Natural Language: Relational Learning using Propositional Algorithms

Dan Roth
University of Illinois, Urbana-Champaign
danr@cs.uiuc.edu
http://L2R.cs.uiuc.edu/~danr

An Owed to the Spelling Checker

I have a spelling checker, it came with my PC
It plane lee marks four my revue
Miss steaks aye can knot sea.
Eye ran this poem threw it, your sure reel glad two no.
Its vary polished in it's weigh
My checker tolled me sew.
A checker is a bless sing, it freeze yew lodes of thyme.
It helps me right awl stiles two reed
And aides me when aye rime.
Each frays come posed up on my screen
Eye trussed to bee a joule...

Ambiguity Resolution

Illinois' bored of education board

...Nissan Car and truck plant is ...
...divide life into plant and animal kingdom

(This Art) (can N) (will MD) (rust V) V,N,N

The dog bit the kid. He was taken to a veterinarian
a hospital

More NLP Tasks

- Prepositional Phrase Attachment

  buy shirt with sleeves, buy shirt with a credit card

- Word Prediction

  She ___ the ball on the floor (wrote, dropped;…)

- Shallow Parsing

  [$_{NP}$ He] [$_{VP}$ reckons] [$_{NP}$ the current account deficit] [$_{VP}$ will narrow] [$_{PP}$ to] [$_{NP}$ only # 1.8 billion] [$_{PP}$ in] [$_{NP}$ September]

- Name Entity/ Categorization

  Tiger was in Washington for the GPA Tour

- Information Extraction Tasks

  afternoon, Dr. Ab C will talk in Ms. De. F class..

The Game

$$\mathbf{S} = \{\mathbf{s} = < (\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_k), \mathbf{p}(\mathbf{s}) > \} \longrightarrow \mathbf{p}(\mathbf{s}) = \mathbf{f}(\Phi_1, \Phi_2, \ldots, \Phi_n)$$

$\Phi_1, \Phi_2, \ldots, \Phi_n$ are "formulas" over the sentence

- Would like to learn <u>many definitions</u>

$$\mathbf{p}_1(\mathbf{s}), \mathbf{p}_2(\mathbf{s}), \ldots, \mathbf{p}_i(\mathbf{s}), \ldots$$

- Some might be defined in terms of others
- Chaining and inference with these are necessary for natural language understanding
  (Punyakanok, Roth, NIPS 2000)
- Here: learning a <u>single definition</u>.

Learning Concepts (Definitions)

- Define/Identify some properties of the given input

  The theory presented claims that the algorithm runs...

  Subject-Verb Phrase

- Definitions are complex in terms of raw data
- Might involve relational/quantified expressions
- Structural information is crucial
- ILP ?? Algorithmic issues – theoretical and practical

$\forall$ x $\exists$ y, z bef(y,x) $\wedge$ bef(x,z) $\wedge$ ppos(y,verb) $\wedge$ ppos(z,det) $\rightarrow$pos(x,verb)

- Typically a lot more complex (e.g., lexical items)
- Representation is very large; Learning is hard

Plan of the Talk

- Learning approach

  Generalizes well in the presence of a large # of features.

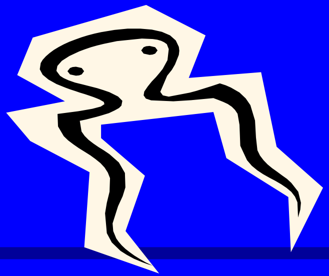
  along with

- A paradigm for relational intermediate representations (features)

  A language for Relation Generation functions

- Examples
- Final Thoughts

The Game



$$S = \{ s =< (x_1, x_2, \ldots x_k), p(s) > \} \longrightarrow p(s) = f(\Phi_1, \Phi_2, \ldots, \Phi_n)$$

S= I don't know {whether, weather} to laugh or cry

- Learn to make a decision in terms of:
  - word/pos tags around target word

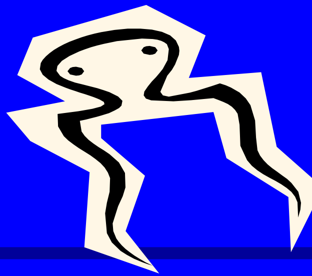
    don't within +/-3 know within +/-3

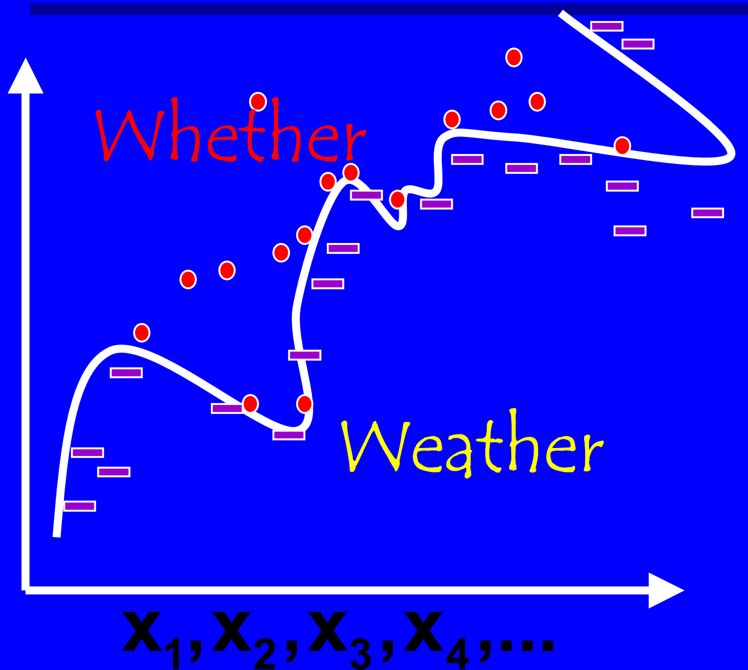
    to within +/- 3 laugh within +/-3
  - Size 2 conjunctions of word/pos tags

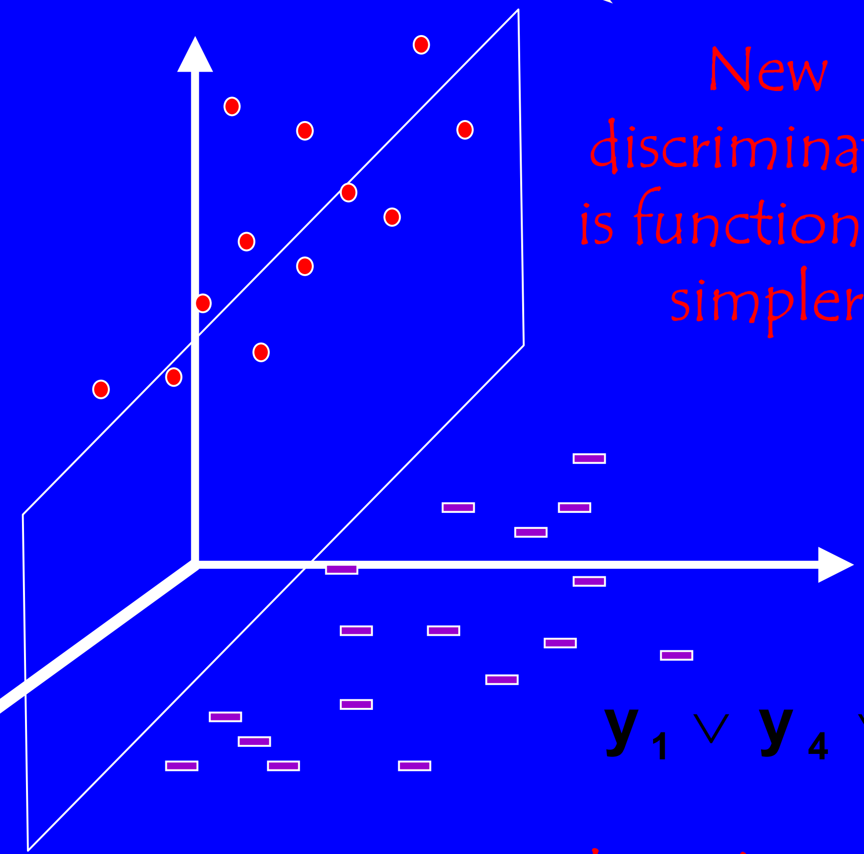
    words: know__; know__to; ___to laugh

    pos+words: Verb__; Verb__to; ____to Verb

Scenario







New discriminator is functionally simpler

$x_1 x_2 \bar{x}_3 \vee \bar{x}_1 x_4 \bar{x}_3 \vee x_3 \bar{x}_2 x_5$

$y_1 \vee y_4 \vee y_5$

Input Transformation

Learning

$\{x_1 x_2 \bar{x}_3, x_1 x_2 x_3, x_1 \bar{x}_2 \bar{x}_3, x_1 x_2 \bar{x}_4, \bar{x}_1 x_4 \bar{x}_3, x_3 \bar{x}_2 x_5, \ldots\}$

Intermediate Representations

- Features are indicator functions $\chi : \mathbf{X} \rightarrow \{0,1\}$
  - ❖ Define subsets of the instance space

$$\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = ((\mathbf{w}_1, \mathbf{t}_1), (\mathbf{w}_2, \mathbf{t}_2), \ldots, (\mathbf{w}_n, \mathbf{t}_n))$$

χ_1 : the condition $\exists \mathbf{i}(\mathbf{w}_i = \mathbf{good}, \mathbf{w}_{i+1} = \mathbf{talk})$

is active ($\chi_1(\mathbf{x}) = 1$) in x= "is this a good talk"

χ_2 : the condition $\exists \mathbf{i}(\mathbf{w}_i = \mathbf{talk}, \mathbf{t}_i = \mathbf{verb})$

is active ($\chi_2(\mathbf{x}) = 1$) in x= "It's good to talk to you"

Intermediate Representations

- Features are indicator functions $\chi : \mathbf{X} \rightarrow \{0,1\}$
  - Define subsets of the instance space

The collection Z of features maps the instance space into a feature space:

$$\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \rightarrow (\chi_1, \chi_2, \ldots \chi_{|Z|}) \in \{0,1\}^{|Z|\infty}$$

Learning is in terms of the intermediate representations

$$\mathbf{f}(\chi_1, \chi_2, \ldots \chi_{|Z|}) : \{0,1\}^{|Z|\infty} \rightarrow \{0,1\}$$

Old; in ILP: Propositionalization Lavrac,Dzerovsky(91);Kramer(01)

Decoupling of input transformation and Learning

Practical Approaches

- Most methods blow up original feature space

$$X(x_1, x_2, x_3, \ldots x_k) \rightarrow \mathbf{Z}(\chi_1(\mathbf{x}), \chi_2(\mathbf{x}), \chi_3(\mathbf{x}) \ldots \chi_n(\mathbf{x})) \qquad \mathbf{n >> k}$$

- And make predictions using a linear representation over the new feature space

$$\arg\max_j \sum_{\mathbf{i}} \mathbf{c}_{\mathbf{i}}^{\mathbf{j}} \chi_{\mathbf{i}}(\mathbf{x})$$

Note: Methods do not have to <u>actually</u> do that; But: they produce same decision as a hypothesis that does that. (Roth 98; 99,00)

Practical Approaches

- Most methods blow up original feature space

$$\boldsymbol{X}(\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_3,\ldots\boldsymbol{x}_k) \rightarrow \boldsymbol{Z}(\chi_1(\mathbf{x}),\chi_2(\mathbf{x}),\chi_3(\mathbf{x})\ldots\chi_n(\mathbf{x})) \qquad \mathbf{n >> k}$$

- And make predictions using a linear representation over the new feature space

$$\arg\max_j \sum_{\mathbf{i}} \mathbf{c}_{\mathbf{i}}^{\mathbf{j}} \chi_{\mathbf{i}}(\mathbf{x})$$

- Probabilistic Methods
- Rule based methods

  (TBL; decision lists; exponentially decreasing weights)
- Linear representation

  (SNoW;Perceptron; SVM;Boosting)
- Memory Based Methods

  (subset features)

Practical Approaches

- Most methods blow up original feature space

$$X(x_1, x_2, x_3, \ldots x_k) \rightarrow \mathbf{Z}(\chi_1(\mathbf{x}), \chi_2(\mathbf{x}), \chi_3(\mathbf{x}) \ldots \chi_n(\mathbf{x})) \qquad \mathbf{n >> k}$$

- And make predictions using a linear representation over the new feature space

$$\arg\max_{j} \sum_{\mathbf{i}} \mathbf{c}_{\mathbf{i}}^{\mathbf{j}} \chi_{\mathbf{i}}(\mathbf{x})$$

Q 1: How are weights determined?

Q 2: How is the new feature-space determined?

Relations? Implications?

Algorithmic Approaches

- Focus: Two families of algorithms
  (will discuss the on-line representative)

- Additive update algorithms: Perceptron

  SVM (not on-line, but a close relative of Perceptron)

- Multiplicative update algorithms: Winnow

  SNoW
  Close relatives: Boosting; Max Entropy

Algorithm Descriptions

Examples : $x \in \{0,1\}^n$; **Hypothesis** : $w \in R^n$

Prediction is 1 iff $w \bullet x \geq \theta$

- Additive weight update algorithm

  (Perceptron, Rosenblatt, 1958. Variations exist)

If Class $= 1$ but $w \bullet x \leq \theta$, $w_i \leftarrow w_i + 1$ (if $x_i = 1$) (promotion)

If Class $= 0$ but $w \bullet x \geq \theta$, $w_i \leftarrow w_i - 1$ (if $x_i = 1$) (demotion)

- Multiplicative weight update algorithm

  (Winnow, Littlestone, 1988. Variations exist)

Relative Entropy Based

If Class $= 1$ but $w \bullet x \leq \theta$, $w_i \leftarrow 2w_i$ (if $x_i = 1$) (promotion)

If Class $= 0$ but $w \bullet x \geq \theta$, $w_i \leftarrow w_i/2$ (if $x_i = 1$) (demotion)

How to Compare?

- Generalization

  (since the representation is the same)

  How many examples are needed to get to a given level of accuracy?

- Efficiency

  How long does it take to evaluate a hypothesis?

- Robustness; Adaptation to a new domain, ….

Learning in NLP: Characteristics

- The number of potential features is very large
- The instance space is sparse
- Decisions depend on a small set of features (sparse)

- Want to learn from a number of examples that is small relative to the dimensionality

Generalization

- Dominated by the sparseness of the function space

  Most features are irrelevant

  Advantage multiplicative: # of examples required depends mostly on # of relevant features

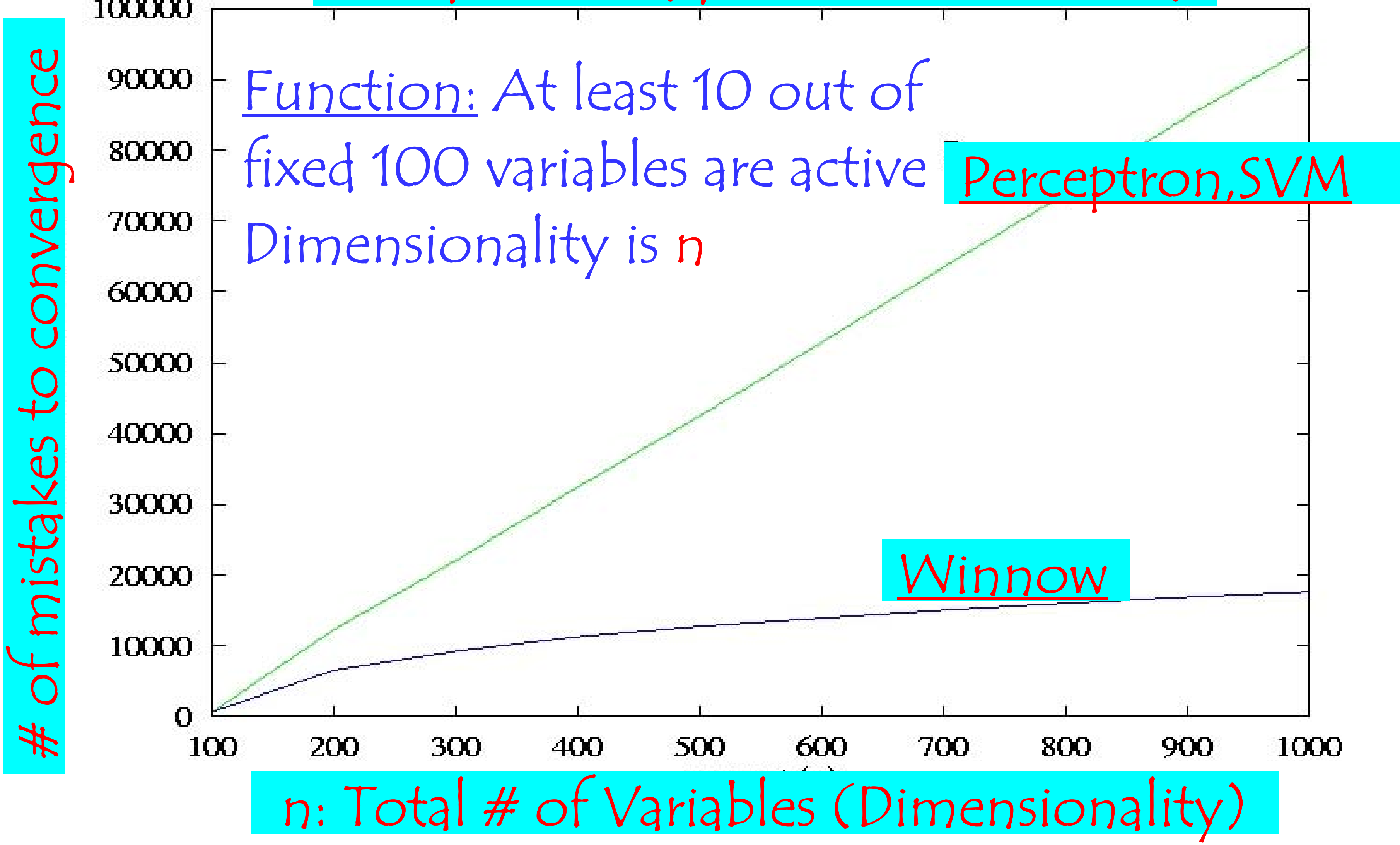
  (Generalization bounds depend on ||w||;)

- Lesser issue: Sparseness of features space:

  Advantage additive. Generalization depend on ||x||

  (Kivinen/Warmuth 95)

Mistakes bounds for 10 of 100 of n



Efficiency

- Dominated by the size of the feature space
- Most features are functions (e.g., n-grams) of raw attributes

$$X(x_1, x_2, x_3, \ldots x_k) \rightarrow \mathbf{Z}(\chi_1(\mathbf{x}), \chi_2(\mathbf{x}), \chi_3(\mathbf{x}) \ldots \chi_n(\mathbf{x})) \qquad \mathbf{n >> k}$$

- Additive algorithms allow the use of Kernels

  No need to explicitly generate the complex features

$$\sum_{\mathbf{i}} \mathbf{c_i\, K(x, x_i)}$$

- Irrelevant here due to blow-up methods
- But, wait for discussion

SNoW Learning Architecture

- The most successful approach tried on several NLP problems
- A learning architecture tailored for high dimensional problems
- Multi Class Learner; Robust confidence in prediction
- A network of linear representations
- Several update algorithms are available
- Most successful – a multiplicative update algorithm a variation of Winnow (Littlstone'88)

SNoW http://L2R.cs.uiuc.edu/~danr/snow.html

- Feature space: Infinite Attribute Space $\{0,1\}^\infty$
  - examples of variable size: only active features
  - determined in a data driven way
- Makes Possible:

Generation of many complex/relational types of features

Only a small fraction is actually represented

- Computationally efficient (on-line!)

Work Done

- Context Sensitive Text Correction

  (peace;piece) (among;between)
- Prepositional Phrase Attachment

  (car with..., buy with...)
- Part of Speech Tagging

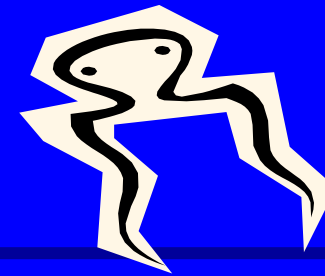
  (Verb, Noun, Adj,...)
- Shallow Parsing Tasks

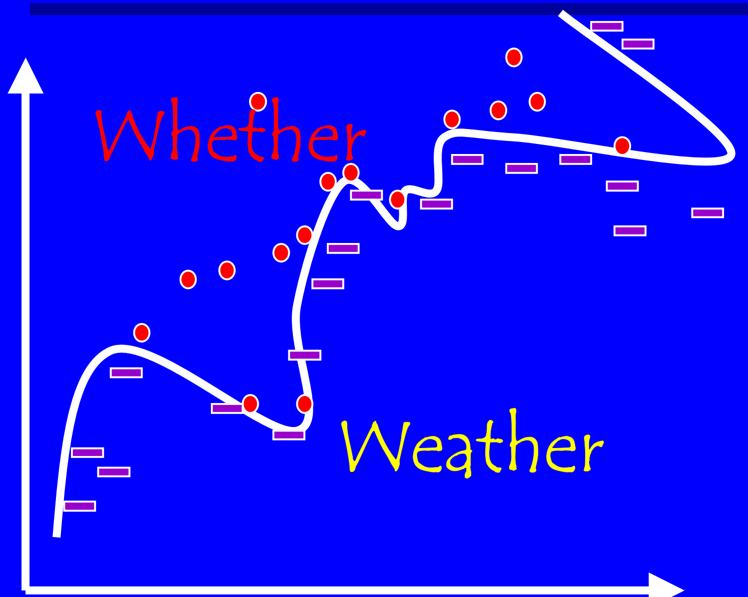
  (noun phrases; subject-verb)
- Information Extraction
- Comparable or Superior in performance and efficiency

Outline

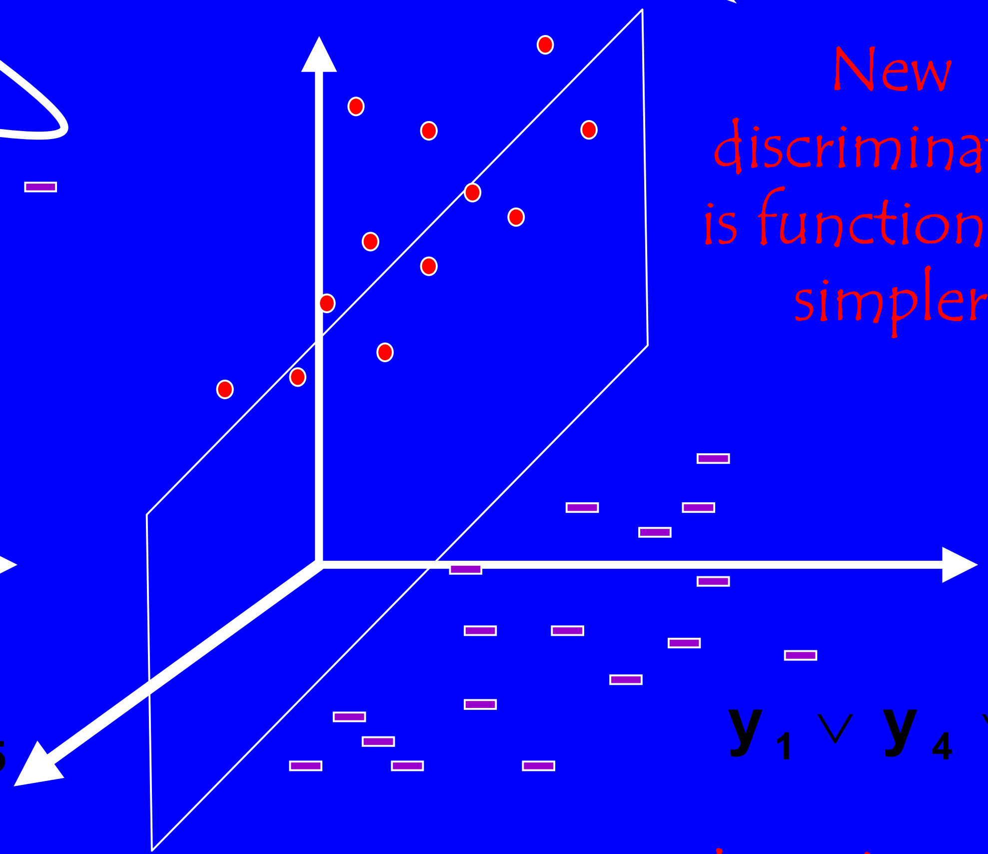
- Learning approach
  - Generalizes well in the presence of a large # of features.
- A paradigm for relational intermediate representations (features)
  - A language for Relation Generation functions
- Examples
- Final Thoughts

Scenario





$x_1, x_2, x_3, x_4, \ldots$



New discriminator is functionally simpler

$x_1x_2\bar{x}_3 \vee \bar{x}_1x_4\bar{x}_3 \vee x_3\bar{x}_2x_5$

$y_1 \vee y_4 \vee y_5$

Input Transformation

Learning

$\{x_1x_2\bar{x}_3, x_1x_2x_3, x_1\bar{x}_2\bar{x}_3, x_1x_2\bar{x}_4, \bar{x}_1x_4\bar{x}_3, x_3\bar{x}_2x_5, \ldots\}$

Feature Space

- Traditionally, only simple functions of the raw input were used as features

  Bi-grams/Tri-grams

  (conjunctions of consecutive tokens)

- Influence of probabilistic models (Markov etc.)

But

- Representing interesting concepts often requires
  - The use of relational expressions.
  - Better exploitation of the structure

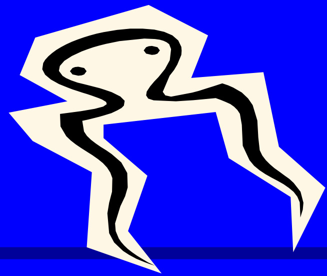
A Better Feature Space

- Feature efficient algorithms allow us to the extend the types of intermediate representations used.
- More potential features is not a problem

- Generate complex features that represent (also) relational (FOL) constructs

- Structure: Extend the generation of features beyond the linear structure of the sentence.

A Relational View



1. Instead of a rule representation

$$\mathbf{R} = [\forall \mathbf{x},(\exists \mathbf{y},\Phi_1(\mathbf{x},\mathbf{y}) \wedge \Phi_2(\mathbf{x},\mathbf{y})) \rightarrow \mathbf{f(x)}]$$

We use generalized rules:

$$\cancel{\mathbf{R} = [\forall \mathbf{x},(\exists \mathbf{y},[\mathbf{w}_1\Phi_1(\mathbf{x},\mathbf{y}) + \mathbf{w}_2\Phi_2(\mathbf{x},\mathbf{y})] \geq 1) \rightarrow \mathbf{f(x)}]}$$

- More expressive; Easier to learn

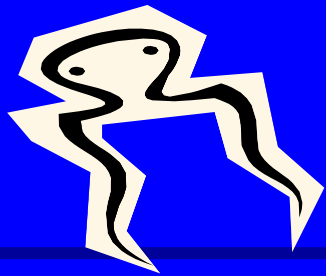
Single predicate in scope

2. Restrict to Quantified Propositions

$$\mathbf{R'} = [\forall \mathbf{x},[\mathbf{w}_1 \cdot \underline{(\exists \mathbf{y}_1,\mathbf{c}_1(\mathbf{x},\mathbf{y}_1)}) + \mathbf{w}_2 \cdot (\underline{\exists \mathbf{y}_2,\mathbf{c}_2(\mathbf{x},\mathbf{y}_2)})) > 1] \rightarrow \mathbf{f(x)}]$$

- Allows use of Propositional Algorithms; but more predicates are required to maintain expressivity

Expressivity



$$\mathbf{R} = [\forall x, (\exists y, c_1(x,y) \wedge c_2(x,y)) \rightarrow f(x)]$$

Restricting to using quantified proposition

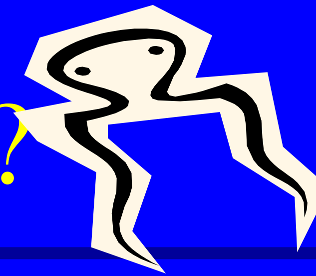
$$\mathbf{R'} = [\forall x, (\ (\exists y_1, c_1(x,y_1)) \wedge (\exists y_2, c_2(x,y_2))\ ) \rightarrow f(x)] \qquad \neq \mathbf{R}$$

can be overcome using new predicates (features)

$$\mathbf{R''} = [\forall x, y, (\ c_1(x,y) \wedge c_2(x,y)\ ) \rightarrow f'(x,y)]$$

$$\mathbf{R} = [\forall x, (\exists y, f'(x,y)\ ) \rightarrow f(x)]$$

Why Quantified Propositions?



Allow different parts of the program's conditions to be evaluated separately from others.

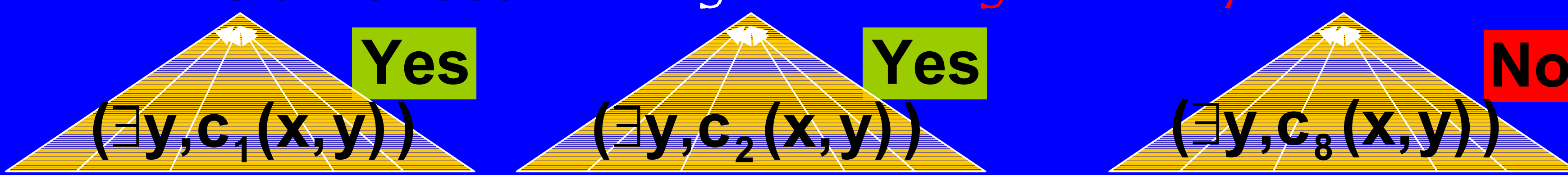
$$R' = [\forall x, ((\exists y_1, c_1(x,y_1)) \wedge (\exists y_2, c_2(x,y_2))) \rightarrow f(x)]$$

Given a sentence –

binding of x determines the example

Given a binding –

$(\exists y, c(x,y))$ is assigned a single binary value



(Sentence, x = this)

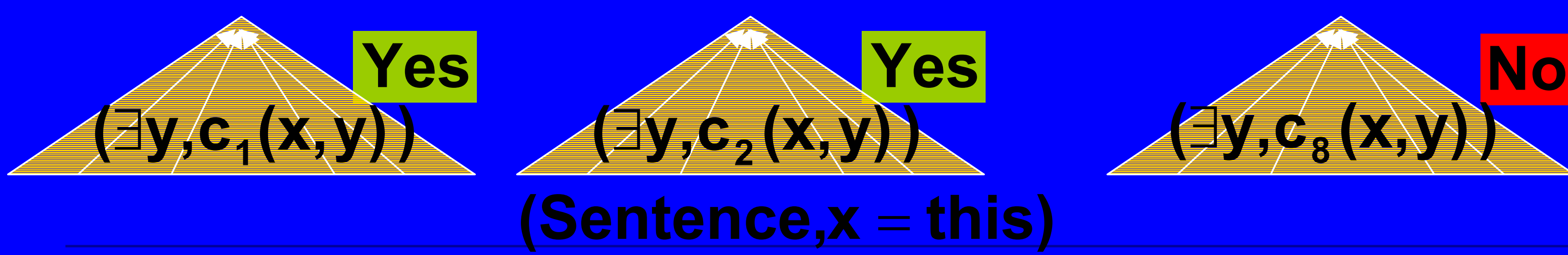
Why Quantified Propositions?

Allow different parts of the program's condition to be evaluated separately from others.

$$R' = [\forall x, ((\exists y_1, c_1(x,y_1)) \wedge (\exists y_2, c_2(x,y_2))) \rightarrow f(x)]$$

For each x: the sentence is mapped into a collection of binary features in the relational space

Important in inference, but even more so in Learning



Relational Features

- Features are indicator functions $\chi : \mathbf{X} \rightarrow \{0,1\}$

The collection Z of features maps the instance space into a feature space:

$$\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \rightarrow (\chi_1, \chi_2, \ldots \chi_{|Z|}) \in \{0,1\}^{\infty}$$

$$\chi \rightarrow \phi$$

X - instance space (e.g., all sentences)

A formula ϕ maps an instance $x \in X$ to its truth value

A relation: ϕ: $X \rightarrow \{0,1\}$ (ϕ is active/non-active in x)

A Knowledge Representation Language

- A restricted FOL language + A set of structures

- Domain
  - Typed elements and structures over these
- Formulae
  - Primitive Formulae

    Relational mapping from domain to propositions
  - Relation Generation Function

    General formulae are defined inductively and generated in a data-driven manner

Domain

- Domain $\mathcal{D}=(\mathcal{V},\mathcal{G})$
  - $\mathcal{V}$ – a collection of typed elements
  - $\mathcal{G}$ – a set of partial orders (acyclic graphs) over $\mathcal{V}$
- $\mathcal{V}$ induces type on predicates
- attributes: $\mathcal{A} \subseteq \mathcal{V}$ objects: $\mathcal{O} \subseteq \mathcal{V}$
- p(o,a) – properties $q(o_1,o_2)$ – defined in $g \in \mathcal{G}$

pos(w,noun) before(a,b)

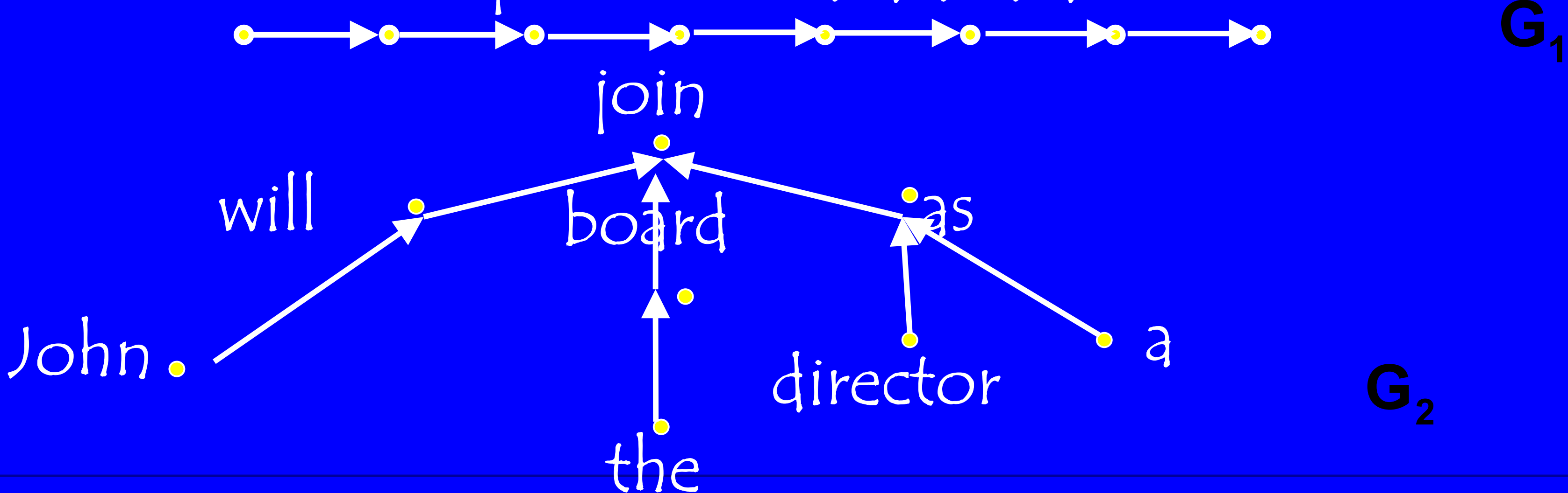
part_of(a,b)

Structured Domain

afternoon, → Dr. → Ab → C → …in → Ms. → De. F class..

[$_{NP}$ Which type] [$_{PP}$ of] [$_{NP}$ submarine] [$_{VP}$ was bought]
[$_{ADVP}$recently] [$_{PP}$ by] [$_{NP}$ South Korea] (. ?)

S = John will join the board as a director



Primitive Formulae

- Atomic formula:

  $F=p(t_1, \ldots, t_k)$, for k-ary predicate p.
- Primitive formula

  $(\forall zF)$, $(\exists zF)$

  $(\neg F)$, $(F \wedge G)$, $(F \vee G)$.

A unique predicate in the scope of each variable.

\+ A formula F maps an instance $x \in X$ to its truth value

A relation: $F: X \rightarrow \{0,1\}$ (F is active/non-active in x)

\- Not expressive enough

Relation Generation Functions

X - instance space (e.g., all sentences)
A formula F maps an instance $x \in X$ to its truth value
A relation: F: $X \rightarrow \{0,1\}$ (F is active/non-active in x)

A Relation Generation Function (RGF) is a mapping

$$G: X \rightarrow 2^{\mathcal{F}}$$

that maps $x \in X$ to a set of relations in $\mathcal{F}$ with $F(x)=1$.

$x \rightarrow$ set of all formulae in $\mathcal{F}$ that are active in x

Relation Generation Function (2)

- Sensor
  - A sensor is a basic relation generation function that maps an instance $\boldsymbol{x}$ into a set of atomic formulas.
- Relational Calculus
  - Allows to inductively compose RGFs, along domain structures
- Binding (focus); Existential; Condition; ….

Sensors

- A sensor is a relation generation function that maps an instance *x* into a set of atomic formulas.
- When evaluated on an instance *x*, a sensor s outputs all atomic formulas in its range which are active.
- Sensors understand the domain (background knowledge)
  - They can be read directly from the raw data ("word")
  - Encode knowledge ("is-a" ; wordnet)
  - Be previously learner functions ("pos tag" ; "subject")

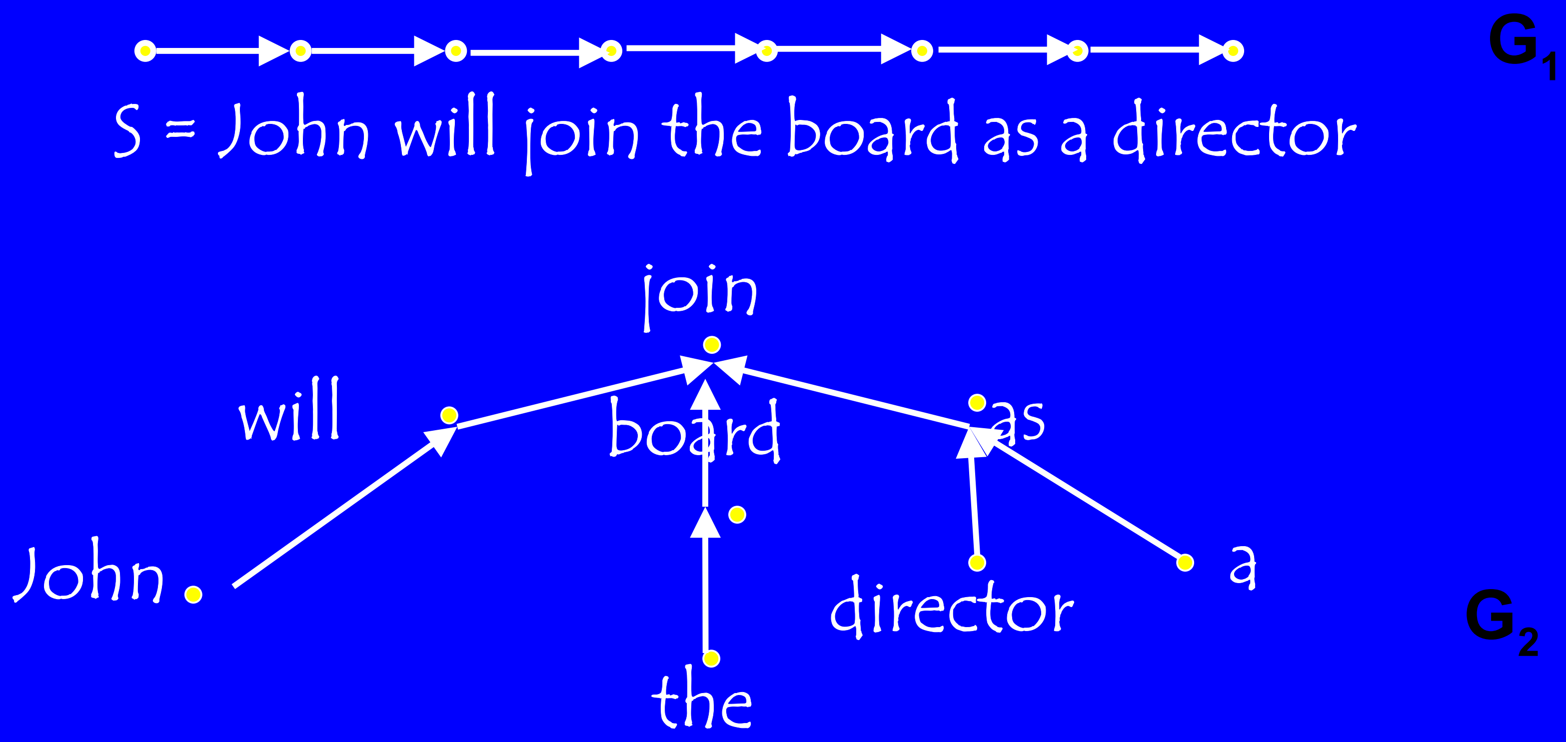
Relational Calculus

- Allows to inductively compose RGFs.
  The collection of formulae is defined inductively.

- Simple Connectives
  - word&tag, number | prefix[X], …

- Structural Calculus
  Allow formulae like $\exists y, p(x,y) \wedge q(y,z)$
  restricted to be local relative to the relational structure of the domain

Example: Structure



Example (2)

S = John will <u>join</u> the board as a director

colloc(G_1)(word, tag)

word(will)-tag(Verb), word(join)-tag(Det),.......

scolloc(G_1)(word, word,word

word(John)-word(will)-word(director),

word(John)-word(join)-word(as),.........

Example (3)

S = John will join the board as a director

- collocations relative to G_2 ; sensors: subject, Verb

  A= colloc(G_2)(subject, verb)

  B= colloc(G_2)(subject, aux, verb)

  (# of intermediate words does not matter)

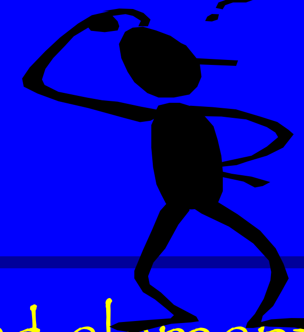
  C= A | B

- Similar feature-based representations for:

  S = John {will;may} join the board as a director

- Achieved the abstraction required for learning.

What is going on?



- Input (sentences) represented as structured elements
- A small number (5) of RGFs is used to encode kinds of formulae of potential interest
- Active formulae (relations, features) are generated in a data drive way to re-represent input instances

$$\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \rightarrow (\chi_1, \chi_2, \ldots \chi_{|Z|}) \in \{0,1\}^\infty$$

- Most of the generated formulae are junk.
- Some are very important (e.g., agreements; KR2000)
- Some, in between, but still important for learning.

Work

- FEX: (KR 2000) Software available

Many Applications:

- Disambiguation tasks (Even-Zohar,Roth NAACL'00)
- Information Extraction (Roth, Yih IJCAI'2001)

  Identifying functional phrases
- Family relationships (Cumby, 2001)
- Gene Identification; Visual Recognition

More accurate; orders of magnitude more efficient

Summary (I) The Problem

Subject(x)=F(after(x,verb),before(x,determiner),noun(x)..)

- Problems in NLP are relational, but representations require many lexicalized ground items, not only predicates with variables.

  grandfather(x,z):-father(x,y)parent(y,z)

- ILP offers unlimited induction over structures but is strongly intractable; successful heuristics do not scale up well enough

Summary (II) Our Solution

- A paradigm that allows the use of general purpose propositional algorithms to learn relational representations.
  Conceptually: propsitionalization (Kramer 2001)
- Key: A Knowledge Representation language for representing and evaluating relational structures.
- Generate features that represent (also) relational (FOL) constructs and map them to propositions
- Exploits structure in the domain: RGFs restricted to be local relative to the domain's relational structure.
- Enabled by the use of feature efficient learning algorithms

Conclusions

Relational Representations that Facilitate Learning

Learning approach

- Generalizes well in the presence of a large # of features.
- Handles variable size examples
- Learns "generalized" rules (linear threshold functions)

A paradigm for relational intermediate representations

- A language for Relation Generation functions

Experimental Evidence

Final Thoughts

- The paradigm suggests that we need to think only in terms of "kinds" of features (RGFs)
  - What are good RGFs? Principles?
- What is lost?
  - Algorithmic exploitation of Lattice of features
- Recent progress: RGFs can be viewed as kernels
- But [Khardon, Roth, Servedio, NIPS 2001, to appear]
  - The kernel idea cannot be used by multiplicative algorithms.
  - Additive algorithms using Kernels do not gain in generalization - still depends on the blown up dimensionality.
- Hope?

An Owed to the Spelling Checker

I have a spelling checker, it came with my PC
It plane lee marks four my revue
Miss steaks aye can knot sea.
Eye ran this poem threw it, your sure reel glad two no.
Its vary polished in it's weigh
My checker tolled me sew.
A checker is a bless sing, it freeze yew lodes of thyme.
It helps me right awl stiles two reed
And aides me when aye rime.
Each frays come posed up on my screen
Eye trussed to bee a joule...