# CSE4502/5717

# BIG DATA ANALYTICS

## EXAM III REVIEW

TOPICS:

* ASSOCIATION RULES MINING

* POLYNOMIAL ARITHMETIC

* MACHINE LEARNING:

LINEAR REGRESSION, MULTILEVEL PERCEPTRONS

NEURAL NETWORKS.

# ASSOCIATION RULES MINING:

**INPUT:** A DB of $n$ transactions

$I$ = A set of possible items

$|I| = d$.

A transaction $\subseteq I$.

**OUTPUT:** Rules of the kind $X \rightarrow Y$

where $X \neq \phi$, $Y \neq \phi$; $X \cap Y = \phi$

$X \subseteq I$, $Y \subseteq I$

A Subset of items is also known as an itemset.

A k-itemset is an itemset with k items.

Two Steps in Generating Rules:

1. IDENTIFY all the FREQ. itemsets.
2. IDENTIFY Rules from them.

$\sigma(X) \triangleq$ the # of transactions in which $X$ occurs, $X \subseteq I$.

An itemset is frequent if

$\sigma(X) \geq n \cdot$ minSupport, when

minSupport is a fraction supplied by the user.

We are interested in Rules for which the support is $\geq$ min Support & the Confidence is $\geq$ min Confidence.

Support for a Rule $X \to Y$ is

$$\frac{\sigma(X \cup Y)}{n}$$

Confidence for $X \to Y$ is $\frac{\sigma(X \cup Y)}{\sigma(X)}$.

# A SIMPLE ALGORITHM:

Assume that each transaction is an indicator array.

TO FIND FREQ. k itemsets:

1. Generate all possible k itemsets. There are $\binom{d}{k}$ of them.
2. For each k-itemset compute support

TOTAL TIME = $O\left(\binom{d}{k} kn\right)$.

# APRIORI PRINCIPLE:

* If X is FREQUENT, then every Subset of X is also FREQ.

* If X is not FREQUENT, then no Superset of X can be.

Let $F_k$ denote the set of all frequent $k$-itemsets.

APRIORI Alg

Generate $F_1$; $k=1$

Repeat

1) Generate Candidates $C_{k+1}$; $F_{k+1} = \phi$;

2) Compute the support for every $q \in C_{k+1}$;
   If $q$ is freq. add it to $F_{k+1}$;

3) $k = k+1$. If $F_k$ is empty, quit;

Forever

# Generation of $C_{k+1}$ from $F_k$

① $F_k \times F_1$

② $F_k \times F_k$

$$\boxed{\begin{matrix} a_1 & a_2 & \cdots & a_{k-1} \\ | & | & & | \\ b_1 & b_2 & \cdots & b_{k-1} \end{matrix}} \begin{matrix} a_k \in F_k \\ \\ b_k \in F_k \end{matrix}$$

& $a_i = b_i$ for $1 \le i \le (k-1)$, then

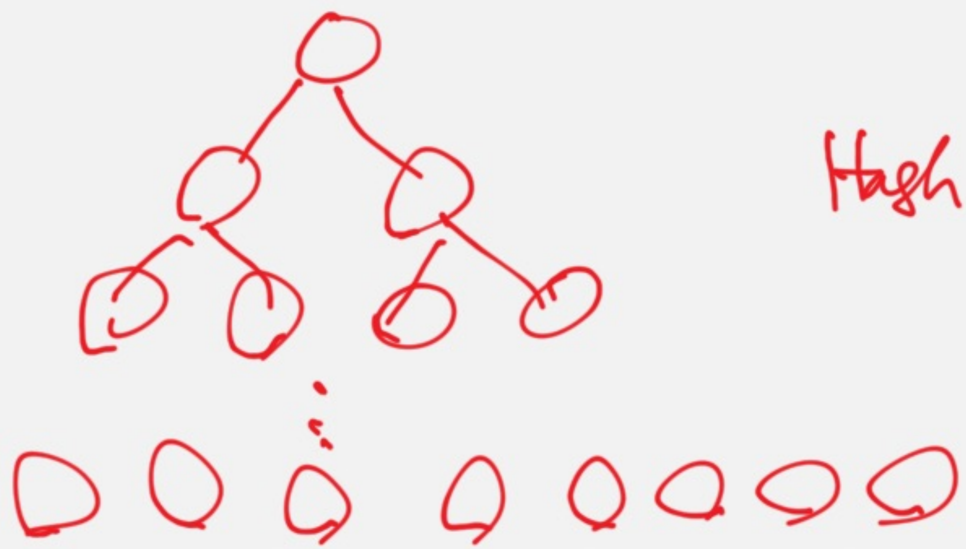
add $\boxed{a_1, a_2, \cdots, a_{k-1}, a_k, b_k}$ to $C_{k+1}$.

CANDIDATE PRUNING USING Hash tree.

If $Z = (a_1, a_2, \dots, a_{k+1}) \in C_{k+1}$,

then for every $k$-subset $Q$ of $Z$

Check if $Q \in F_k$.



Hash tree

Compute the support for every $Q \in C_{k+1}$.

Hash tree can be used here as well.

---

Generation of Rules:

Let $X$ be FREQ. & Let $X' \subseteq X$, $X' \neq \phi$,

Consider the Rule: $X' \rightarrow X - X'$.

# A RANDOMIZED ALG.

Pick a ~~Sample~~ Random S.

Use a threshold of $(1-\epsilon)$ minSupport

Find all the Freq. itemsets in S.

Output those that are ~~FREQ~~ in DB.

## Computing $F_1$:

Bit array.

transaction

For every item $i \in I$ do

Compute the support of $i$;

if it is $\geq$ min support output $i$;

Total Run time = $O(d \cdot n)$

## POLYNOMIAL ARITHMETIC:

How fast can we multiply two degree-$n$ polynomials?

We can multiply in $O(n^2)$ TIME.

Polynomials are supplied in coefficients FORM.

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

VALUE FORM:

Pick $(n+1)$ distinct points

$u_0, u_1, u_2, \ldots, u_n$.

$(u_0, f(u_0)), (u_1, f(u_1)), \ldots, (u_n, f(u_n))$.

$$(c_0, f(c_0)), (c_1, f(c_1)), \ldots, (c_{2n}, f(c_{2n}))$$

$$(c_0, g(c_0)), (c_1, g(c_1)), \ldots, (c_{2n}, g(c_{2n}))$$

$\Downarrow$ PRODUCT

$$(c_0, f(c_0)g(c_0)), (c_1, f(c_1)g(c_1)), \ldots, (c_{2n}, f(c_{2n})g(c_{2n})).$$

PERFORM EVALUATION & INTERPOLATION

USING $N^{th}$ ROOTS OF UNITY.

$1, \omega, \omega^2, \ldots, \omega^{N-1}$

$\omega$ = PRIMITIVE $N^{th}$ ROOT OF UNITY

$$= e^{2\pi i/N} = \cos\left(\frac{2\pi}{N}\right) + i \sin\left(\frac{2\pi}{N}\right).$$

Theorem: we can multiply two degree-$n$ polynomials in $O(n \log n)$ time.

MACHINE LEARNING:

$$f: \mathbb{R}^m \to \mathbb{R}^n.$$

INPUT: Examples: $(\vec{x}_1, f(\vec{x}_1))$, $(\vec{x}_2, f(\vec{x}_2))$, ..., $(\vec{x}_m, f(\vec{x}_m))$.

Output: A Guess for $f$.

Each guess is a Model.

# LINEAR REGRESSION:

$$f: \mathbb{R}^n \to \mathbb{R}.$$

$$f(x_1, x_2, \dots, x_n) = w_1 x_1 + w_2 x_2 + \dots + w_n x_n$$

Model parameters.

## Examples:

$$(x_1^1, x_2^1, \dots, x_n^1, y_1)$$

$$\vdots$$

$$(x_1^m, x_2^m, \dots, x_n^m, y_m).$$

## GRADIENT DESCENT:

$$f(x+\epsilon) \cong f(x) + \epsilon f'(x).$$

**Note:**

$$\boxed{f(x-\epsilon\ \mathrm{sign}(f'(x))) \leq f(x)}$$

**Case 1:** $f'(x)$ is +ve:

$$f(x-\epsilon) \cong f(x) - \epsilon f'(x) \leq f(x).$$

**Case 2:** $f'(x)$ is −ve:

$$f(x+\epsilon) \cong f(x) + \epsilon f'(x) \leq f(x).$$

ITERATIVE Alg.

Start with $x_0$; $i=0$;

Repeat

$i = i+1$;

$x_i = x_{i-1} - \epsilon \,\text{sign}\, f'(x_{i-1})$;

Until Convergence $\rightarrow$ STATIONARY POINT

$\rightarrow f'(x) = 0$.

$$X = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_n^1 \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & & & \\ x_1^m & x_2^m & \cdots & x_n^m \end{bmatrix}; \quad \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

Model output: $X\vec{w}$

$$w = \begin{bmatrix} w_1 & w_2 & \cdots & w_n \end{bmatrix}^T$$

We want to Minimize $\boxed{\frac{1}{m}\|X\vec{w}-\vec{y}\|_2^2}$ C

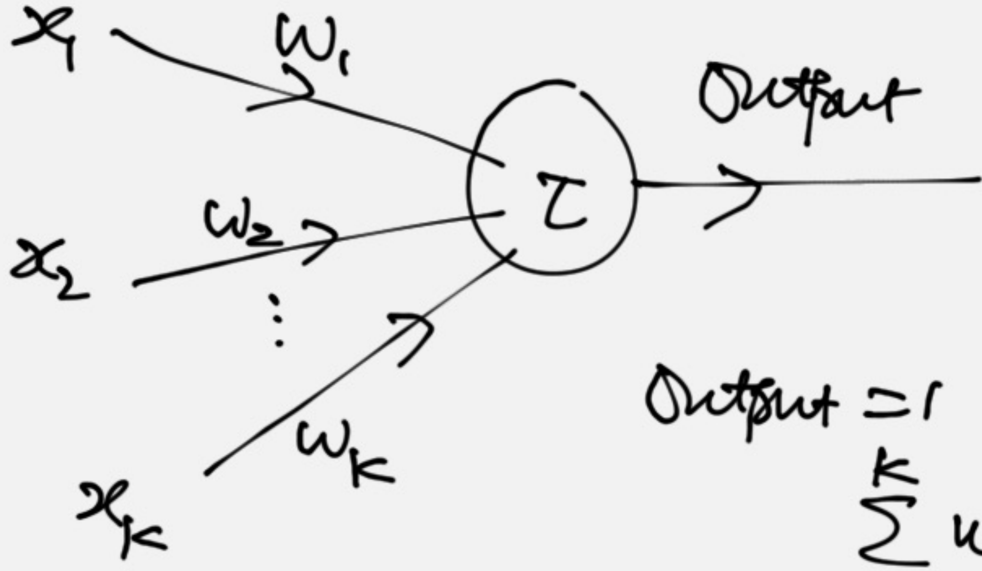
Loss function.

$$\nabla_{\vec{w}} C = 0; \qquad C = (X\vec{w}-\vec{y})^T(X\vec{w}-\vec{y})$$

$$\boxed{\vec{w} = (X^TX)^{-1}X^T\vec{y}}$$

RUN TIME $= O(n^2 m + n^3)$.

## PERCEPTRONS:



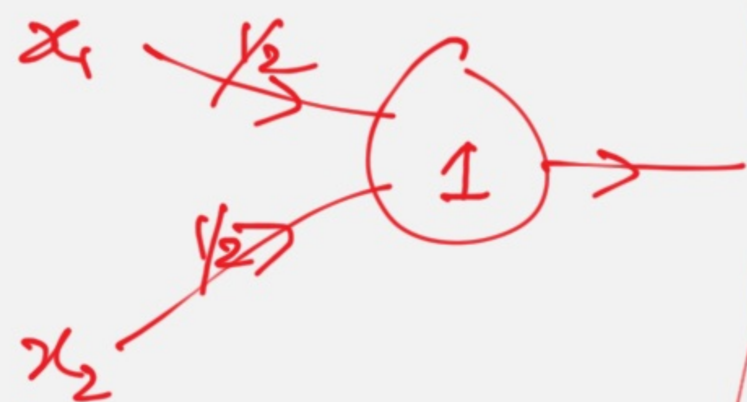

Output = 1 if

$$\sum_{i=1}^{K} w_i x_i \geq \tau$$

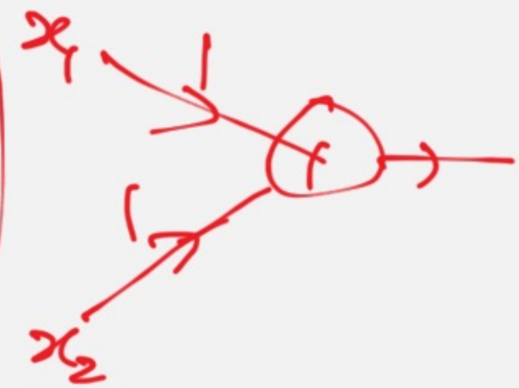
= 0 otherwise.

FACT. Any Boolean Function Can be realized as a Multilevel Perceptron.

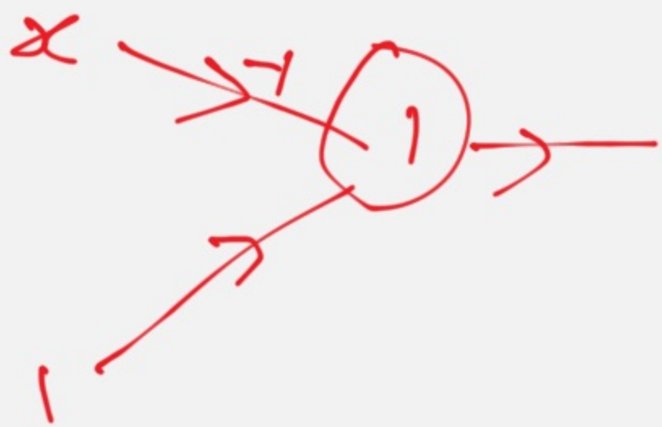
Boolean AND:



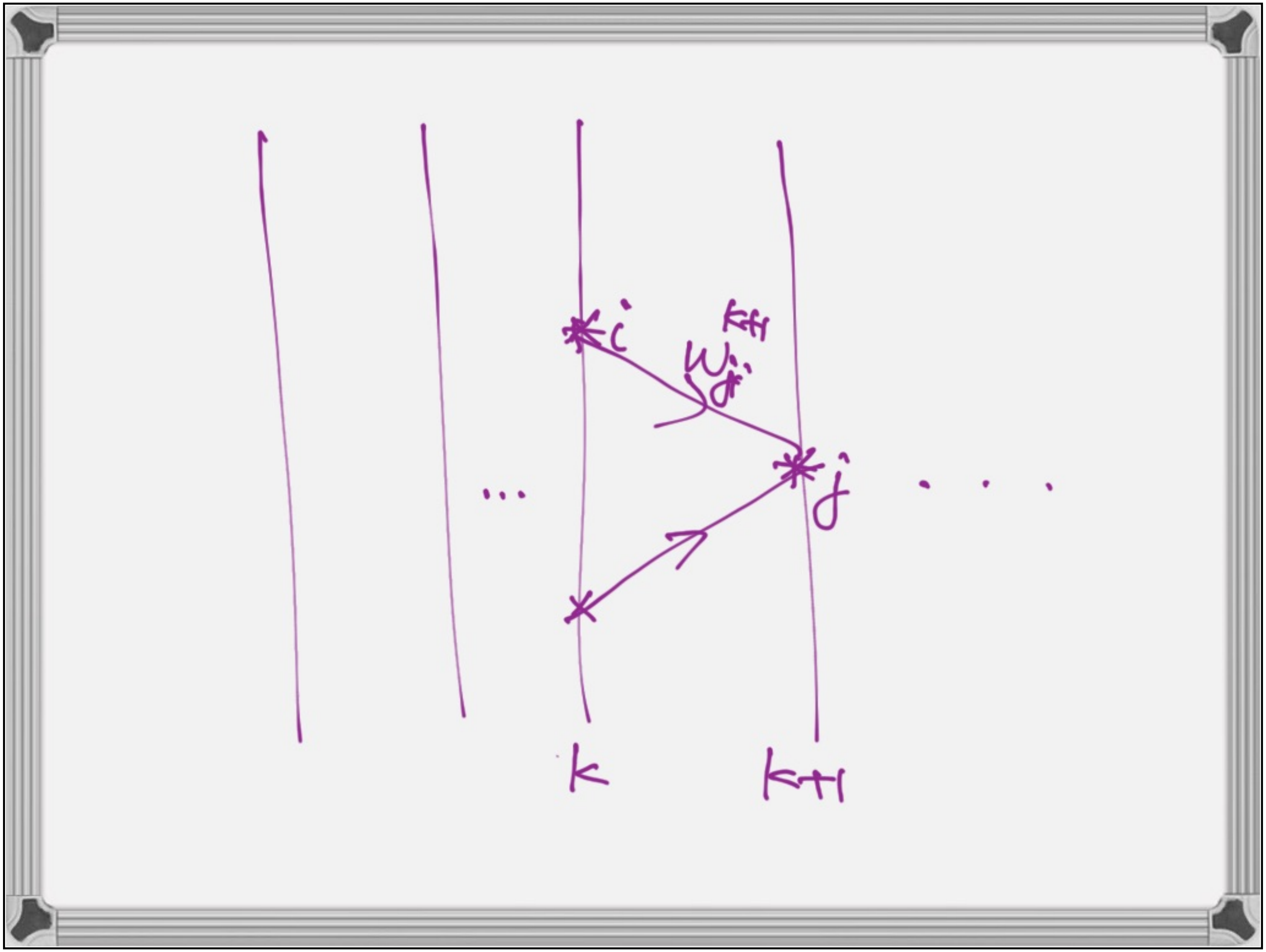

Boolean OR:



Not:



# Neural Networks:


i
$w_{ji}^{k+1}$
j
...
. . .
k
k+1

$$z_j^{k+1} = \sum_{i=1}^{n_k} w_{ji}^{k+1} a_i^k$$

$a_i^k \rightarrow$ output of node $i$ at level $k$.

$$a_i^k = \sigma\left(z_i^k\right)$$

$\sigma \rightarrow$ Activation Function.

FACT.

FORWARD & Back Propagation Can be done in:

1. $O(n^2 L)$ time, $n$ = # of Neurons in each layer

OR

2. $O(|V| + |E|)$ time

where $G(V, E)$ is the Neural Network.

# Model Exam:

(1) Total # of items in all of the transactions = $O(n)$.

Pick a Random Sample $S$ of $O(\log n)$ transactions.

Identify items Frequent in the Sample with a support of $\frac{1}{8}$.

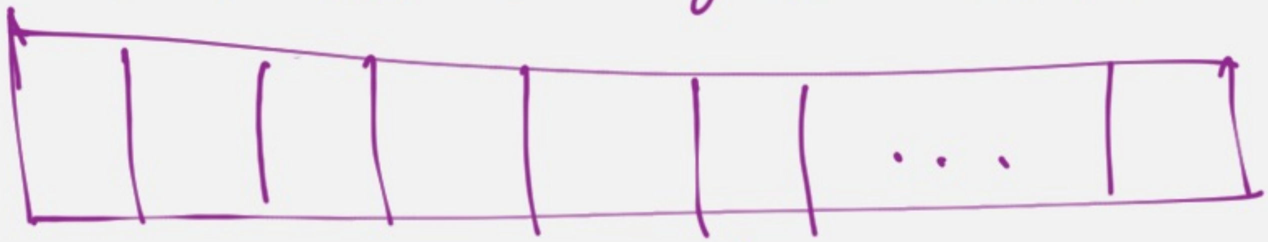
Output the Freq. items;

(2) Generate all the possible $k$-itemsets. There are $\binom{d}{k} = O(d^k)$ of them.

Compute the support for each $k$-itemset.

Assume that $\exists$ $n$ proc. per itemset.

Have a bit array of size n:




B

One for every transaction.

For $1 \le i \le n$ in || do

If $t_i$ contains the itemset,

set $B[i]=1$.

All the n proc. perform a prefix sums on $B[1]$, $B[2]$, $\ldots$, $B[n]$.

3. **For** $1 \le i \le cn$ **do**

    Evaluate $f(i)$. If $f(i) = 0$ then output $i$;

---

4. Boolean AND