM)$ time complexity.

Trace back to find p^* from D, given that $p(l) = (n, m)$: p_{l-1} is $(1, m-1)$ if $n = 1$, $(n-1, 1)$ if $m = 1$, and otherwise argmin$\{D(n-1, m-1), D(n-1, m), D(n, m-1)\}$

Naive Time and Frequency Domain

Sometimes series data need to be transformed into Fourier domain to evaluate. $X_f = \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} x_t \exp(-j2\pi ft/n)$, $f = 0, 1, \ldots, n$

Parseval's Theorem: $\sum_{t=0}^{n-1} |x_t|^2 = \sum_{f=0}^{n-1} |X_f|^2$, Euclidean dist in time / freq domains are the same. Keep only first few coefficients brings no false dismissals.

Prediction

j^{th} lag of Y_t: Y_{t-j}, first diff $\Delta Y_t = Y_t - Y_{t-j}$, j^{th} autocorrelation ρ_j: $corr(Y_t, Y_{t-j}) = \frac{cov(Y_t, Y_{t-j}}{\sqrt{var(Y_t)var(Y_{t-j})}}$, $cov(Y_t, Y_{t-j}) = \frac{1}{T-j-1} \sum_{t=j+1}^{T} (Y_t - \overline{Y}_{j+1,T})(Y_{t-j} - \overline{Y}_{1,T-j})$. AR(1) check $Y_t = \beta_0 + \beta_1 Y_{t-1} + u_t$, $\beta_1 = 0$ for useless.

Naive Bayes for Text

Bayes' Theorem: $P(h|X) = \frac{P(X|h)P(h)}{P(X)}$

X data samples (evidence), $h : class(X) = Y$, $P(X)$ **fixed**, $P(h) = \pi$ prior probability, $P(X|h)$ likelihood $\prod_n \beta_{y_n}^{x_n}$, $P(X) = \sum_h P(X|h)P(h)$, $P(h|X)$ **posterior** probability. Maximum a posteriori $h_{MAP} = \text{argmax}_h P(X|h)P(h)$. $y^* = \text{argmax}_y \prod_n \beta_{y_n}^{x_n} \times \pi_y = \text{argmax}_y \sum_n x_n \log \beta_{y_n} + \log \pi_y$

Optimization: n word, j class, D documents, d document, $\beta_{jn} = \frac{\text{count of word n in class j}}{\text{count of all words in class j}}$ (smoothing: $\frac{\cdots+1}{\cdots+N}$), $\pi_j = \frac{\text{number of d in class j}}{|D|}$

For test document t, $p(y = c|x_t) \propto p(y = c) \times \prod_n (\beta_{cn})_{tn}^{x}$ where x_{tn} is n's appearance in x_t.

A generative model (not discriminative - like log reg).

Generative model $P(XY)$, discriminative $P(Y|X)$

Multinoulli distribution is two options, multi-tryout ($z \sim multinoulli(\pi)$), while **Multinomial** means multi-class, one tryout $((x_d \sim multinoumial(\beta_d)))$.

pLSA

corpus: a collection of documents, word w, doc d, topic z, word count in doc $c(w, d)$, word distribution each topic $\beta_{zw} = p(w|z)$, topic (soft) distribution each document $\theta_{dz} = p(z|d)$.

$\max \log L = \sum_{dw} c(w, d) \log \sum_z \theta_{dz}\beta_{zw}$ $s.t.$ $\sum_z \theta_{dz} = 1, \sum_w \beta_{zw} = 1$ is optimized by EM until converge. Generally E: $p(z|w, d) \propto p(w|z, d)p(z|d) = \beta_{zw}\theta_{dz}$, M: $\beta_{zw} \propto \sum_d p(z|w, d)c(w, d)$, $\theta_{dz} \propto \sum_w p(z|w, d)c(w, d)$. e.g. E: $p(z|w, d) = \frac{\beta_{zw}\theta_{dz}}{\sum_{z'} \beta_{zw}\theta_{dz'}}$, M: $\beta_{zw} = \frac{\sum_d p(z|w,d)c(w,d)}{\sum_{w',d} p(z|w',d)c(w',d)}$, $\theta_{dz} = \frac{\sum_w p(z|w,d)c(w,d)}{N_d}$, where N_d is the count of words in the document.