APRENDIZAJE BASADO EN GRAFOS

Jesús González y Eduardo Morales

Aprendizaje Basado en Grafos

- Funciona para dominios estructurados
  - Como ILP
- Algunos dominios estructurados
  - Bioinformática
  - Visión por computadora
  - Recuperación de texto
  - Análisis de archivos web-logs

Aprendizaje Basado en Grafos

- Encuentran subestructuras a partir de un conjunto de grafos
  - Caracterizar conjuntos de grafos
  - Discriminar diferentes grupos de grafos
  - Clasificar y agrupar grafos
  - Construir índices de grafos

Aprendizaje Basado en Grafos

- 2 principales formas de buscar subestructuras frecuentes
  - Con generación de candidatos
  - Sin generación de candidatos

Aprendizaje Basado en Grafos

- Generación de Candidatos
  - Genera subestructuras candidatas
  - Verifica la frecuencia de esas subestructuras
  - Como lo hace el alg. de Agrawal para encontrar reglas de asociación
  - Realizan operación "join" entre 2 o más subestructuras frecuentes para generar una nueva subestructura candidata
  - Búsqueda BFS para generar todos los candidatos de tamaño k
  - Dos pasos muy tardados
    - Unir dos grafos frecuentes de tamaño-k para generar grafos candidatos de tamaño-k+1
    - Verificar la frecuencia de los candidatos en otro paso
    - Cuello de botella para algoritmos tipo a priori o generación de candidatos
  - Ejemplos: Algoritmos AGM, FSG, path-join

Aprendizaje Basado en Grafos

Tabla 7.1: Apriori-like

Apriori$(D, min\_support, S_k)$
  $S_{k+1} \leftarrow \emptyset$
  for each frequent $g_i \in S_k$ **do**
    for each frequent $g_j \in S_k$ **do**
      for each size $(k+1)$ graph g formed by the merge of g_i and g_j **do**
        if g is frequent in D and $g \notin S_{k+1}$ **then**
          insert g to S_{k+1};
  if $S_{k+1} \neq \emptyset$ **then**
    call Apriori$(D, min_support, S_{k+1})$;
  return
end

Aprendizaje Basado en Grafos

- Sin generación de candidatos
  - Evitar cuello de botella de alg. que generan candidatos
  - Utilizan método en que van creciendo el patrón
  - Extienden un patrón a partir de un solo grafo
  - Al extender el patrón ya se está seguro de que es frecuente, ya no se generan candidatos
  - Pueden utilizar BFS ó DFS
  - Ejemplos: gSpan, MoFa, FFSM, SPIN, Gaston
  - Muy importante: paso de crecimiento de patrones podando lo más posible el espacio de búsqueda

Aprendizaje Basado en Grafos

Tabla 7.2: Pattern Growth

PatternGrowth$(g, D, min\_support, S)$
Input: A frequent graph g, a graph dataset D, and $min\_support$.
Output: A frequent substructure set S.
    if $ginS$ **then return**;
    else insert g to S;
    scan D once, find all the edges e such that g can be extended to $g \otimes e$;
    for each frequent $g \otimes e$ **do**
        Call PatternGrowth$(g \otimes e, D, min_suppport, S)$;
    return
end

Introducción a Subdue

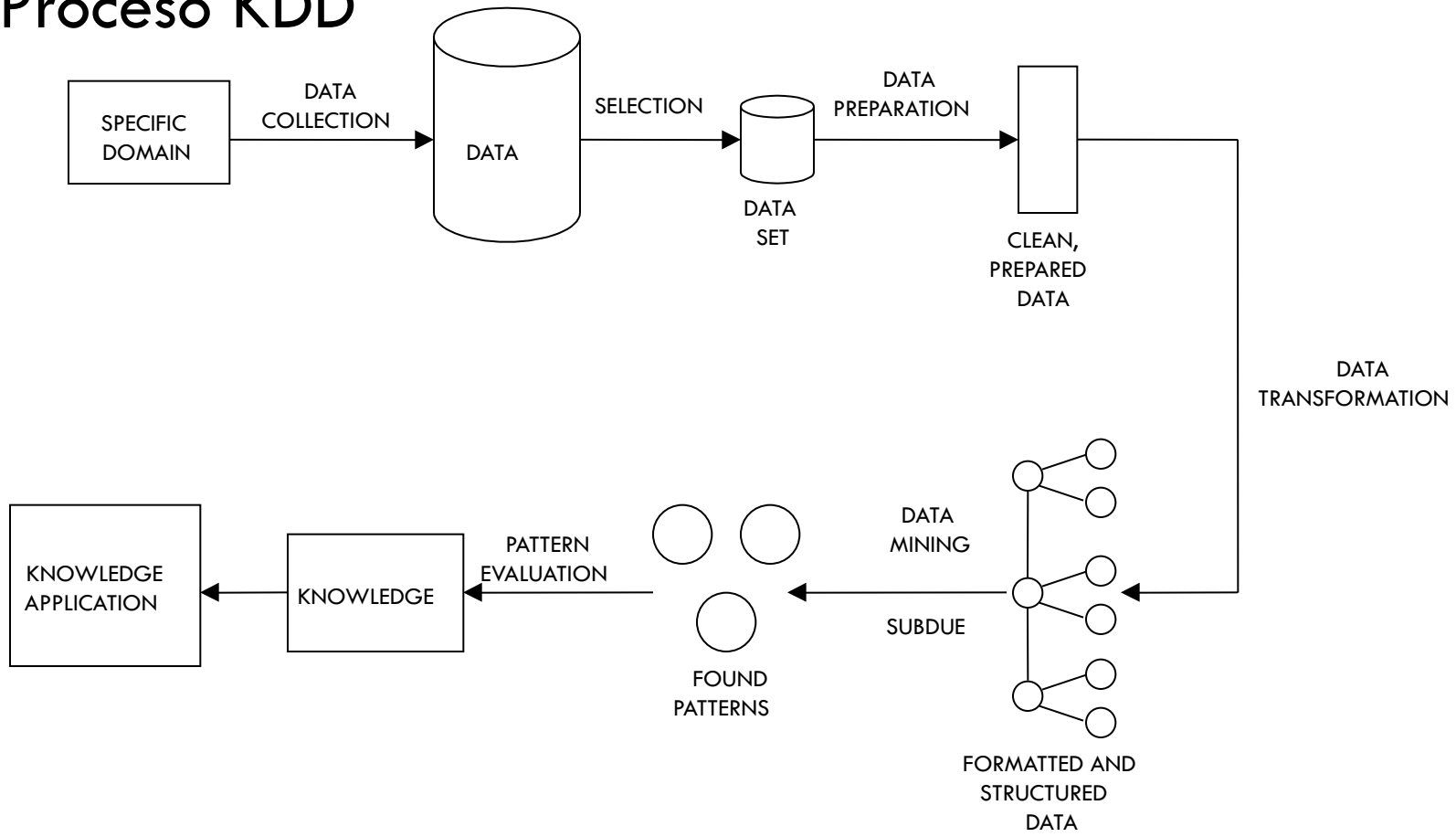
- Representación de conocimiento
  - Basado en grafos
  - Fase de preparación de datos
    - Transformación de los datos a un formato de grafo
- Espacio de búsqueda
  - Todos los subgrafos que se pueden derivar a partir del grafo de entrada
    - Espacio de búsqueda es exponencial
    - También el tiempo de ejecución
    - Se puede restringir para hacerlo polinomial (pero ya no completo)

Introducción a Subdue

- Criterio de evaluación
  - Para determinar cuáles subgrafos del espacio de búsqueda son relevantes y formar parte de los resultados
  - Subdue utiliza el "Principio de Longitud de Descripción Mínima" ó "MDL"
    - MDL dice que la mejor descripción del conjunto de datos es aquella que minimiza la longitud de la descripción de todo el conjunto de datos
    - Subdue lo utiliza para determinar que tan bien un grafo comprime al grafo de entrada

Introducción a Subdue

- Proceso KDD

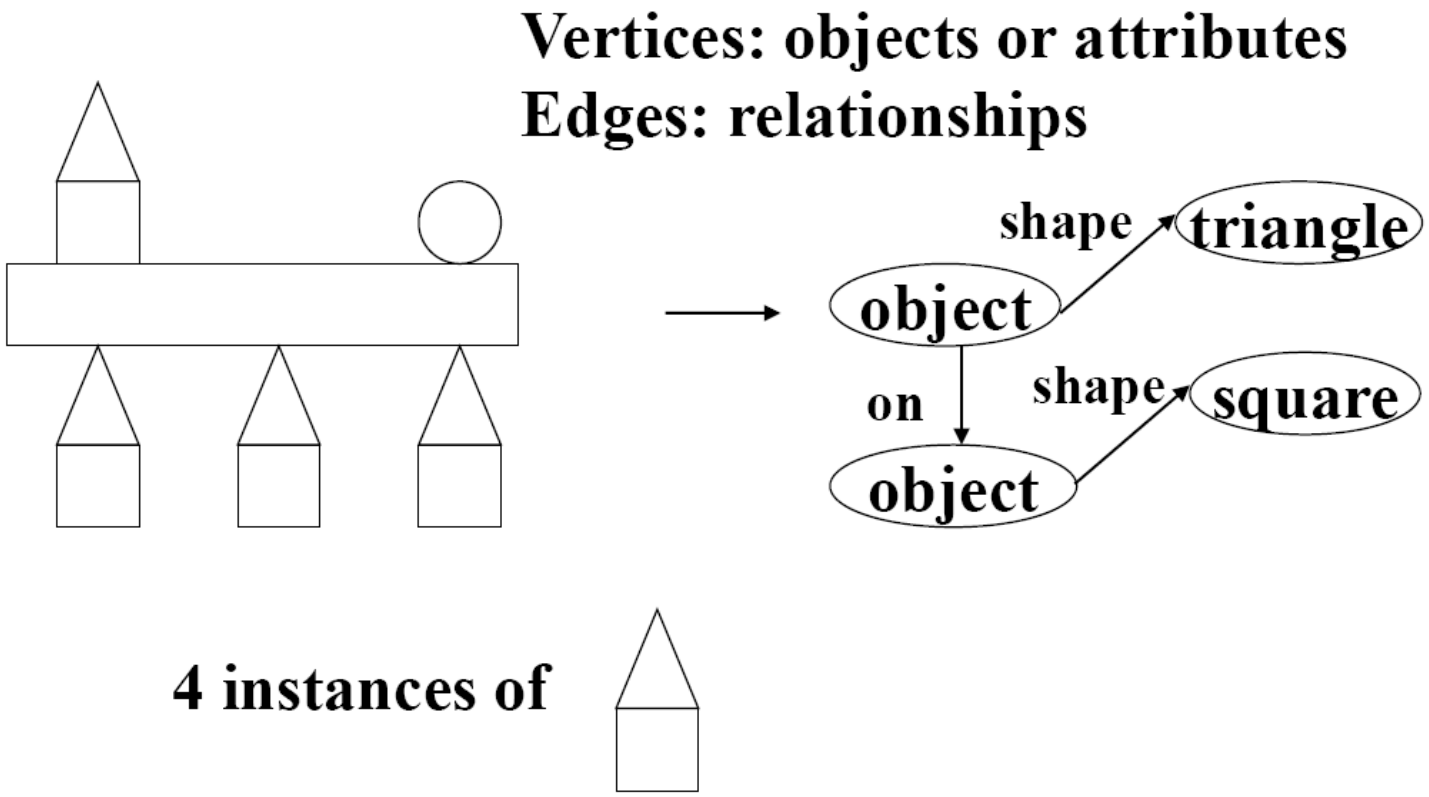


Introducción a Subdue

- Versión no supervizada
  - Descubre patrones (subestructuras) en conjuntos de datos estructurales
  - Representa los datos como un grafo etiquetado
  - Entrata: Vértices y arcos
  - Salida: Patrones descubiertos y sus instancias

Introducción a Subdue

- Ejemplo



Búsqueda de Subdue

- Inicia con un solo vértice por cada etiqueta diferente y lo expande
  - Añade un arco y un vértice
  - Añade un arco (entre 2 vértices de la subestructura)
- Restringido computacionalmente
  - Beam Search
- Espacio de búsqueda
  - Todos los sub-grafos del grafo de entrada
- Guiado por heurísticas de compresión

MDL en Subdue

- Codificación mínima (Minimum Encoding)
  - Basada en MDL (Rissanen 1989)
- La mejor descripción del conjunto de datos es la que minimiza I(S) + I(G|S)
  - S es la subestructura utilizada para describir el grafo de entrada G
  - I(S) es la longitud (número de bits) requeridos para codificar S
  - I(G|S) es la longitud de la codificación del grafo G después de comprimirlo utilizando la subestructura S

MDL en Subdue

- La conectividad del grafo se representa con una matriz de adyacencia "*A*" de tamaño *n* x *n*
  - Valores *A*[*i*,*j*] corresponden a la conectividad del grafo
    - Arcos no dirigidos se guardan en una sola entrada de la matriz
      - *A*[*i*,*j*] = 1 si los vértices *i* y *j* están conectados
      - *A*[*i*,*j*] = 0 si no lo están
- Hay una tabla que contiene las l_u etiquetas únicas del grafo G

MDL en Subdue

- El número de bits requerido para codificar las etiquetas de los vértices del grafo es:
  - $vbits = \lg v + v \lg l_u$
    - $\lg v$ es el # de bits para codificar el # de vértices v en el grafo
    - $v \lg l_u$ es el # de bits requerido para codificar las etiquetas de todos los v vértices del grafo de entrada

MDL en Subdue

- El número de bits requerido para codificar los renglones de la matriz de adyacencia *A* es:
  - $$rbits = \lg(b+1) + \sum_{i=1}^{v} \lg(b+1) + \lg\binom{v}{k_i}$$
    - donde cada renglón *i* tiene k_i 1's y $(n - k_i)$ 0's, y $k_i << (n - k_i)$
    - *b* es el número máximo de 1's en un renglón
    - Para codificar el valor de k_i necesitamos $\lg(b + 1)$ bits
    - El número de posibilidades para acomodar k_i 1's en un renglón de longitud *v* esta dado por

$$\binom{v}{k_i} = \frac{v!}{k!(v-k)!}$$

MDL en Subdue

- El número de bits requeridos para codificar los arcos de la matriz *A* representada por las entradas *A*[*i*,*j*] = 1 esta dado por
  - $$ebits = \lg m + \sum_{i=1}^{v} \sum_{j=1}^{v} A[i,j](\lg m + e(i,j)[1 + \lg l_u])$$
  - $$ebits = \lg m + e(1 + \lg l_u) + \sum_{i=1}^{v} \sum_{j=1}^{v} A[i,j] \lg m$$
  - $$ebits = e(1 + \lg l_u) + (K+1) \lg m$$

MDL en Subdue

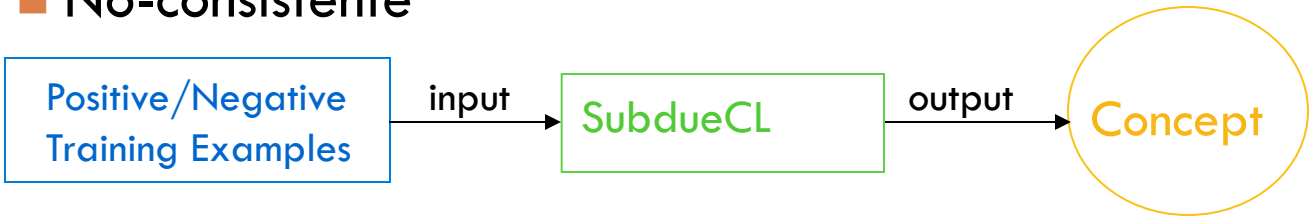
- e(i,j) es el # de arcos entre los vértices i y j en el grafo y la matriz A (arcos no dirigidos se cuentan sólo una vez
- m es el máximo e(i,j)
- e es el número de arcos en el grafo
- K es el número de 1's en la matriz de adyacencia A
- Para cada par de vértices i, j necesitamos lg$m$ bits para codificar el número de arcos entre ellos
- Necesitamos [1+lg l_u] bits por cada arco para codificar la etiqueta del arco y si el arco es dirigido o no dirigido
- También requerimos codificar el número de bits lg$m$ para especificar el número de arcos por entrada

MDL en Subdue

- El número total de bits necesarios para codificar el grafo G esta dado por:
  - *vbits* + *rbits* + *ebits*
- Cada que subdue encuentra una subestructura la evalúa utilizando el principio MDL
  - Compara el valor con el de otras subestructuras
  - Elige la que mejor comprime el grafo en términos de MDL
  - Subdue reemplaza las instancias de las subestructuras por un vértice que representa esa subestructura
  - Inicia una nueva iteración
    - Nuevas subestructuras podrían contener otras encontradas previamente

SubdueCL (Concept Learning)

- Versión de aprendizaje de conceptos de Subdue
  - Dos versiones
    - Consistente (cuestiones teóricas)
    - No-consistente



  - Criterio de evaluación: "Set-Covering"

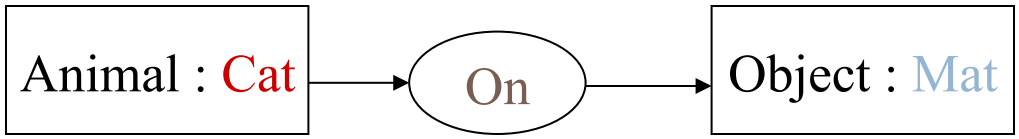
$$Value = 1 - \frac{\#\,PosEgsNotCovered + \#\,NegEgsCovered}{\#\,PosEgs + \#\,NegEgs}$$

SubdueCL (Análisis Teórico)

- Grafos conceptuales
- Análisis PAC learning
- Comparación con método de “Galois Lattice”

SubdueCL (Grafos Conceptuales)

- Grafos Conceptuales
  - Representación de conocimiento basada en lógica derivada de las redes semánticas y los grafos existenciales de Pierce
  - Tienen vértices para conceptos y relaciones



  - Traducción directa a lógica
    - onRelation(cat, mat).

PAC Learning

- PAC Learning
  - Determina si una clase de conceptos C se puede aprender en tiempo polinomial a partir de ejemplos

$$m \geq \frac{1}{\varepsilon}(\ln |H| + \ln(\frac{1}{\delta}))$$

  - Asumimos
    - El alg. de aprendizaje corre en tiempo polinomial
    - Algoritmo consistente
    - Ejemplos de entrenamiento y prueba tomados de acuerdo a la misma distribución de probabilidad *D*

PAC Learning con SubdueCL

- Espacio de búsqueda de SubdueCL
  - Peor caso – cota superior: 2^{n^2}

$$m \geq \frac{1}{\varepsilon}(n^2 \ln 2 + \ln(\frac{1}{\delta}))$$

  - Operaciones de subgrafo-isomorfo restringidas a tiempo polinomial
    - No 100% preciso
  - SubdueCL usa beam search → definir el beam infinito
  - Teóricamente no podemos “PAC learn” con SubdueCL

Galois Lattice

- Marco formal para aprendizaje de conceptos basado en grafos
  - Concepto: grafos de descripción y ejemplos cubiertos por esos grafos de descripción
  - Construye una “lattice” (o celosía, o malla) utilizando generalización
  - Mantiene la “lattice” ordenada especialización/ generalización
  - Permite grafos desconectados
  - Mantiene sólo grafos núcleo
  - Restringido a utilizar grafos localmente-inyectivos

SubdueCL como una Galois Lattice

- Marco formal para aprendizaje de conceptos basado en grafos
  - Concepto: grafos de descripción y ejemplos cubiertos por esos grafos de descripción
  - Construye una "lattice" utilizando "least-general-generalization"
  - Mantiene la "lattice" ordenada especialización/ generalización
  - No permite grafos desconectados
  - No mantiene sólo grafos núcleo
  - No restringido a utilizar subclases de grafos

SubdueCL como una Galois Lattice

- ¿Modificar SubdueCL para satisfacer lo que asume "Galois Lattice"?
  - ¿Usar sólo grafos núcleo?
  - ¿Permitir grafos desconectados?
  - Sub-isomorfismo de grafos: polinomial para subclases de grafos?

- No, SubdueCL es exitoso en dominios complejos

Análisis Empírico

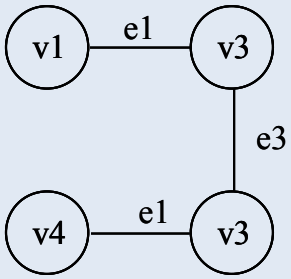
- Experimentos con un dominio artificial
- Comparación con sistemas ILP FOIL y Progol
  - Dominios planos
    - Golf, Vote, Diabetes y TTT
  - Dominios relacionales
    - Ajedrez y cáncer
  - Experimentos con el dominio del Web

Dominio Artificial

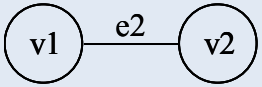
- Generar grafos conteniendo un concepto y una parte aleatoria
  - Utilizando grafos normales y conceptuales
  - Estudiar el desempeño de SubdueCL
    - Tamaño de los grafos
    - Densidad de los grafos
    - 10-Fold Cross Validation
    - Curva de aprendizaje

Resultados Experimentales

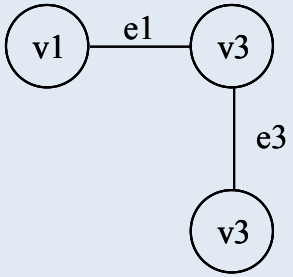
- Prueba #5, grafos con 20 vértices y 40 arcos (pequeños)



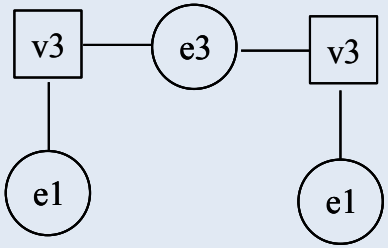
Positive sub



Negative sub



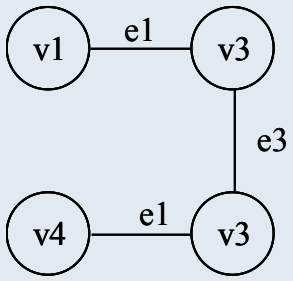
Normal graph result



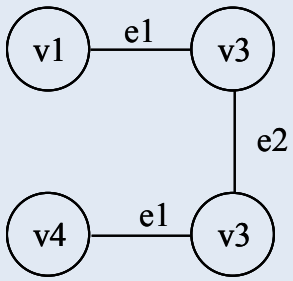
Conceptual graph result

Resultados Experimentales

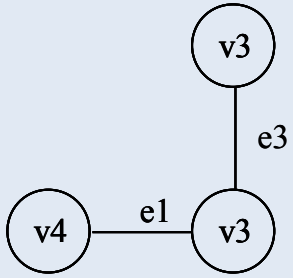
- Prueba #6, grafos con 20 vértices y 40 arcos (pequeños)



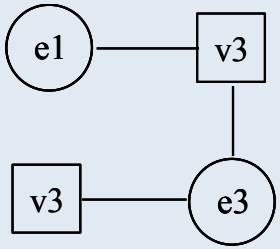
Positive sub



Negative sub



Normal graph result



Conceptual graph result

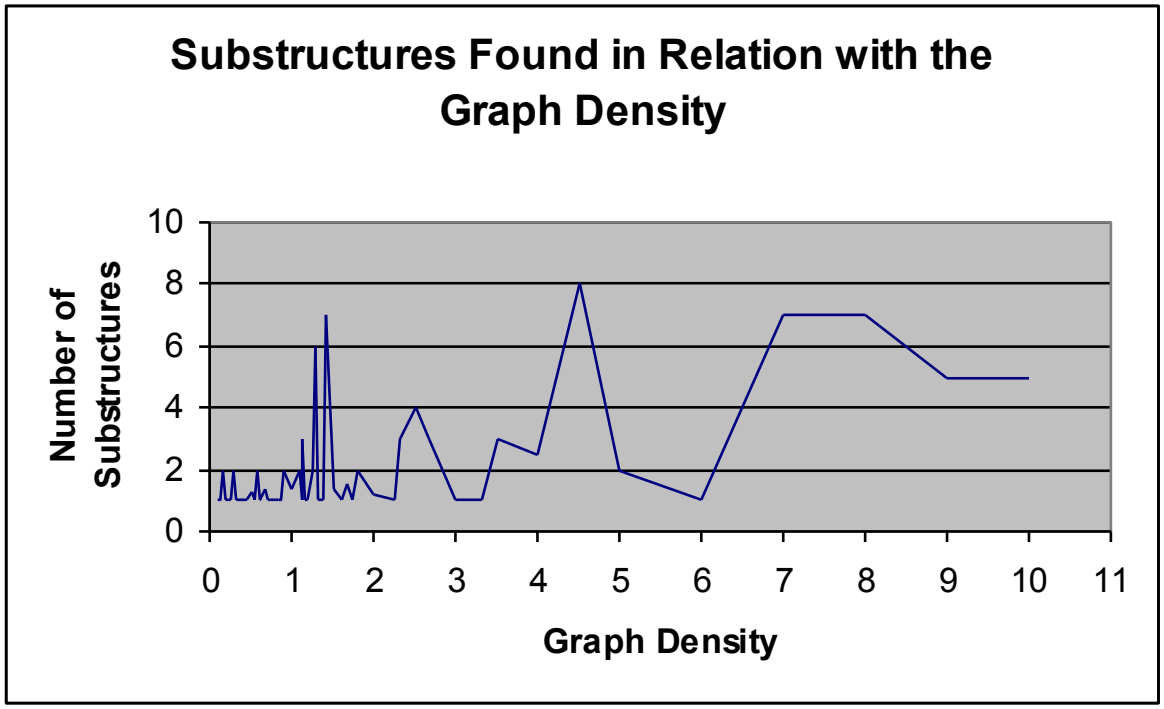
Resultados Experimentales

- Resultados similares utilizando grafos normales y conceptuales
- Resultados de grafos conceptuales son sub-grafos de los grafos normales en la mayoría de las pruebas
- Podemos utilizar grafos conceptuales para la comparación con sistemas ILP

Efecto Densidad de Grafos en SubdueCL

- Densidad: #arcos / #vértices
- Variar la densidad ayuda a saber como se comporta SubdueCL al introducir ruido
- Ruido: La subestructura positiva se reproduce en los ejemplos negativos

Efecto Densidad de Grafos en SubdueCL



- Es más probable que SubdueCL produzca más subestructuras al incrementarse la densidad de los grafos

10-Fold Cross Validation

- Usar los grafos de las pruebas #5 y #6
- 100 ejemplos de entrenamiento, 50 pos y 50 neg
- Grafos pequeños
  - 20 vértices y 40 arcos, grafos normales
  - 60 vértices y 80 arcos, grafos conceptuales
- Grafos grandes
  - 50 vértices y 100 arcos, grafos normales
  - 150 vértices y 200 arcos, grafos conceptuales

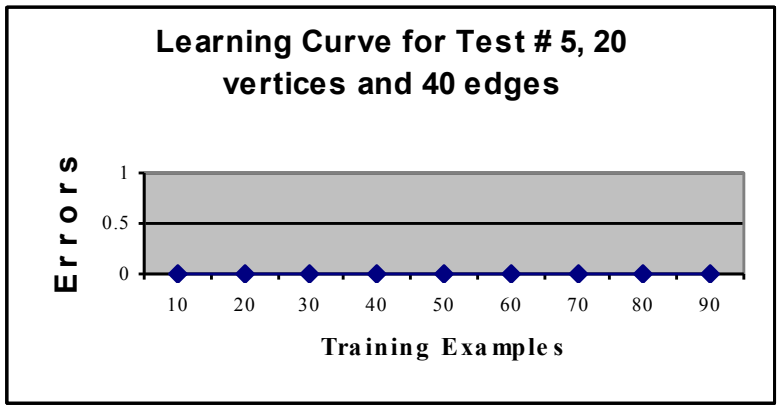
10-Fold Cross Validation

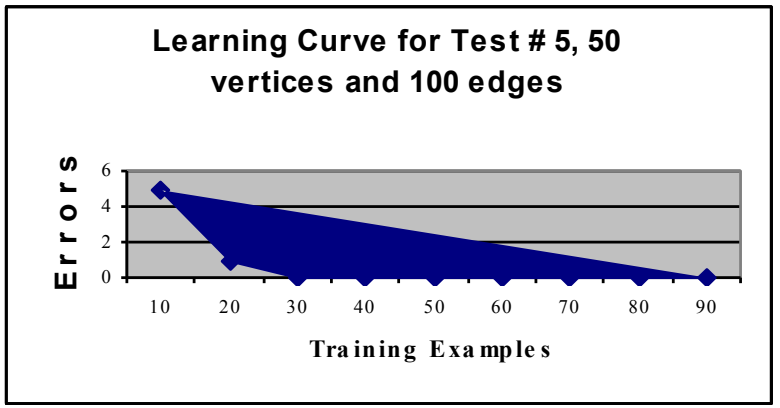
Graph	Test	#Errors	Accuracy	ε	m δ=0.1	m δ=0.05
Normal	5 small	10/0	90%	.10	2,796	2,803
	5 large	5/3	92%	.08	21,691	21,700
	6 small	2/8	90%	.10	2,796	2,803
	6 large	6/0	94%	.06	28,922	28,934
Conceptual	5 small	9/4	87%	.13	19,208	19,214
	5 large	9/3	88%	.12	129,994	130,000
	6 small	20/36	44%	.56	4,460	4,462
	6 large	33/17	50%	.50	31,199	31,200

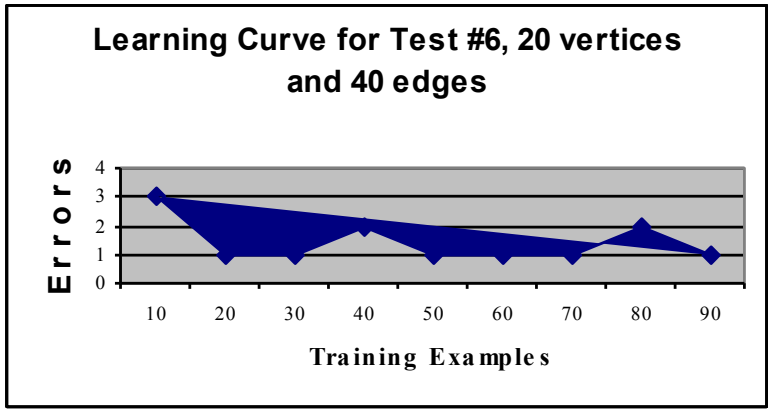
Curva de Aprendizaje

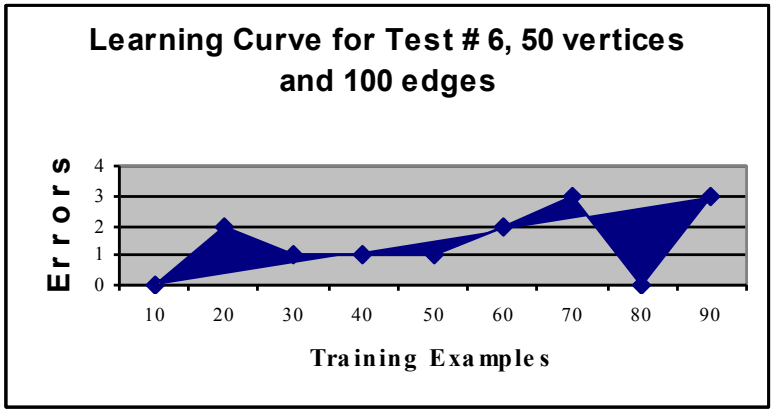
- Evaluar la curva de aprendizaje de SubdueCL
- Entrenar SubdueCL con 10, 20, …, 90 ejemplos y probar con un conjunto diferente de 10 ejemplos
- ¿Cuántos ejemplos necesita SubdueCL para aprender?

Curva de Aprendizaje





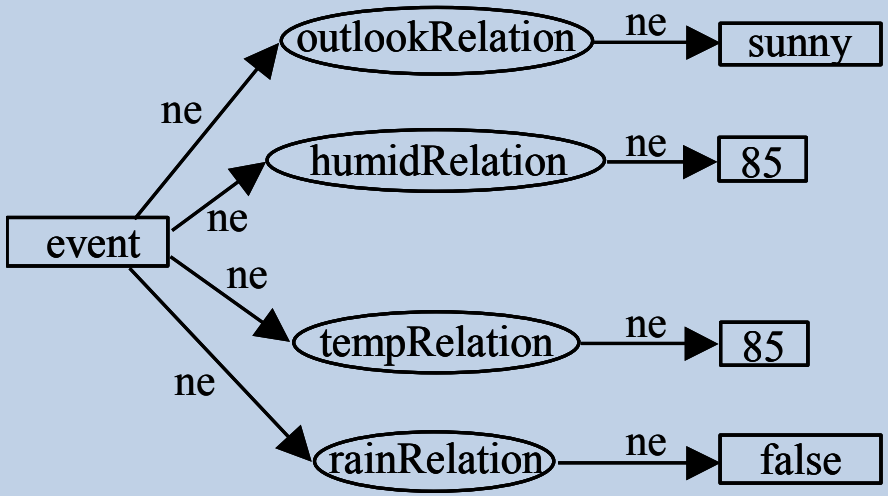




- Errores: # de ejemplos de prueba mal clasificados

Comparación con Sistemas ILP

- Representación



Conceptual Graph Representation
SubdueCL

```
outlookRelation(1,event,sunny).
humidityRelation(1,event,85).
tempRelation(1,event,85).
rainRelation(1,event,false).
```

Logic Representation
Progol

Comparación con Sistemas ILP en Dominios No-Relacionales

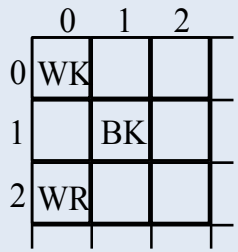
- Precisión en dominios no relacionales

	Golf	Vote	Diabetes	Credit	T-T-T
FOIL	66.67	93.02	70.66	68.60	100.0
Progol	66.67	94.19	63.68	63.20	100.0
SubdueCL	66.67	94.65	65.79	63.50	100.0

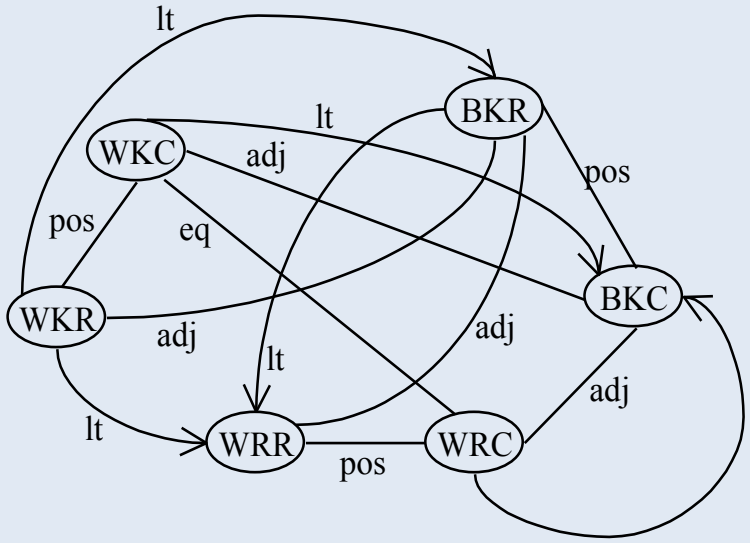
- SubdueCL es competitivo con sistemas ILP
- Mejor en dominios que no tienen valores continuos

Comparación con Sistemas ILP en Dominios Relacionales

- Dominio del Ajedrez



(a)

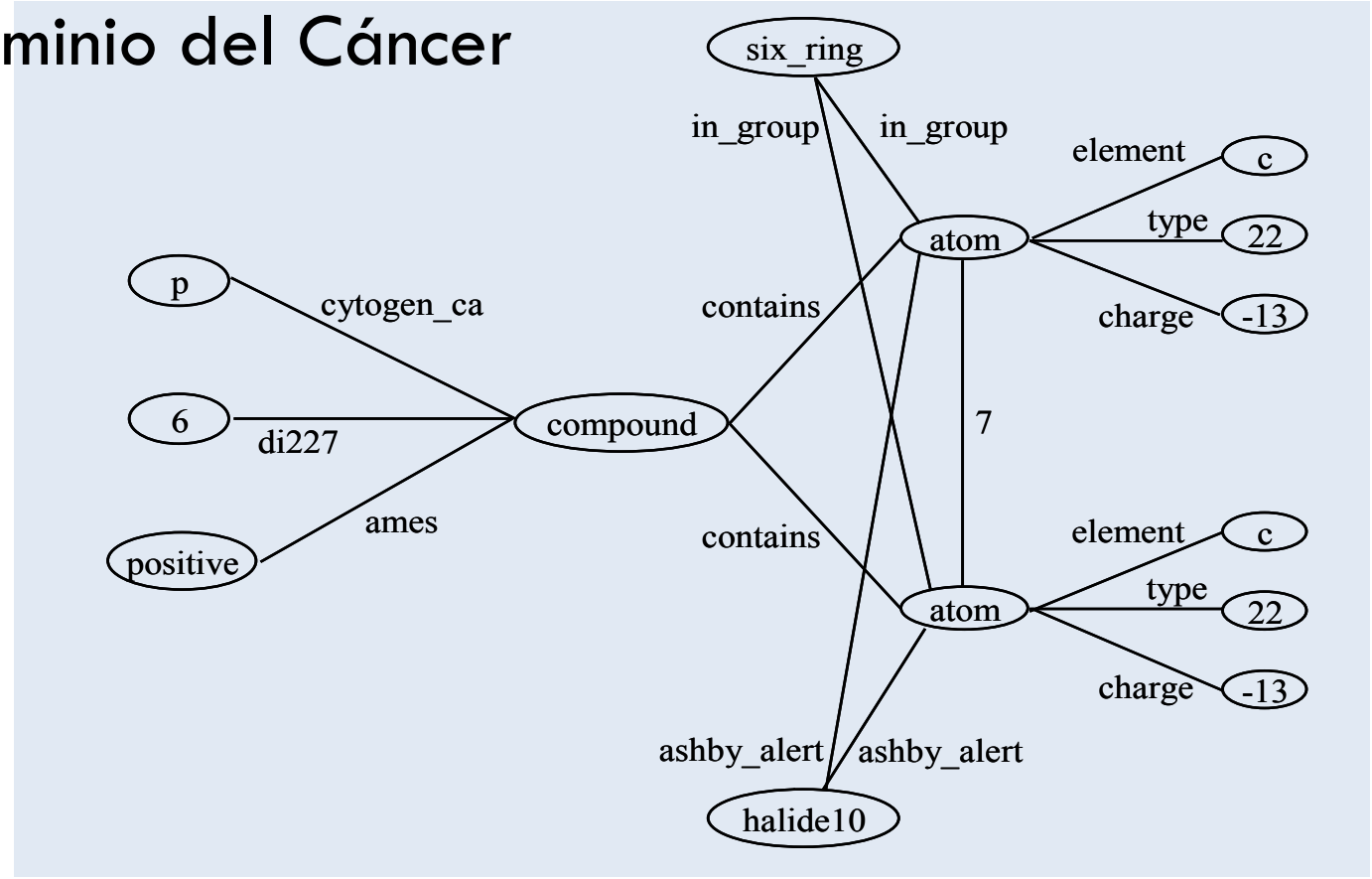


(b)

- Un ejemplo del dominio del ajedrez. (a) Configuración del tablero y (b) Representación gráfica del ejemplo
- Precisión: Progol: 99.74%, FOIL: 99.34%, SubdueCL: 99.74%

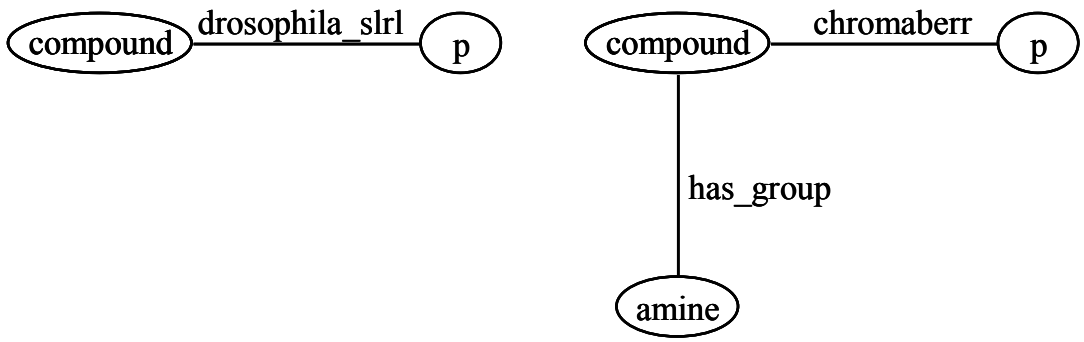
Comparación con Sistemas ILP en Dominios Relacionales

- Dominio del Cáncer



Comparación con Sistemas ILP en Dominios Relacionales

- Dominio del Cáncer
  - Precisión
    - Progol: 64% (reportado 72%)
    - SubdueCL: 61.54%
- SubdueCL no consideró información sobre mutagénesis que Progol utilizó
- 2 Subestructuras encontradas en el dominio del Cáncer



Experimentos con el Dominio del Web

- Dominio Web
  - Tres opciones para el contenido de los grafos
    - Estructura de hiperligas
    - Estructura de hiperligas + títulos de páginas
    - Estructura de hiperligas + contenido de páginas

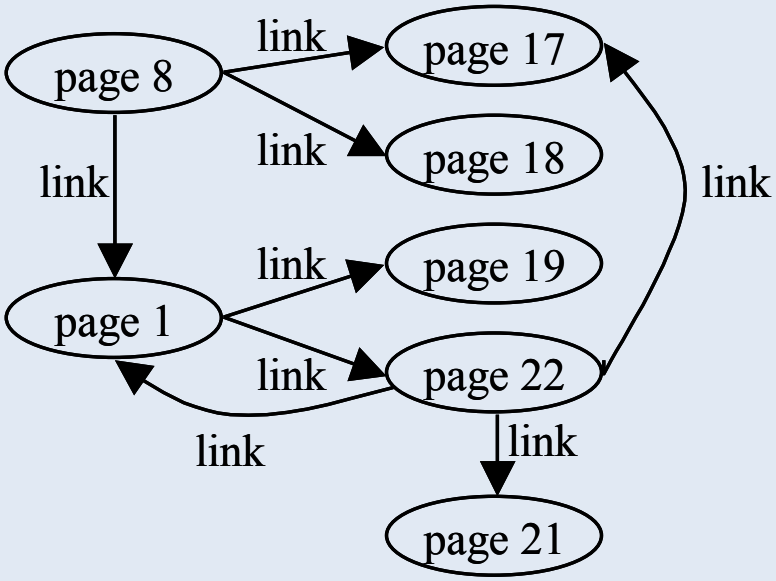
Experimentos con el Dominio del Web

- Experimento 1: Estructura de hiperligas + contenido de las páginas
  - Ejemplos positivos: Páginas web de profesores
  - Ejemplos negativos: Páginas web de estudiantes

  - Subestructura encontrada en el dominio Web (Profesores-Estudiantes)

Experimentos con el Dominio del Web

- Experimento 2: Estructura de hiperligas
  - Ejemplos positivos: Tiendas de computadoras
  - Ejemplos negativos: Páginas web de profesores



Node 8 represents a category of products linked to 3 subcategories of products represented by nodes 1, 17, and 18. Nodes 19 and 22 represent either more specific subcategories derived from the subcategory represented by node 1 or specific products. Node 21 represents a specialization of the subcategory represented by node 22.

Conclusiones

- Análisis Teórico
  - El espacio de búsqueda de SubdueCL es: 2^{n^2}
  - Es posible "PAC Learn" grafos simples utilizando nuestro método
  - SubdueCL crea una "Galois Lattice" definida por la correspondencia entre las descripciones de grafos (subestructuras) y los grafos descritos por ellas (instancias)

Conclusiones

- Análisis empírico
  - Dominio artificial
    - SubdueCL aprende exitosamente un dominio aún con la introducción de ruido
  - Comparación con sistemas ILP
    - SubdueCL competitivo con ILP's
      - Mejora u obtiene la misma precisión que FOIL y Progol
        - Excepto con atributos numéricos

Trabajo Actual y Futuro

- Teoría
  - Encontrar una expresión más formal y precisa para el tamaño del espacio de búsqueda
  - Utilizar grafos núcleo como un mecanismo de podado
  - Utilizar subclases de grafos para las que el subgrafo-isomorfo corra en tiempo polinomial “flexible”

Trabajo Actual y Futuro

- Mejoras a SubdueCL
  - Expresar rangos de valores (Tesis de Oscar Romero)
  - Expresar que una etiqueta de un vértice es la misma que la etiqueta de otro vértice (=, >, < para etiquetas numéricas)
  - Encontrar una representación capaz de describir reglas recursivas "gramáticas de grafos"
    - Istvan Jonyer
  - Mejorar la precisión con grafos conceptuales
  - Probar algoritmos de subgrafo-isomorfo sin generación de candidatos
    - SI-COBRA, tesis de Ivan Olmos