Algorithms for NLP

Classification I

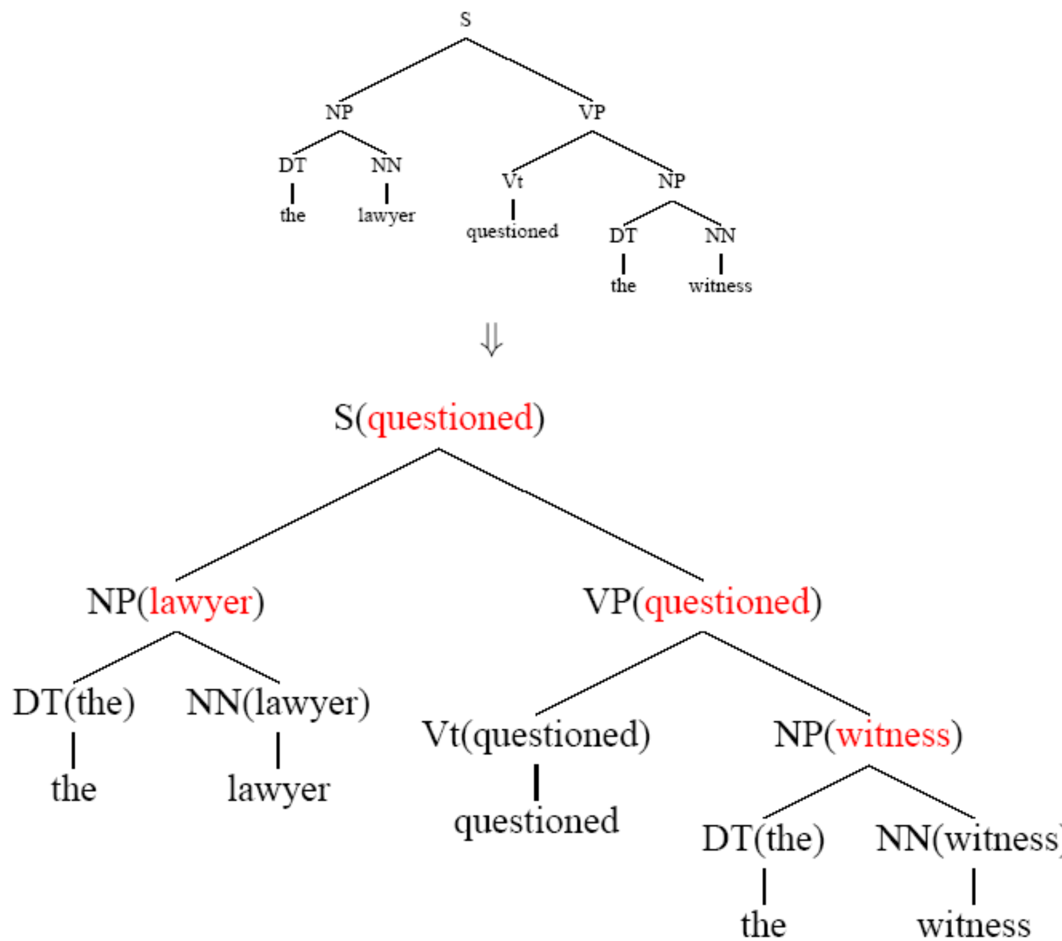
Taylor Berg-Kirkpatrick – CMU

Slides: Dan Klein – UC Berkeley

Efficient Parsing for Lexical Grammars

Lexicalized Trees

- Add “head words” to each phrasal node
  - Syntactic vs. semantic heads
  - Headship not in (most) treebanks
  - Usually *use head rules*, e.g.:
    - NP:
      - Take leftmost NP
      - Take rightmost N*
      - Take rightmost JJ
      - Take right child
    - VP:
      - Take leftmost VB*
      - Take leftmost VP
      - Take left child



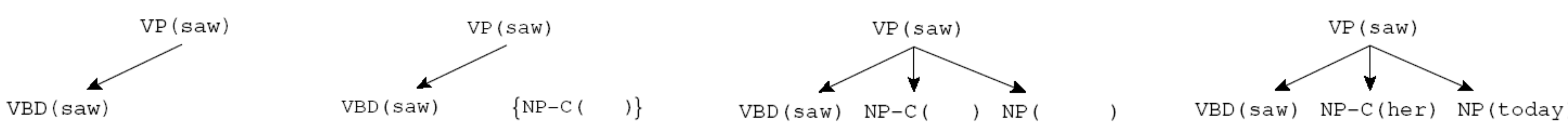
Lexicalized PCFGs?

- Problem: we now have to estimate probabilities like

`VP(saw) -> VBD(saw) NP-C(her) NP(today)`

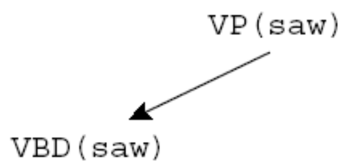
- Never going to get these atomically off of a treebank

- Solution: break up derivation into smaller steps

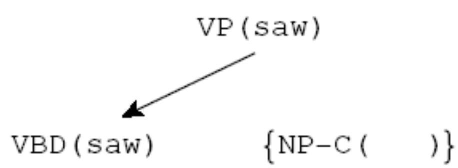


Lexical Derivation Steps

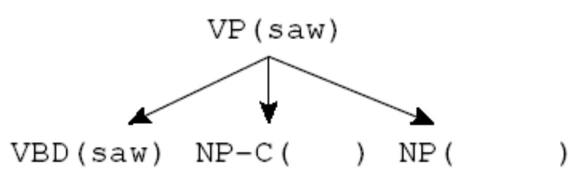
- A derivation of a local tree [Collins 99]



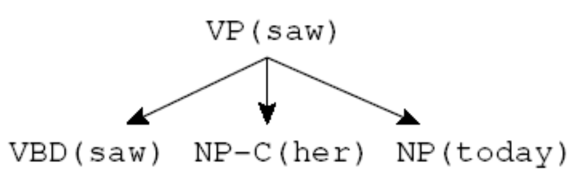
Choose a head tag and word



Choose a complement bag

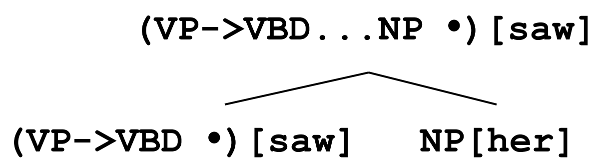


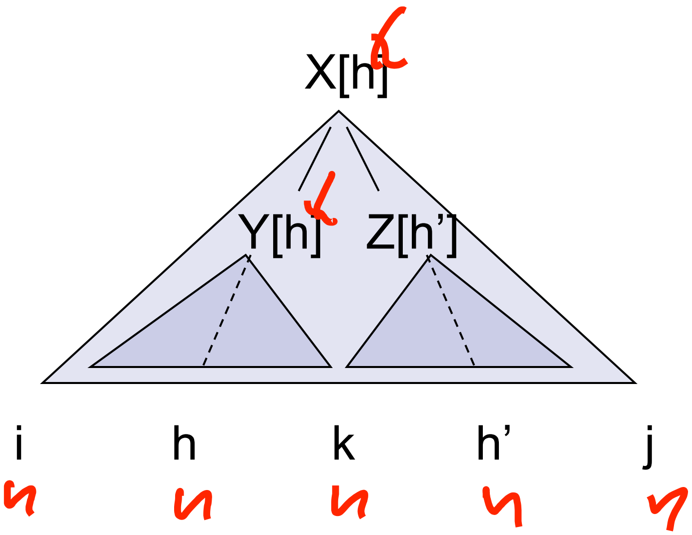
Generate children (~~incl. adjuncts~~)



Recursively derive children

Lexicalized CKY



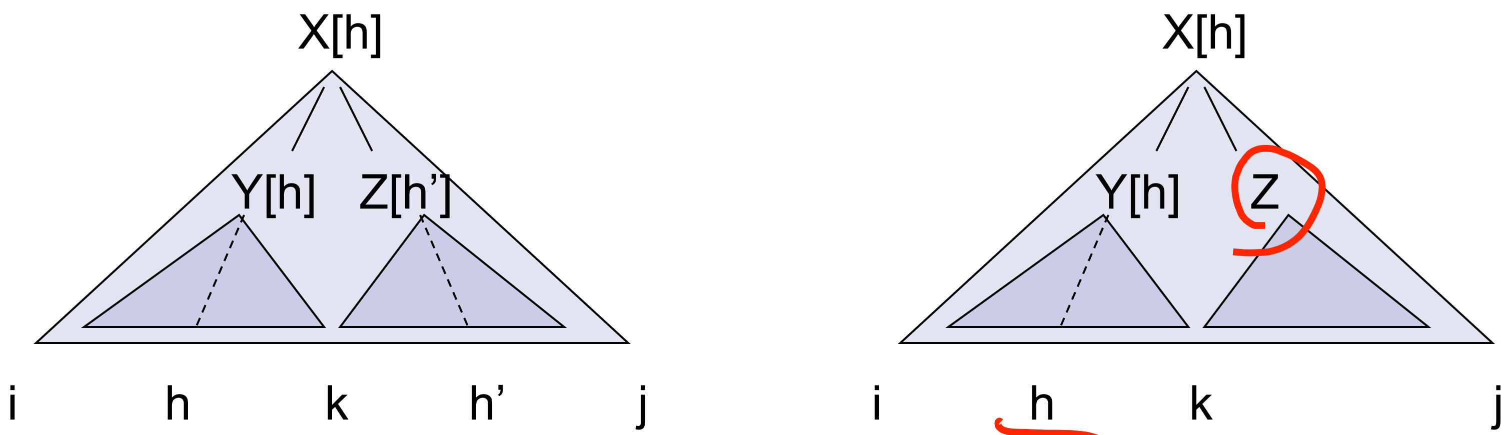



```
bestScore(X,i,j,h)
  if (j = i+1)
    return tagScore(X,s[i])
  else
    return
      max   max      score(X[h]->Y[h] Z[h']) *
      k,h',X->YZ     bestScore(Y,i,k,h) *
                     bestScore(Z,k,j,h')
            max      score(X[h]->Y[h'] Z[h]) *
      k,h',X->YZ     bestScore(Y,i,k,h') *
                     bestScore(Z,k,j,h)
```

$O(n^5)$

Quartic Parsing

- Turns out, you can do (a little) better [Eisner 99]

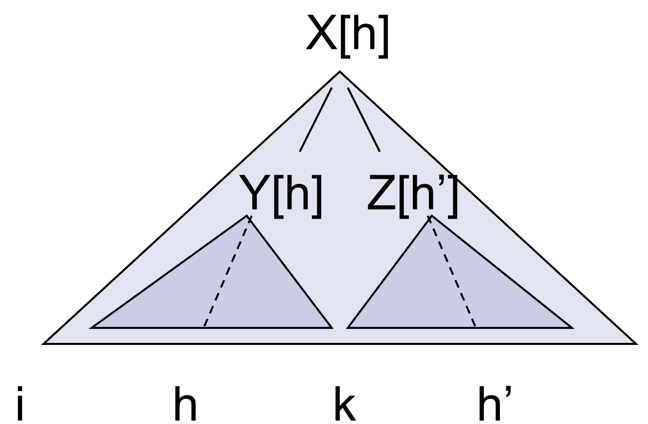


- Gives an $O(n^4)$ algorithm
- Still prohibitive in practice if not pruned

Pruning with Beams

$\{X : s_1, Y : s_2\}$

- The Collins parser prunes with per-cell beams [Collins 99]
  - Essentially, run the $O(n^5)$ CKY
  - Remember only a few hypotheses for each span <i,j>.
  - If we keep K hypotheses at each span, then we do at most $O(nK^2)$ work per span (why?)
  - Keeps things more or less cubic (and in practice is more like linear!) $O(n^3 K^2)$
- Also: certain spans are forbidden entirely on the basis of punctuation (crucial for speed)



Pruning with a PCFG

- The Charniak parser prunes using a two-pass, coarse-to-fine approach [Charniak 97+]
  - First, parse with the base grammar
  - For each X:[i,j] calculate P(X|i,j,s)
    - This isn't trivial, and there are clever speed ups
  - Second, do the full $O(n^5)$ CKY
    - Skip any X :[i,j] which had low (say, < 0.0001) posterior
  - Avoids almost all work in the second phase!

- Charniak et al 06: can use more passes
- Petrov et al 07: can use many more passes

Results

- Stanford Parser – 86.3 (unlex / struct annotation)
- Collins 99 – 88.6 F1 (lexical)
- Charniak and Johnson 05 – 89.7 / 91.3 F1 (lexical + rerank)
- McClosky et al 06 – 92.1 F1 (lexical + rerank + self-train)

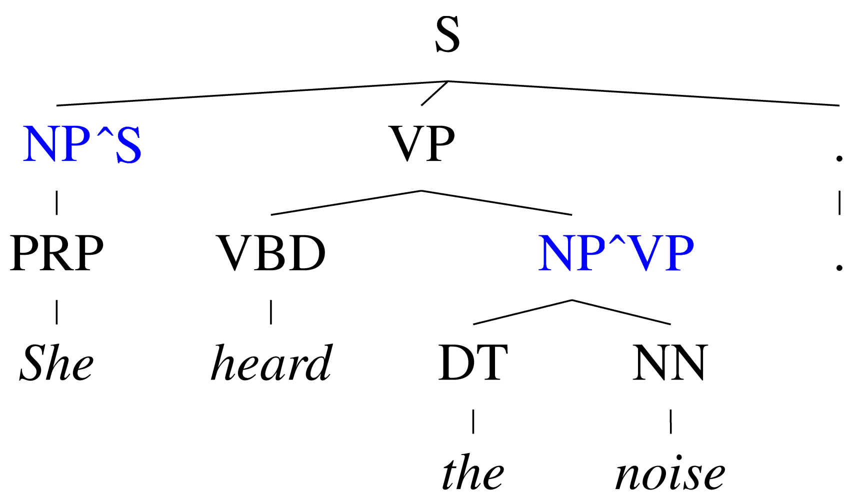
- Petrov et al 06 – 90.7 F1 (unlex / latent vars)
- Petrov et al 10 – 91.8 (unlex / latent vars + ensemble)

- Socher et al 13 – 90.4 (unlex + neural rerank)
- Vinyals et al 15 – 90.5 / 92.1 (neural sequence + self-train)
- Dyer et al 16 – 92.4 (neural shift-reduce)

…many more that are really cool (e.g. Hall and Klein 12,14)

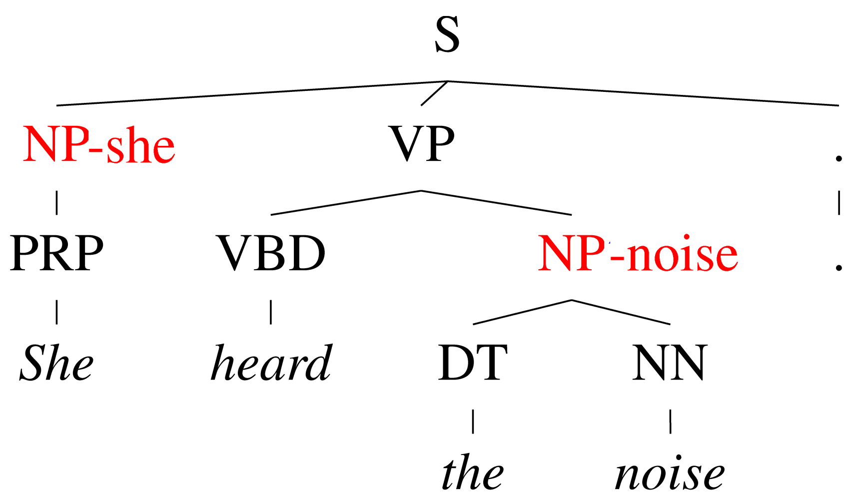
Latent Variable PCFGs

The Game of Designing a Grammar



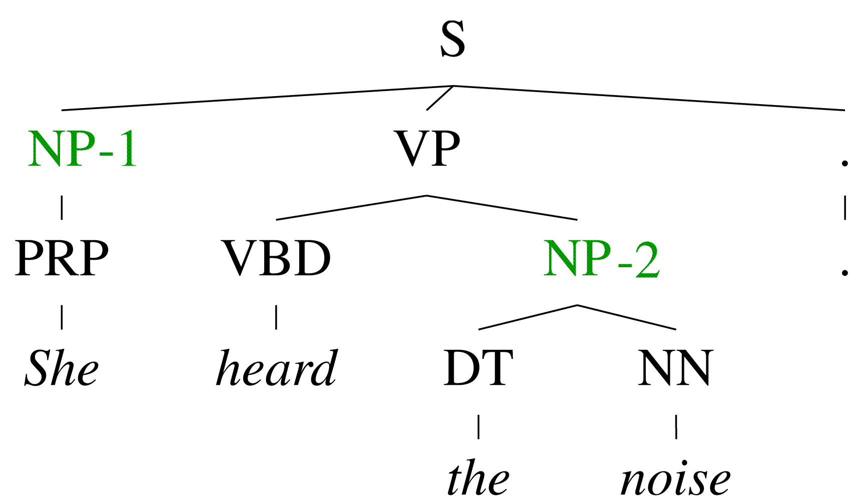
- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Parent annotation [Johnson ’98]

The Game of Designing a Grammar



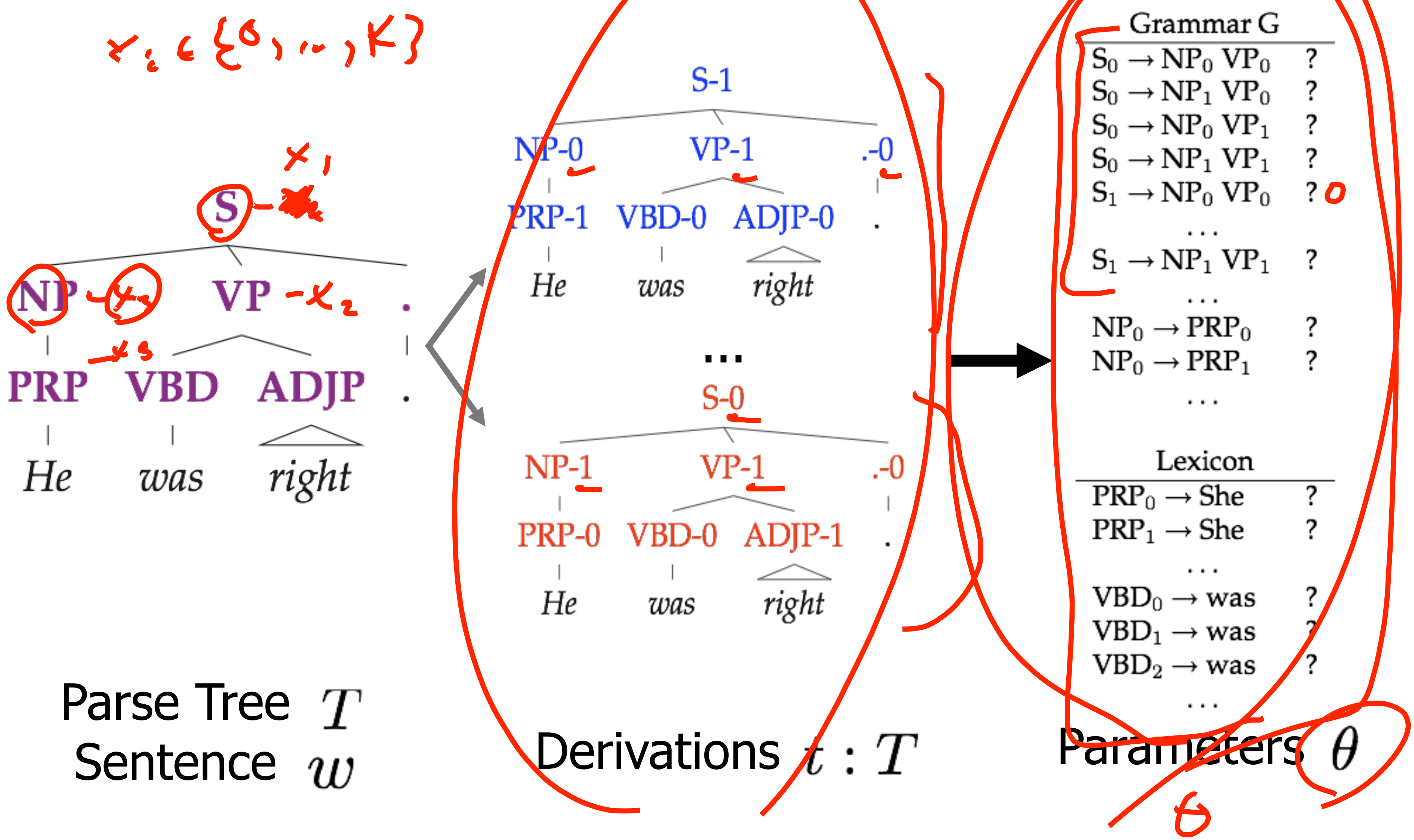
- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Parent annotation [Johnson ’98]
  - Head lexicalization [Collins ’99, Charniak ’00]

The Game of Designing a Grammar



- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Parent annotation [Johnson ’98]
  - Head lexicalization [Collins ’99, Charniak ’00]
  - Automatic clustering?

Latent Variable Grammars

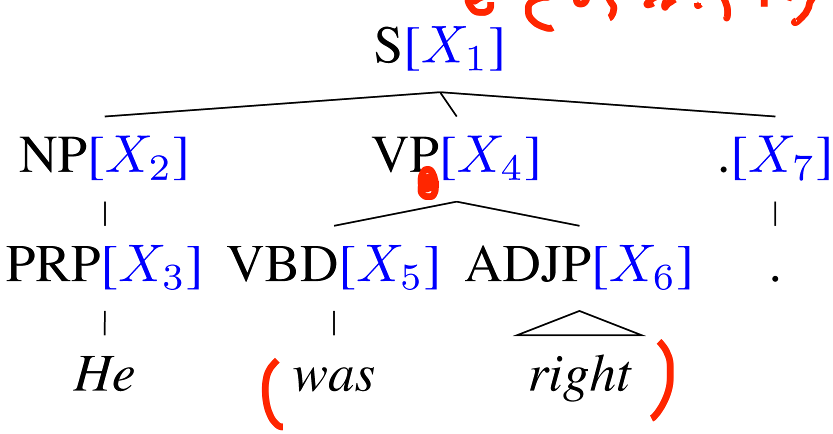


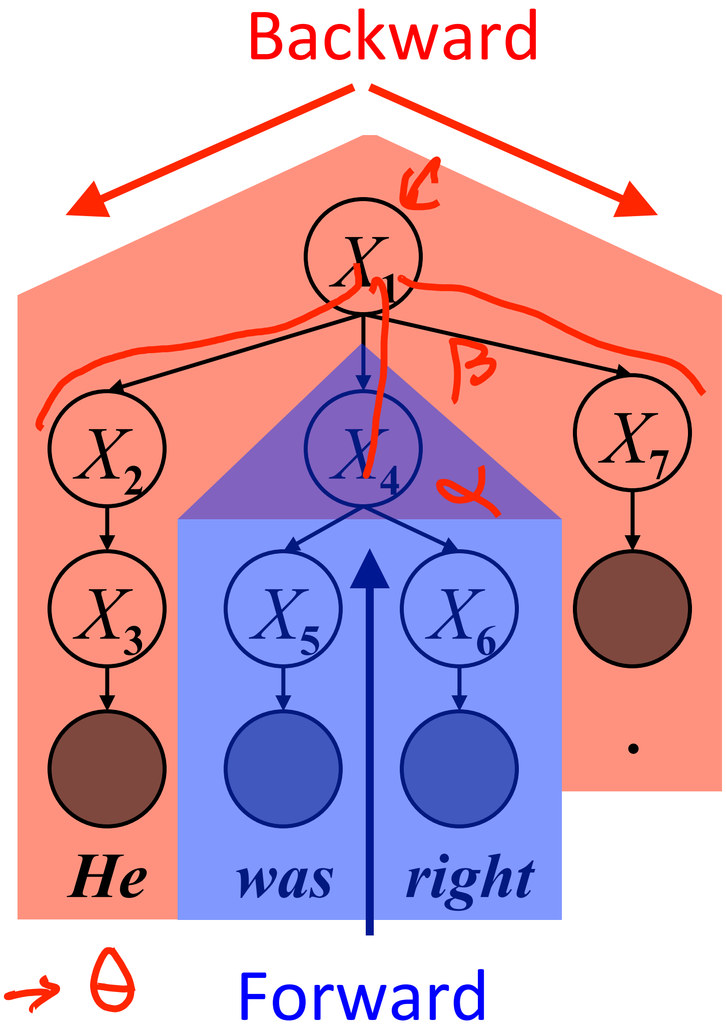
Learning Latent Annotations

EM algorithm:

- Brackets are known
- Base categories are known
- Only induce subcategories

$\in \{0, \dots, K\}$

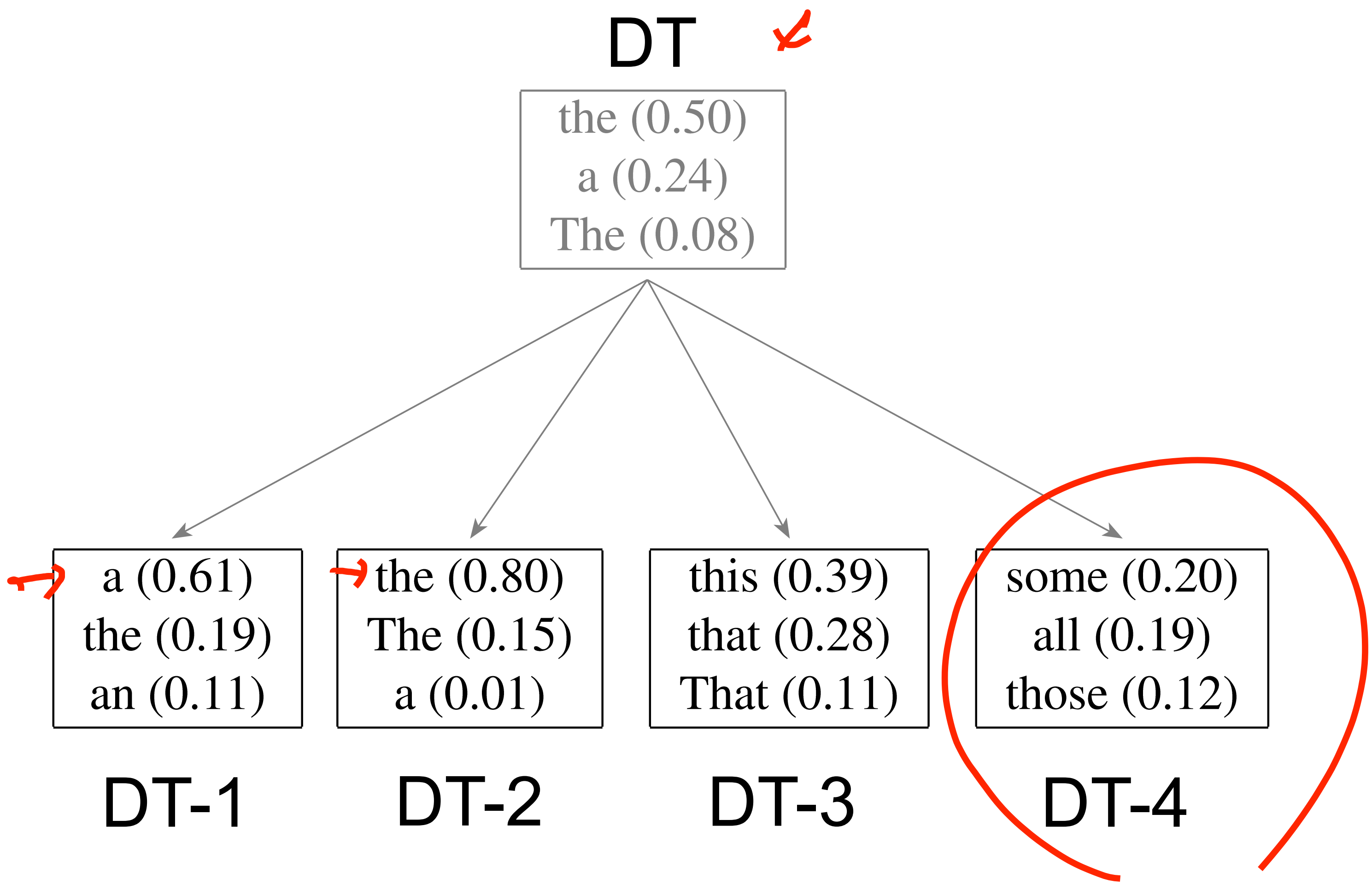




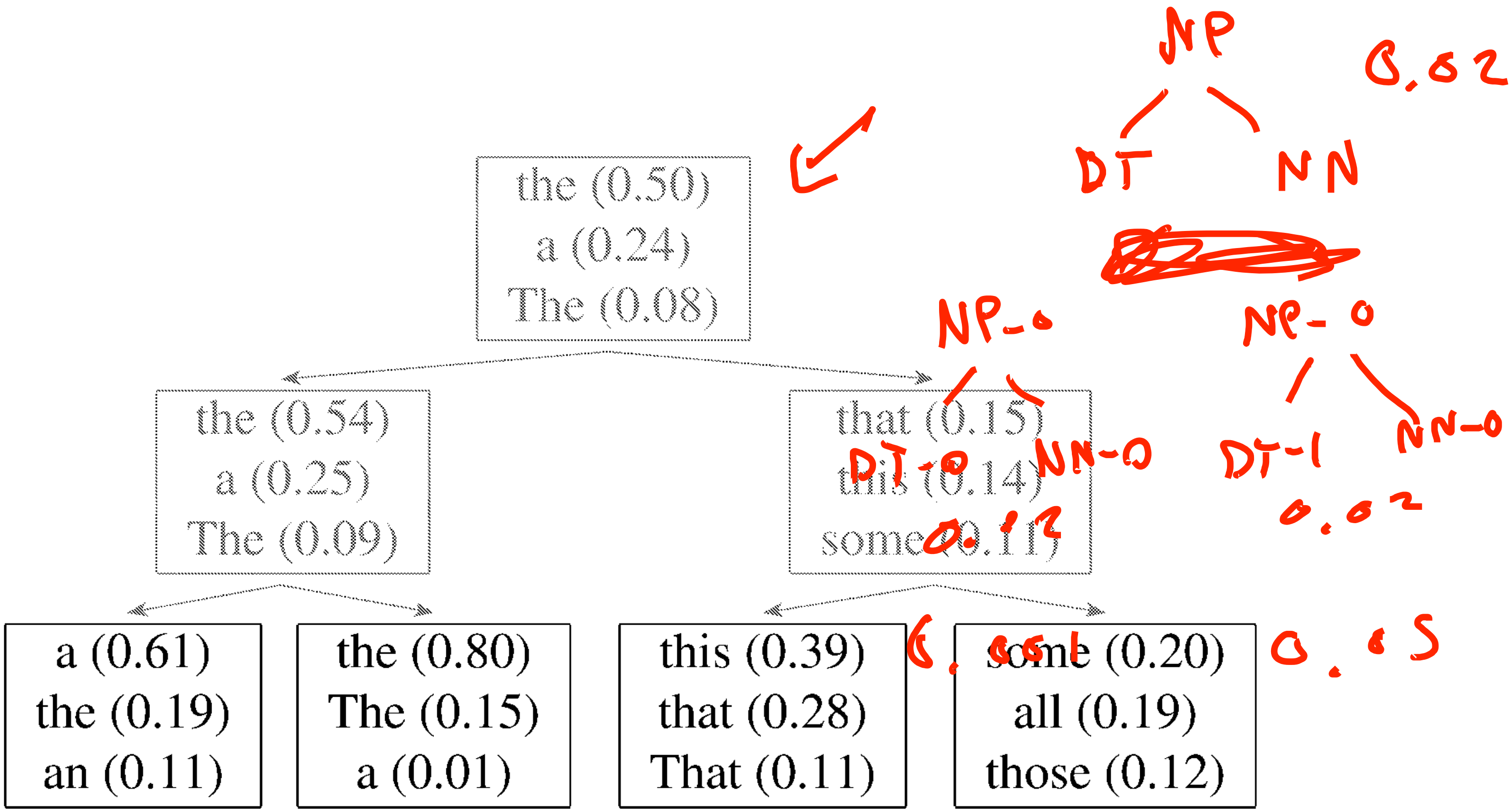
Just like Forward-Backward for HMMs.

$\theta \rightarrow E[X \mid T, w], \; E[x \mid T, w] \rightarrow \theta$

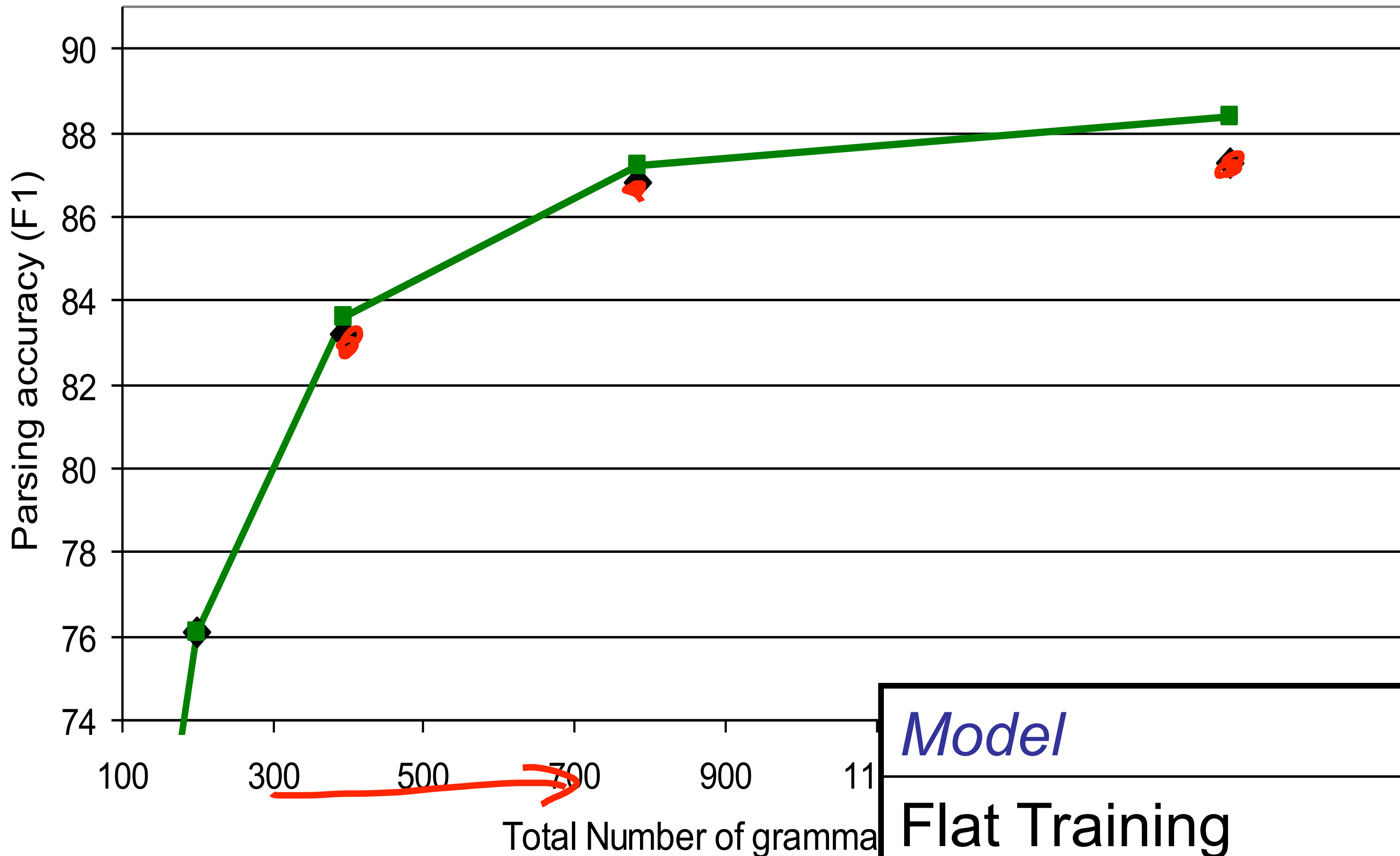
Refinement of the DT tag



Hierarchical refinement



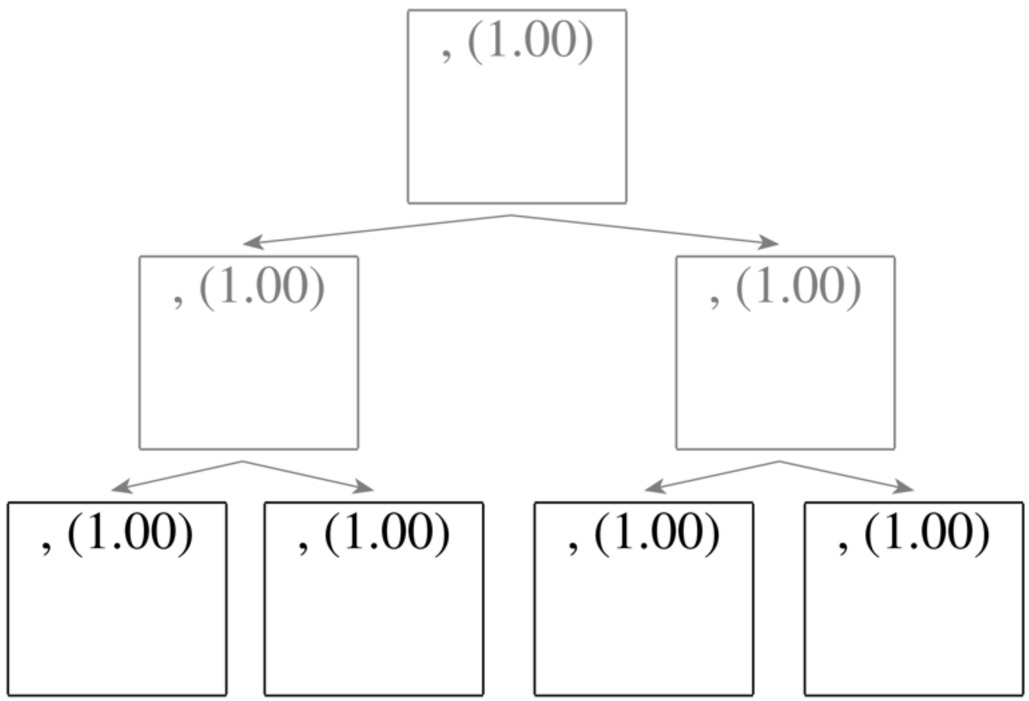
Hierarchical Estimation Results



Model	*F1*
Flat Training	87.3
Hierarchical Training	88.4

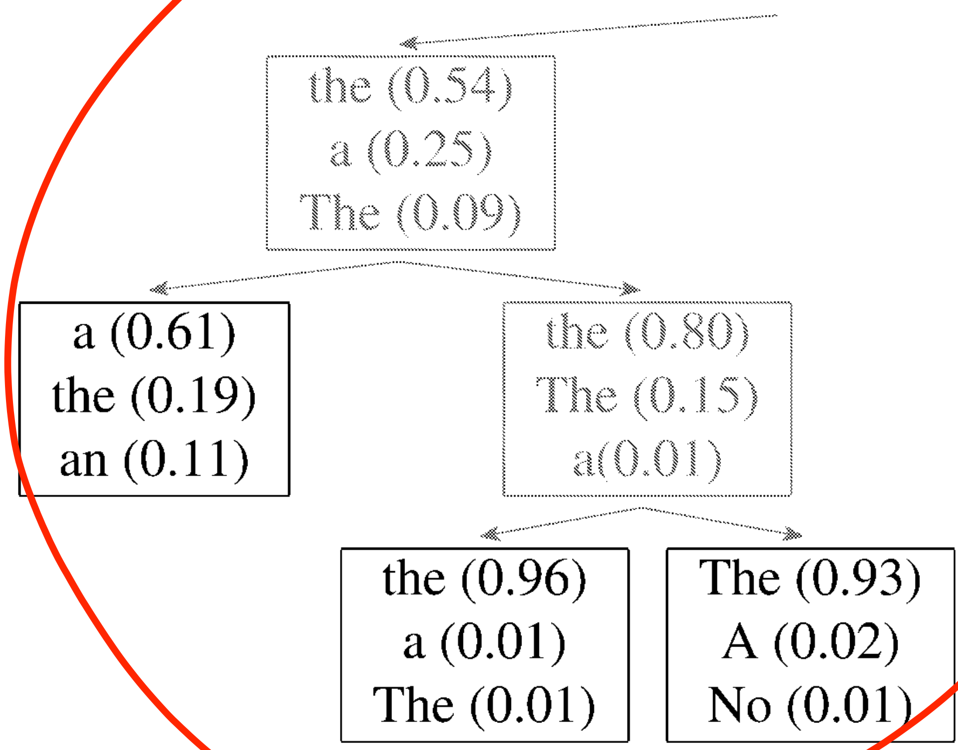
Refinement of the , tag

- Splitting all categories equally is wasteful:

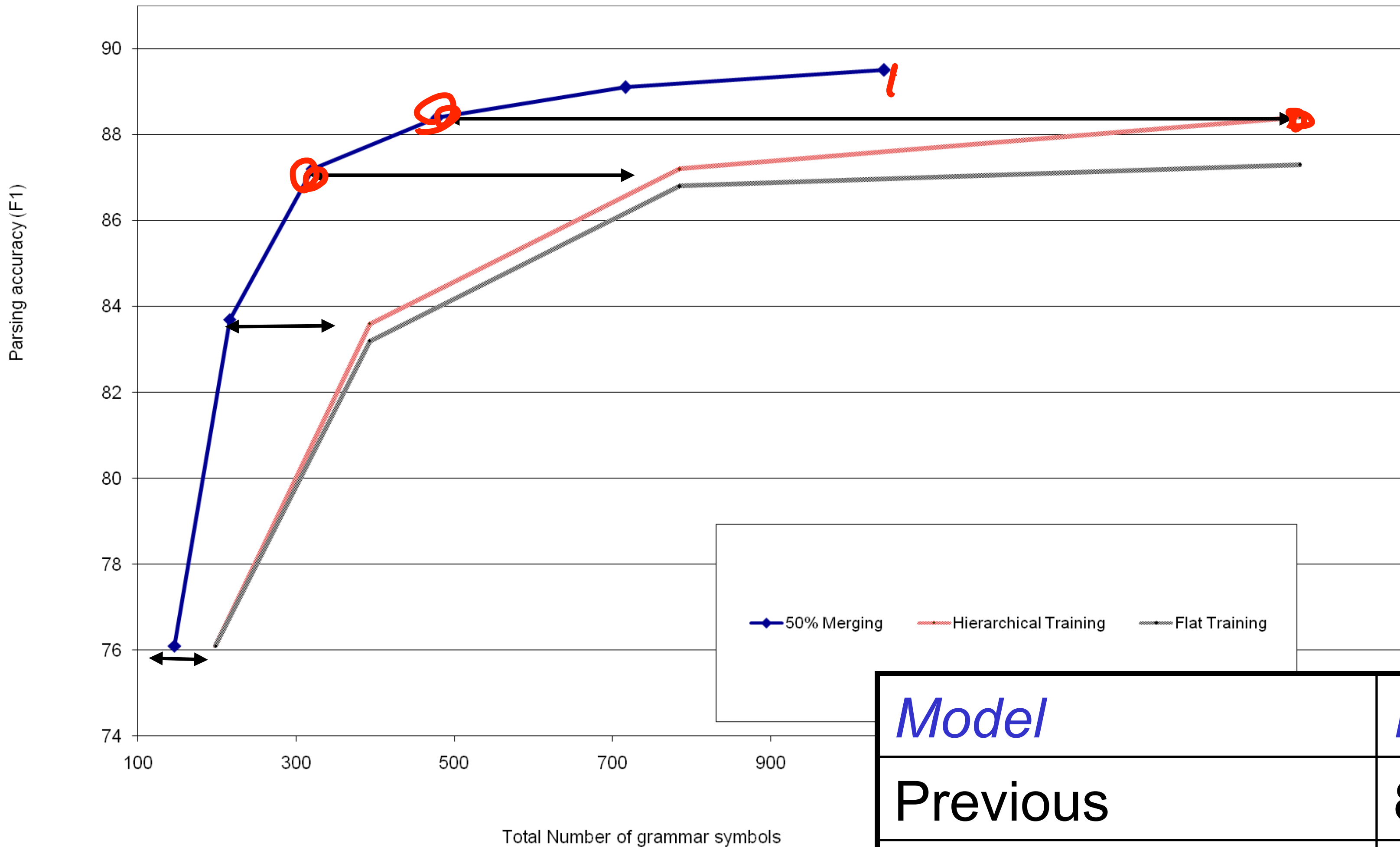


Adaptive Splitting

- Want to split complex categories more
- Idea: split everything, roll back splits which were least useful

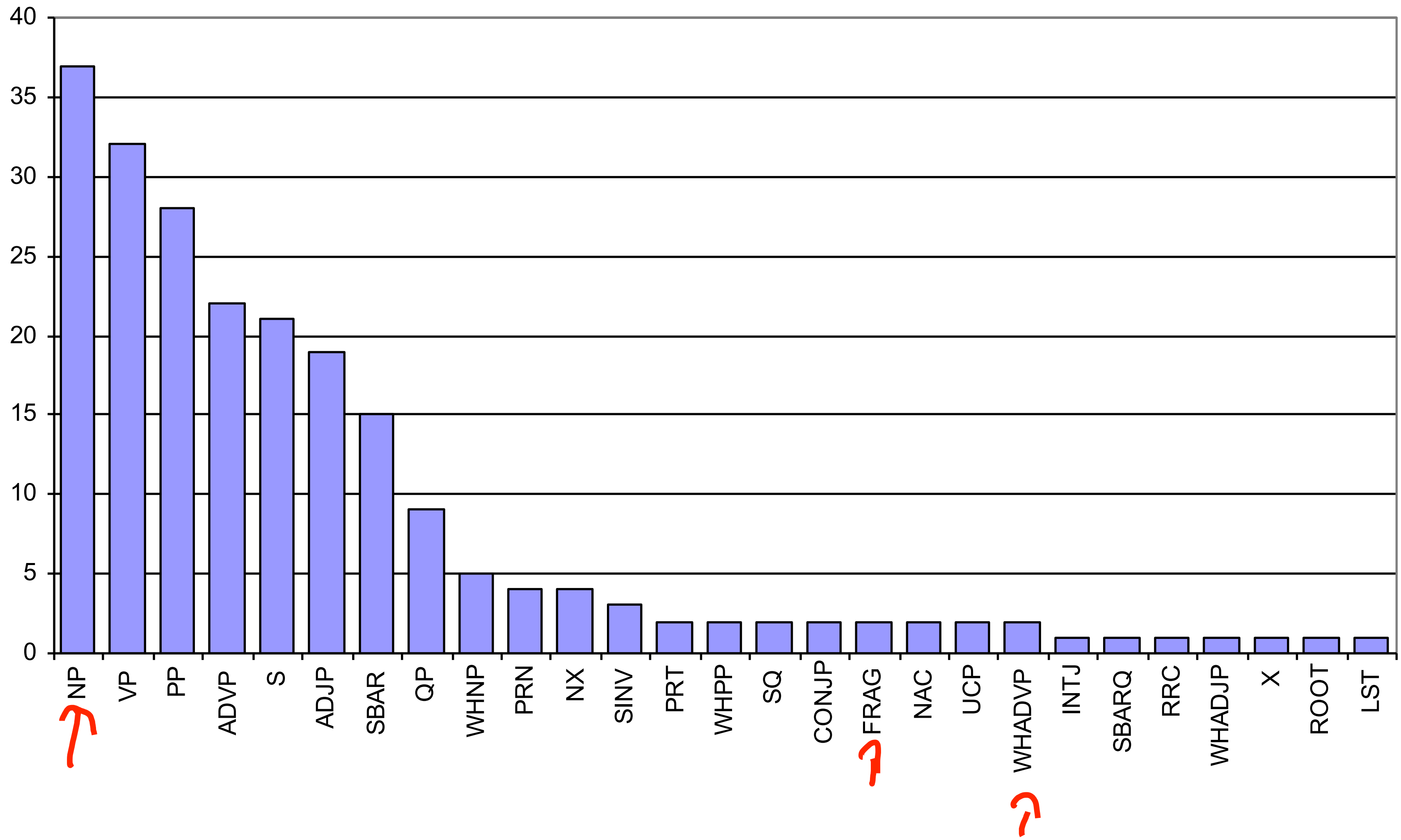


Adaptive Splitting Results

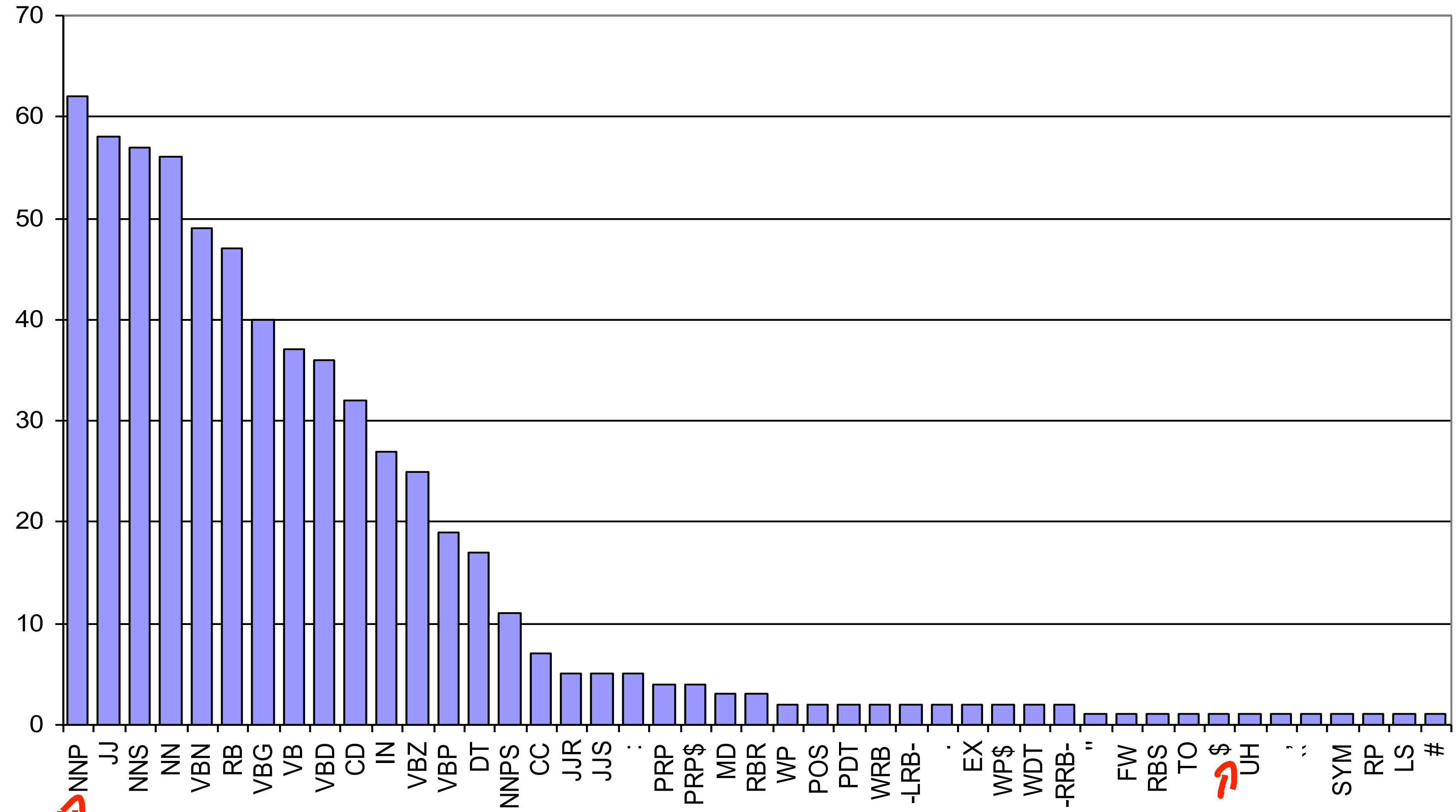


Model	F1
Previous	88.4
With 50% Merging	89.5

Number of Phrasal Subcategories



Number of Lexical Subcategories



Learned Splits

- Proper Nouns (NNP):

NNP-14	Oct.	Nov.	Sept.
NNP-12	John	Robert	James
NNP-2	J.	E.	L.
NNP-1	Bush	Noriega	Peters
NNP-15	New	San	Wall
NNP-3	York	Francisco	Street

- Personal pronouns (PRP):

PRP-0	It	He	I
PRP-1	it	he	they
PRP-2	it	them	him

Learned Splits

- Relative adverbs (RBR):

RBR-0	further	lower	higher
RBR-1	more	less	More
RBR-2	earlier	Earlier	later

- Cardinal Numbers (CD):

CD-7	one	two	Three
CD-4	1989	1990	1988
CD-11	million	billion	trillion
CD-0	1	50	100
CD-3	1	30	31
CD-9	78	58	34

Final Results (Accuracy)

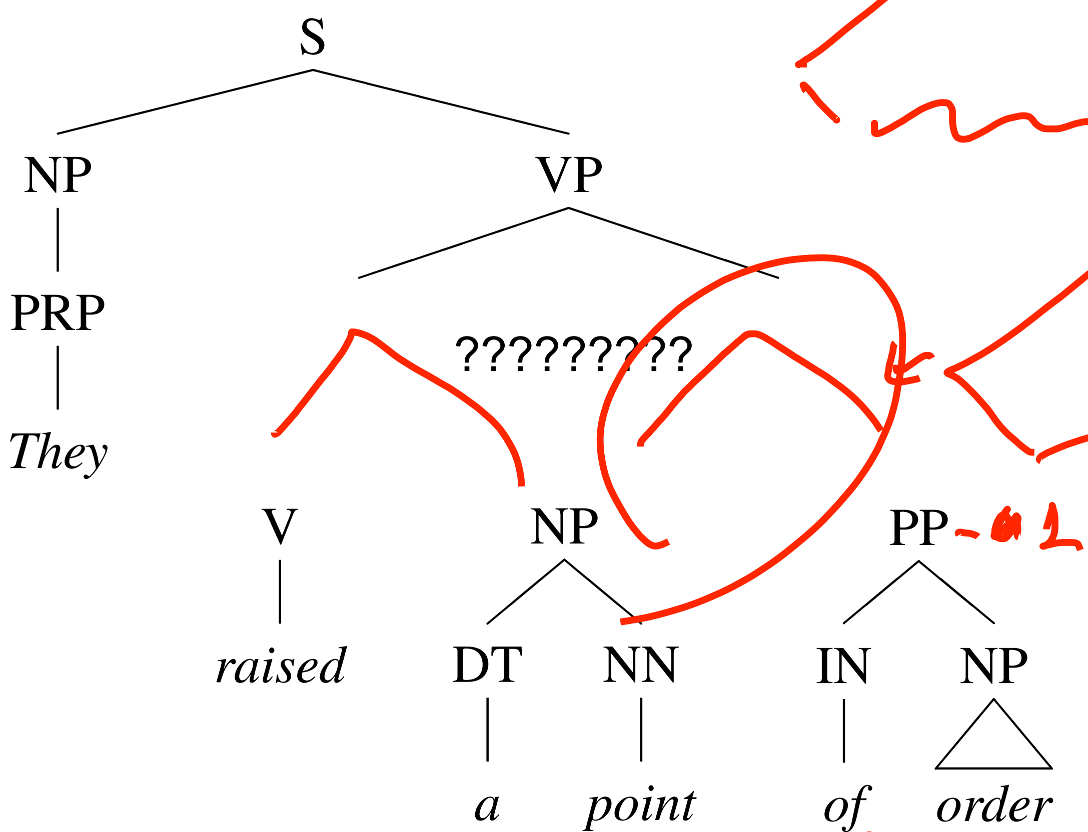
		≤ 40 words F1	all F1
ENG	Charniak&Johnson '05 (generative)	90.1	89.6
	Split / Merge	**90.6**	**90.1**
GER	Dubey '05	76.3	-
	Split / Merge	**80.8**	**80.1**
CHN	Chiang et al. '02	80.0	76.6
	Split / Merge	**86.3**	**83.4**

Still higher numbers from reranking / self-training methods

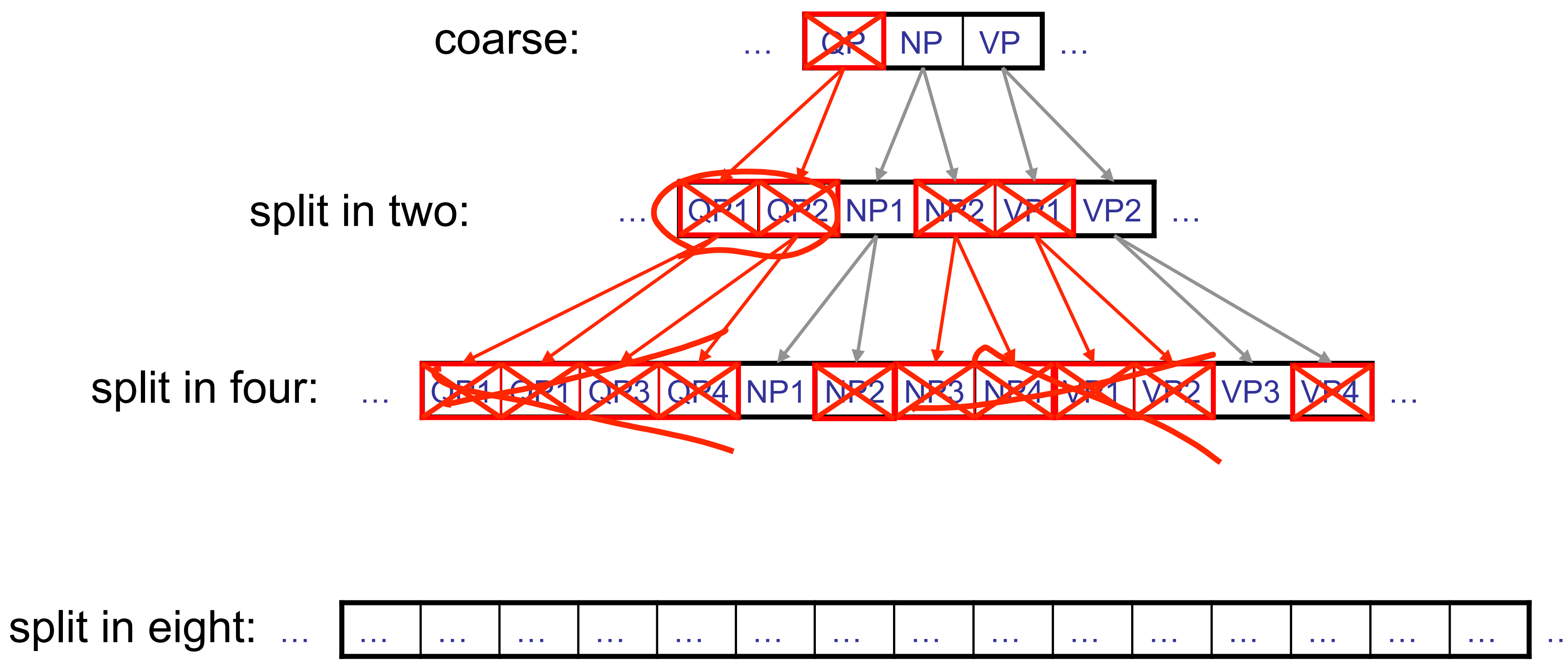
Efficient Parsing for Hierarchical Grammars

Coarse-to-Fine Inference

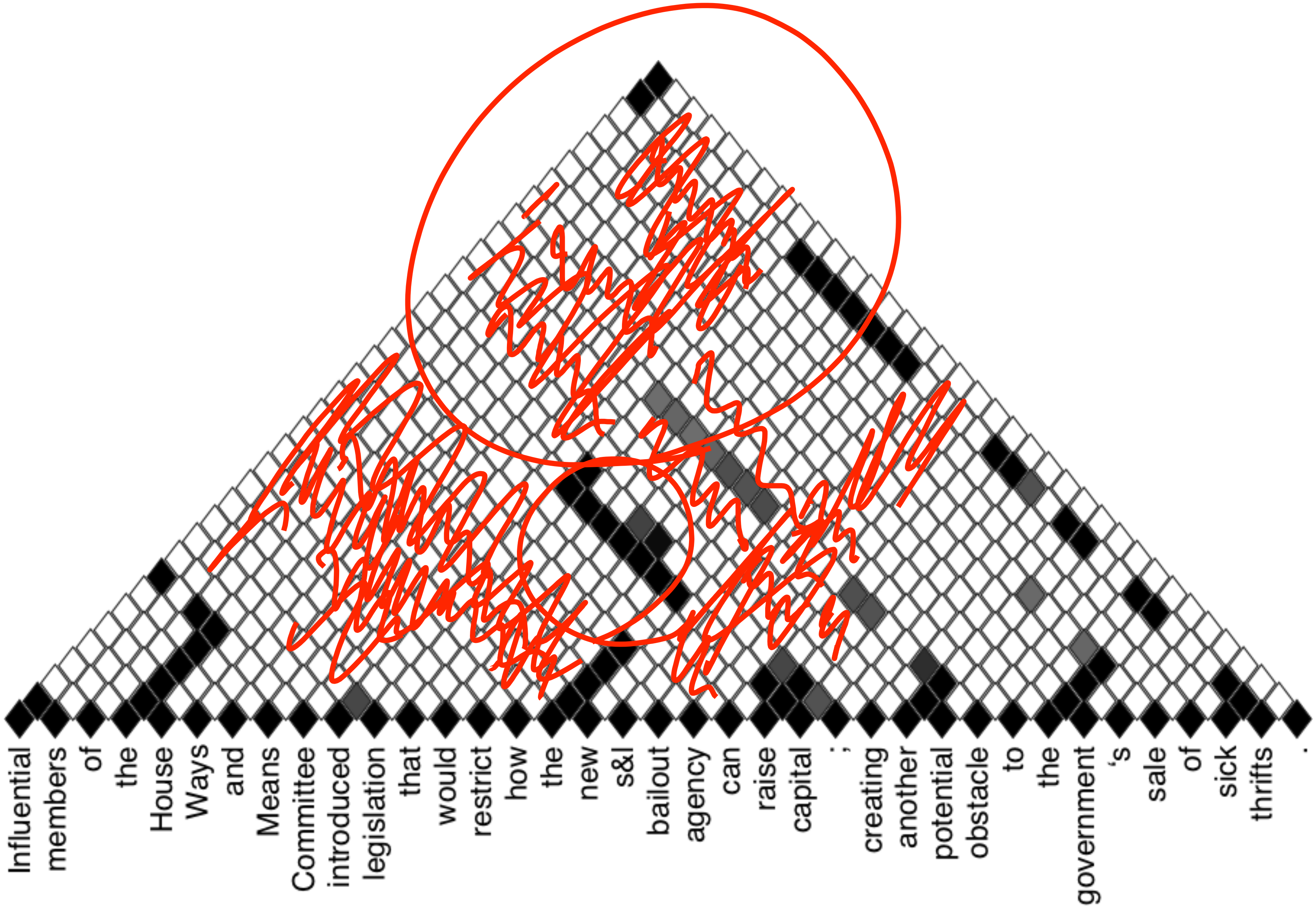
- Example: PP attachment



Hierarchical Pruning



Bracket Posteriors



1621 min

111 min

35 min

15 min
(no search error)

Results

- Stanford Parser – 86.3 (unlex / struct annotation)
- Collins 99 – 88.6 F1 (lexical)
- Charniak and Johnson 05 – 89.7 / 91.3 F1 (lexical + rerank)
- McClosky et al 06 – 92.1 F1 (lexical + rerank + self-train)

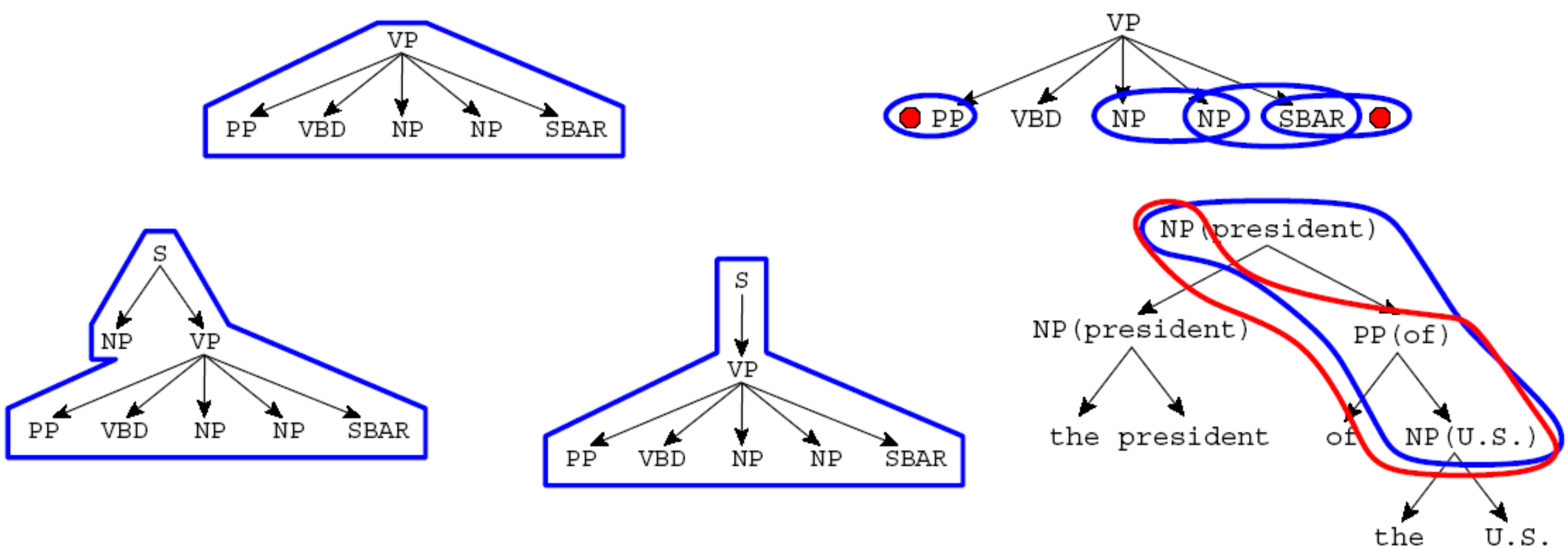
- Petrov et al 06 – 90.7 F1 (unlex / latent vars)
- Petrov et al 10 – 91.8 (unlex / latent vars + ensemble)

- Socher et al 13 – 90.4 (unlex + neural rerank)
- Vinyals et al 15 – 90.5 / 92.1 (neural sequence + self-train)
- Dyer et al 16 – 92.4 (neural shift-reduce)

...many more that are really cool (e.g. Hall and Klein 12,14)

Parse Reranking

- Assume the number of parses is very small
- We can represent each parse T as a feature vector $\varphi(T)$
  - Typically, all local rules are features
  - Also non-local features, like how right-branching the overall tree is
  - [Charniak and Johnson 05] gives a rich set of features



Results

le chat

the cat ran

- Stanford Parser – 86.3 (unlex / struct annotation)
- Collins 99 – 88.6 F1 (lexical)
- Charniak and Johnson 05 – 89.7 / 91.3 F1 (lexical + rerank)
- McClosky et al 06 – 92.1 F1 (lexical + rerank + self-train)

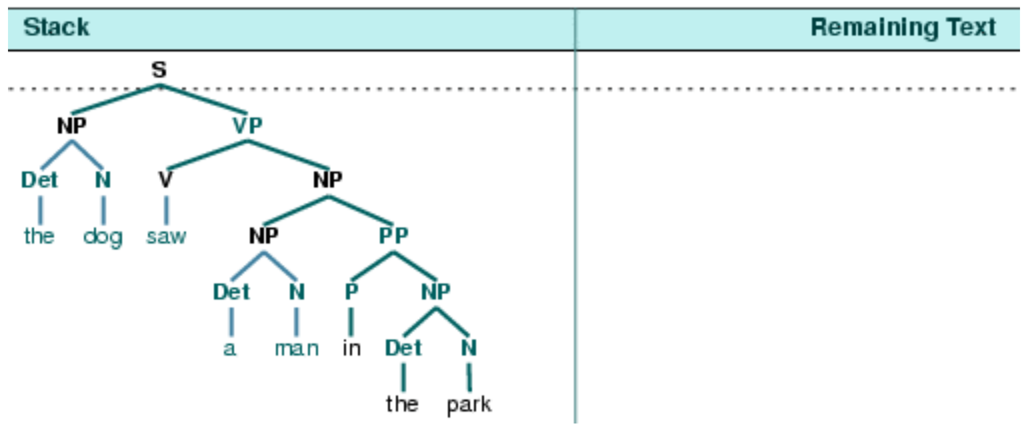
- Petrov et al 06 – 90.7 F1 (unlex / latent vars)
- Petrov et al 10 – 91.8 (unlex / latent vars + ensemble)

- Socher et al 13 – 90.4 (unlex + neural rerank)
- Vinyals et al 15 – 90.5 / 92.1 (neural sequence + self-train)
- Dyer et al 16 – 92.4 (neural shift-reduce)

…many more that are really cool (e.g. Hall and Klein 12,14)

Shift-Reduce Parsers

- Another way to derive a tree:



- Parsing
  - No useful dynamic programming search
  - Can still use beam search [Ratnaparkhi 97]

Results

- Stanford Parser – 86.3 (unlex / struct annotation)
- Collins 99 – 88.6 F1 (lexical)
- Charniak and Johnson 05 – 89.7 / 91.3 F1 (lexical + rerank)
- McClosky et al 06 – 92.1 F1 (lexical + rerank + self-train)

- Petrov et al 06 – 90.7 F1 (unlex / latent vars)
- Petrov et al 10 – 91.8 (unlex / latent vars + ensemble)

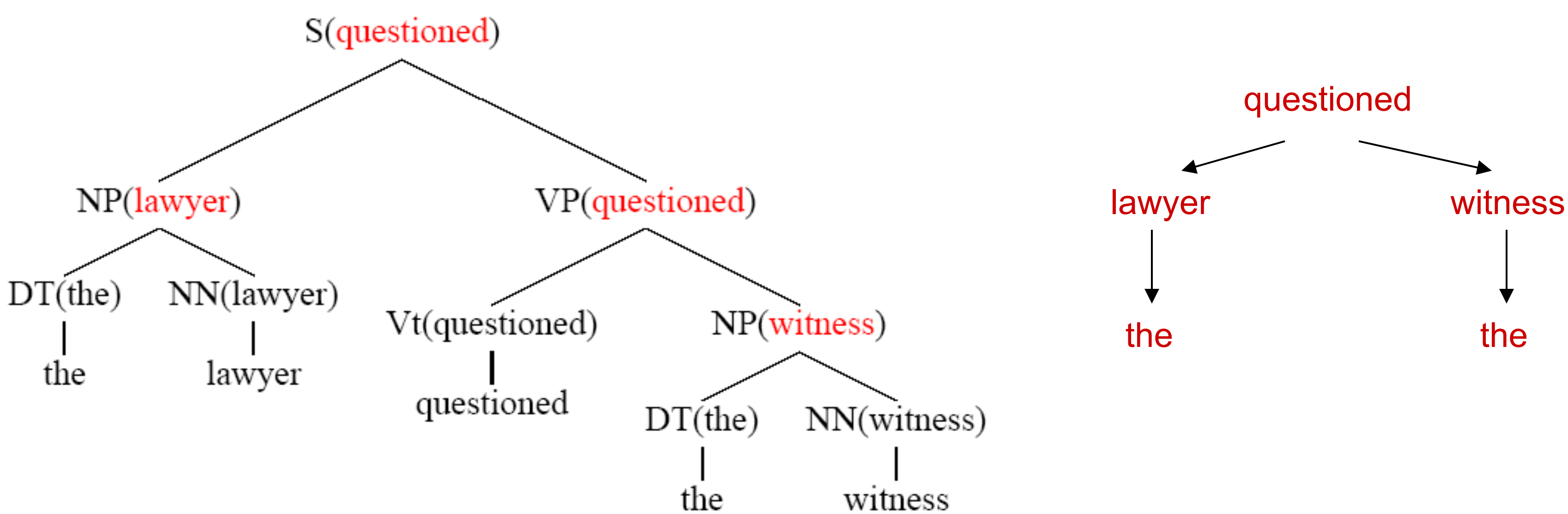
- Socher et al 13 – 90.4 (unlex + neural rerank)
- Vinyals et al 15 – 90.5 / 92.1 (neural sequence + self-train)
- Dyer et al 16 – 92.4 (neural shift-reduce)

…many more that are really cool (e.g. Hall and Klein 12,14)

Other Syntactic Models

Dependency Parsing

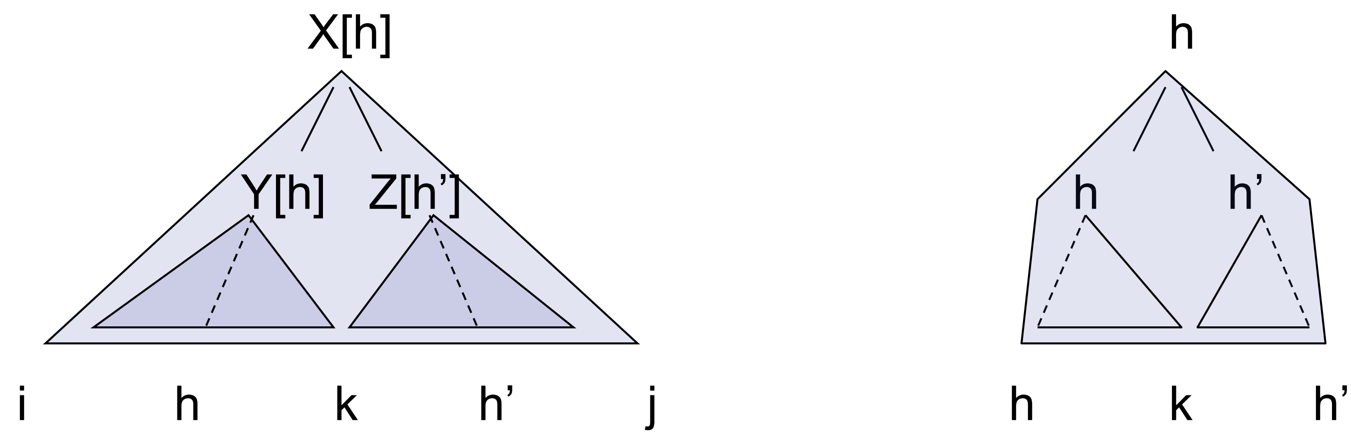
- Lexicalized parsers can be seen as producing *dependency trees*



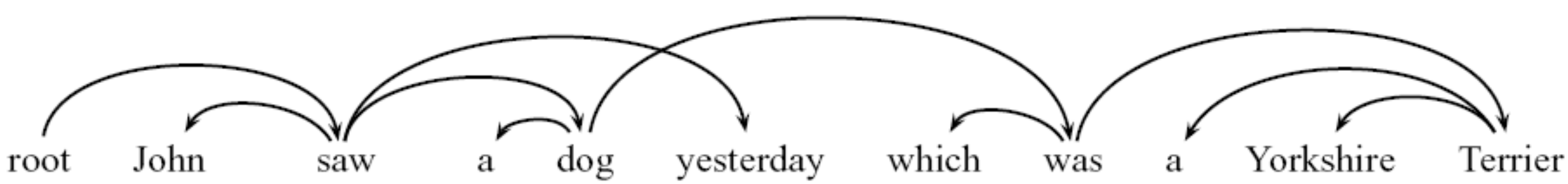
- Each local binary tree corresponds to an attachment in the dependency graph

Dependency Parsing

- Pure dependency parsing is only cubic [Eisner 99]

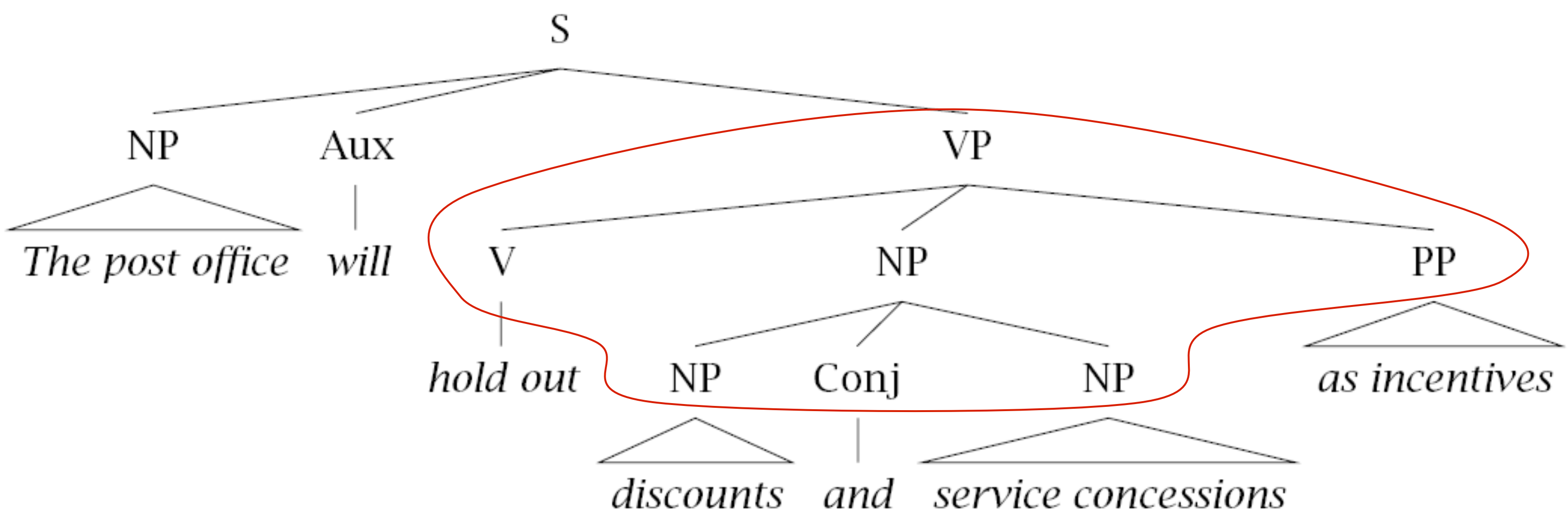


- Some work on *non-projective* dependencies
  - Common in, e.g. Czech parsing
  - Can do with MST algorithms [McDonald and Pereira 05]



Tree Insertion Grammars

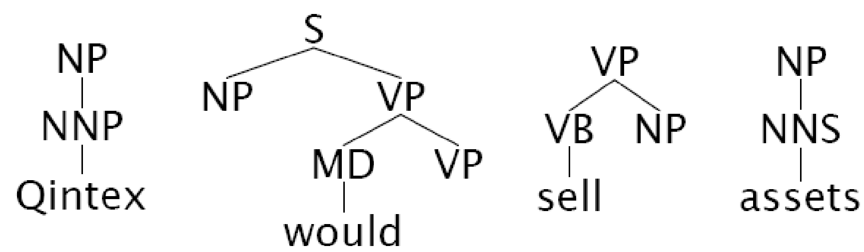
- Rewrite large (possibly lexicalized) subtrees in a single step

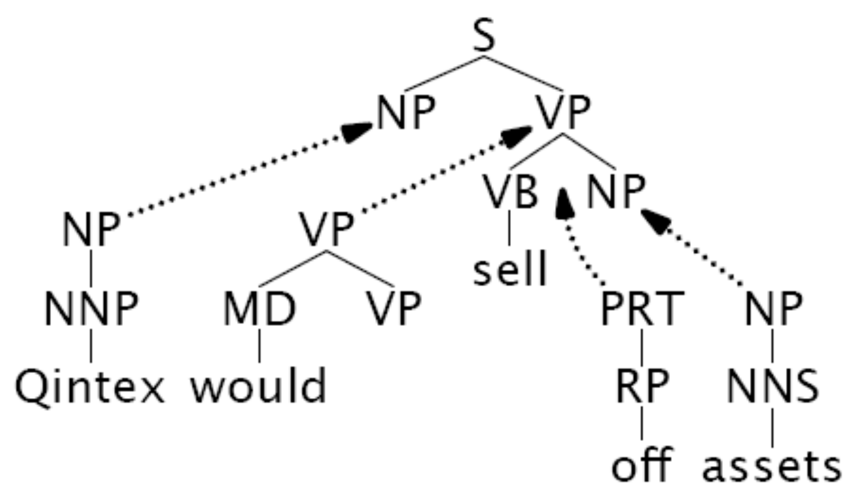


- Formally, a *tree-insertion grammar*
- Derivational ambiguity whether subtrees were generated atomically or compositionally
- Most probable *parse* is NP-complete

Tree-adjoining grammars

- Start with *local trees*
- Can insert structure with *adjunction* operators
- Mildly context-sensitive
- Models long-distance dependencies naturally
- … as well as other weird stuff that CFGs don't capture well (e.g. cross-serial dependencies)





CCG Parsing

- Combinatory Categorial Grammar
  - Fully (mono-) lexicalized grammar
  - Categories encode argument sequences
  - Very closely related to the lambda calculus (more later)
  - Can have spurious ambiguities (why?)

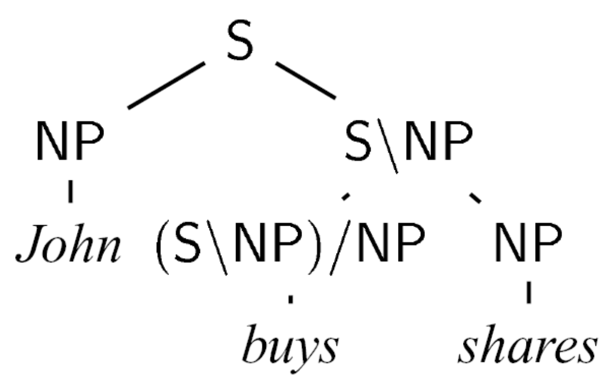
$$\textit{John} \vdash \mathsf{NP}$$

$$\textit{shares} \vdash \mathsf{NP}$$

$$\textit{buys} \vdash (\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}$$

$$\textit{sleeps} \vdash \mathsf{S}\backslash\mathsf{NP}$$

$$\textit{well} \vdash (\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP})$$



Classification

Classification

- Automatically make a decision about inputs
  - Example: document → category
  - Example: image of digit → digit
  - Example: image of object → object type
  - Example: query + webpages → best match
  - Example: symptoms → diagnosis
  - …
- Three main ideas
  - Representation as feature vectors / kernel functions
  - Scoring by linear functions
  - Learning by optimization

Some Definitions

INPUTS	$\mathbf{x}_i$	*close the ____*
CANDIDATE SET	$\mathcal{Y}(\mathbf{x})$	*{door, table, …}*
CANDIDATES	$\mathbf{y}$	*table*
TRUE OUTPUTS	$\mathbf{y}_i^*$	*door*
FEATURE VECTORS	$\mathbf{f}(\mathbf{x}, \mathbf{y})$	[0 0 1 0 0 0 1 0 0 0 0 0]

x_{-1}="the" ∧ y="door"

x_{-1}="the" ∧ y="table"

"close" in x ∧ y="door"

y occurs in x

Features

Block Feature Vectors

- Sometimes, we think of the input as having features, which are multiplied by outputs to form the candidates

$\mathbf{x}$ *… win the election …*

"$\mathbf{f}(\mathbf{x})$" [1 0 1 0]

"*win*" "*election*"

… win the election …

$$\mathbf{f}(\textcolor{red}{SPORTS}) = [\textcolor{red}{1\ 0\ 1\ 0}\ \textcolor{blue}{0\ 0\ 0\ 0}\ \textcolor{green}{0\ 0\ 0\ 0}]$$

… win the election …

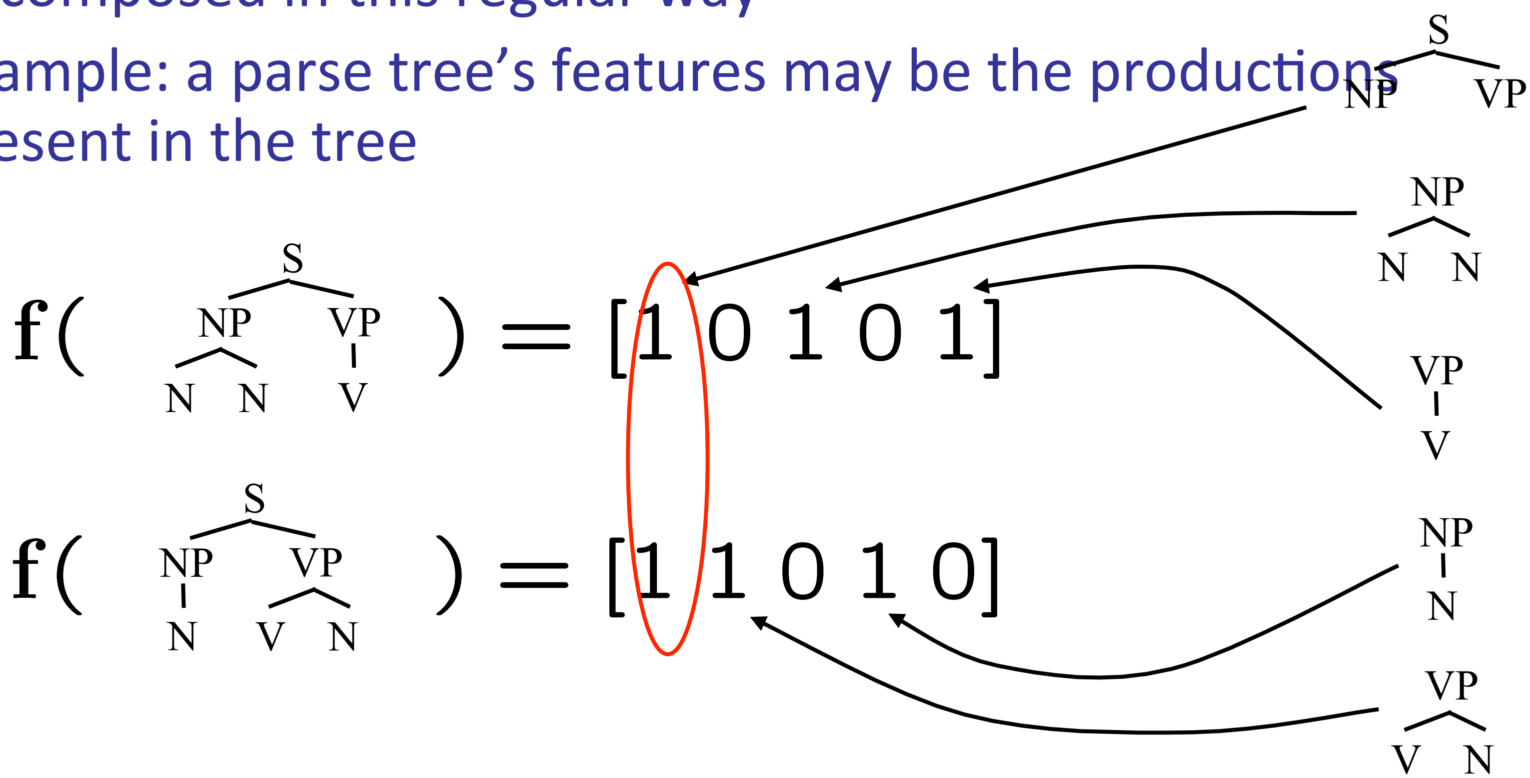
$$\mathbf{f}(\textcolor{blue}{POLITICS}) = [\textcolor{red}{0\ 0\ 0\ 0}\ \textcolor{blue}{1\ 0\ 1\ 0}\ \textcolor{green}{0\ 0\ 0\ 0}]$$

… win the election …

$$\mathbf{f}(\textcolor{green}{OTHER}) = [\textcolor{red}{0\ 0\ 0\ 0}\ \textcolor{blue}{0\ 0\ 0\ 0}\ \textcolor{green}{1\ 0\ 1\ 0}]$$

Non-Block Feature Vectors

- Sometimes the features of candidates cannot be decomposed in this regular way
- Example: a parse tree's features may be the productions present in the tree



- Different candidates will thus often share features
- We'll return to the non-block case later

Linear Models

Linear Models: Scoring

- In a linear model, each feature gets a weight w

$$\mathbf{f}(\overset{\text{... win the election ...}}{POLITICS}) = [\ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]$$

$$\mathbf{f}(\overset{\text{... win the election ...}}{SPORTS}) = [\ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]$$

$$\mathbf{w} = [\ 1\ \ 1\ \ -1\ \ -2\ \ 1\ \ -1\ \ 1\ \ -2\ \ -2\ \ -1\ \ -1\ \ 1]$$

- We score hypotheses by multiplying features and weights:

$$score(\mathbf{y}, \mathbf{w}) = \mathbf{w}^{\top}\mathbf{f}(\mathbf{y})$$

$$\mathbf{f}(\overset{\text{... win the election ...}}{POLITICS}) = [\ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]$$

$$\mathbf{w} = [\ 1\ \ 1\ \ -1\ \ -2\ \ 1\ \ -1\ \ 1\ \ -2\ \ -2\ \ -1\ \ -1\ \ 1]$$

$$score(\overset{\text{... win the election ...}}{POLITICS}, \mathbf{w}) = 1 \times 1 + 1 \times 1 = 2$$

Linear Models: Decision Rule

- The linear decision rule:

$$prediction(\text{... win the election ...}, \mathbf{w}) = \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \mathbf{w}^\top \mathbf{f}(\mathbf{y})$$

$$score(\overset{\text{... win the election ...}}{SPORTS}, \mathbf{w}) = 1 \times 1 + (-1) \times 1 = 0$$

$$score(\overset{\text{... win the election ...}}{POLITICS}, \mathbf{w}) = 1 \times 1 + 1 \times 1 = 2$$

$$score(\overset{\text{... win the election ...}}{OTHER}, \mathbf{w}) = (-2) \times 1 + (-1) \times 1 = -3$$

$$\Downarrow$$

$$prediction(\text{... win the election ...}, \mathbf{w}) = \overset{\text{... win the election ...}}{POLITICS}$$

- We've said nothing about where weights come from

Binary Classification

- Important special case: binary classification
  - Classes are y=+1/-1

$$\mathbf{f}(\mathbf{x}, -1) = -\mathbf{f}(\mathbf{x}, +1)$$

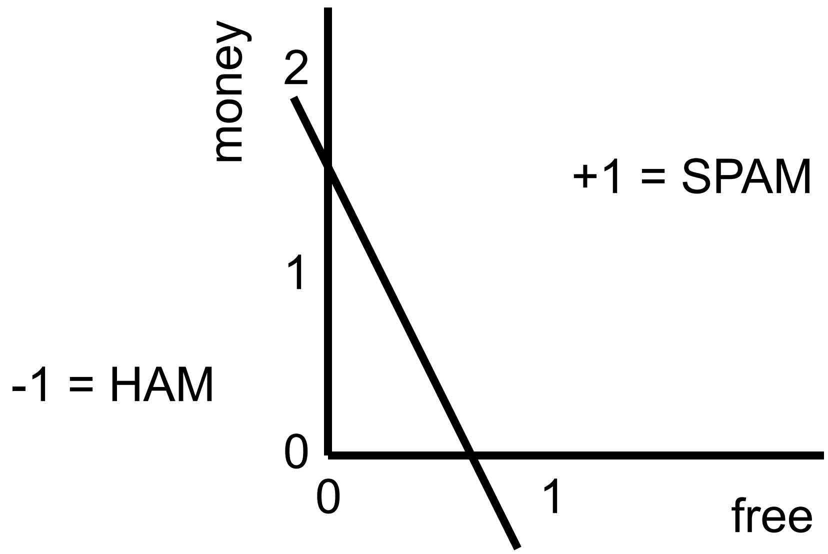
$$\mathbf{f}(\mathbf{x}) = 2\mathbf{f}(\mathbf{x}, +1)$$

  - Decision boundary is a hyperplane

$$\mathbf{w}^\top \mathbf{f}(\mathbf{x}) = 0$$

$\mathbf{w}$

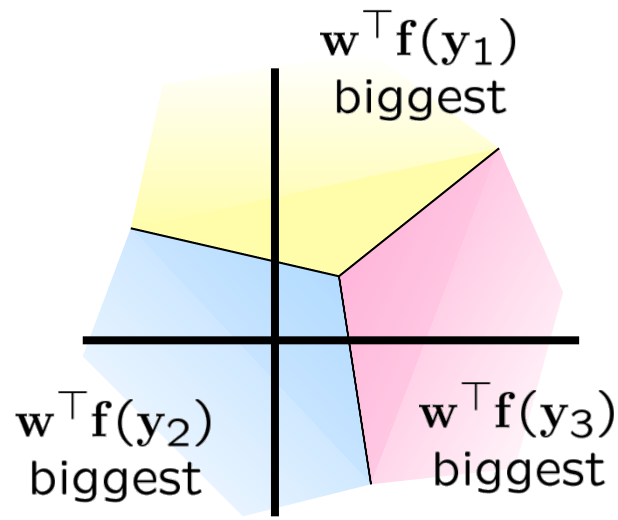
BIAS	:	-3
free	:	4
money	:	2



$$\mathbf{w}^\top \mathbf{f} = 0$$

Multiclass Decision Rule

- If more than two classes:
  - Highest score wins
  - Boundaries are more complex
  - Harder to visualize



$$prediction(\mathbf{x}_i, \mathbf{w}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$$

- There are other ways: e.g. reconcile pairwise decisions

Learning

Learning Classifier Weights

- Two broad approaches to learning weights

- Generative: work with a probabilistic model of the data, weights are (log) local conditional probabilities
  - Advantages: learning weights is easy, smoothing is well-understood, backed by understanding of modeling

- Discriminative: set weights based on some error-related criterion
  - Advantages: error-driven, often weights which are good for classification aren't the ones which best describe the data

- We'll mainly talk about the latter for now

How to pick weights?

- Goal: choose "best" vector w given training data
  - For now, we mean "best for classification"
- The ideal: the weights which have greatest test set accuracy / F1 / whatever
  - But, don't have the test set
  - Must compute weights from training set
- Maybe we want weights which give best training set accuracy?
  - Hard discontinuous optimization problem
  - May not (does not) generalize to test set
  - Easy to overfit

Though, min-error training for MT does exactly this.

Minimize Training Error?

- A loss function declares how costly each mistake is

$$\ell_i(\mathbf{y}) = \ell(\mathbf{y}, \mathbf{y}_i^*)$$

  - E.g. 0 loss for correct label, 1 loss for wrong label
  - Can weight mistakes differently (e.g. false positives worse than false negatives or Hamming distance over structured labels)

- We could, in principle, minimize training loss:

$$\min_{\mathbf{w}} \sum_i \ell_i \left(\arg\max_{\mathbf{y}} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) \right)$$

- This is a hard, discontinuous optimization problem

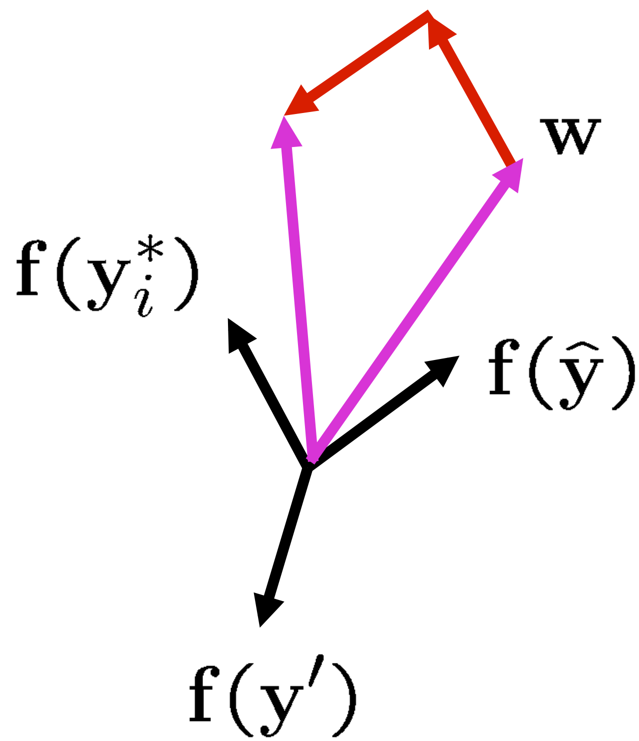
Linear Models: Perceptron

- The perceptron algorithm
  - Iteratively processes the training set, reacting to training errors
  - Can be thought of as trying to drive down training error
- The (online) perceptron algorithm:
  - Start with zero weights w
  - Visit training instances one by one
    - Try to classify

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \mathbf{w}^\top \mathbf{f}(\mathbf{y})$$

    - If correct, no change!
    - If wrong: adjust weights

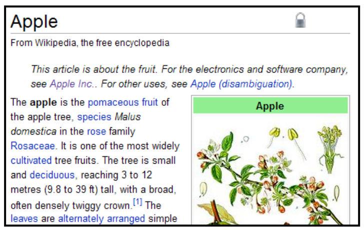
$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{f}(\mathbf{y}_i^*)$$
$$\mathbf{w} \leftarrow \mathbf{w} - \mathbf{f}(\hat{\mathbf{y}})$$



Example: “Best” Web Page

$$\mathbf{w} = [1 \quad 2 \quad 0 \quad 0 \quad \ldots]$$

x_i = “Apple Computers”

$\mathbf{f}_i($  $) = [0.3\ 5\ 0\ 0\ \ldots]$ $\quad \mathbf{w}^\top \mathbf{f} = 10.3 \quad \hat{\mathbf{y}}$

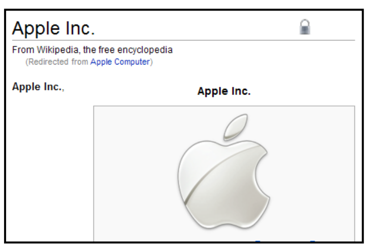
Apple

From Wikipedia, the free encyclopedia

This article is about the fruit. For the electronics and software company, see Apple Inc.. For other uses, see Apple (disambiguation).

The **apple** is the pomaceous fruit of the apple tree, species *Malus domestica* in the rose family Rosaceae. It is one of the most widely cultivated tree fruits. The tree is small and deciduous, reaching 3 to 12 metres (9.8 to 39 ft) tall, with a broad, often densely twiggy crown.[1] The leaves are alternately arranged simple

Apple

$\mathbf{f}_i($  $) = [0.8\ 4\ 2\ 1\ \ldots]$ $\quad \mathbf{w}^\top \mathbf{f} = 8.8 \quad \mathbf{y}_i^*$

Apple Inc.

From Wikipedia, the free encyclopedia

(Redirected from Apple Computer)

Apple Inc.,

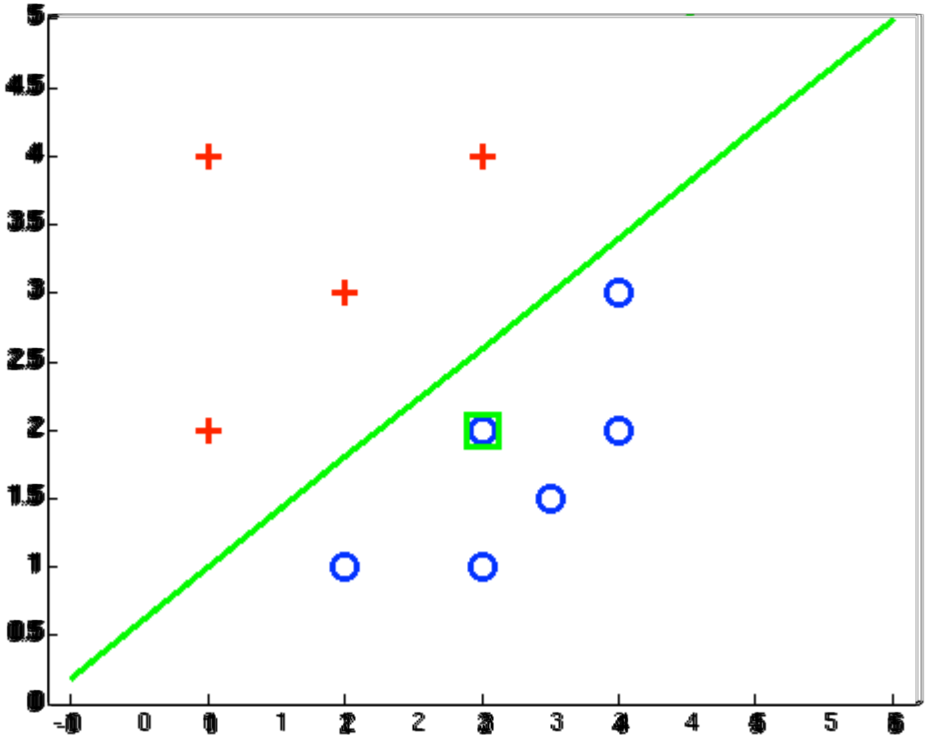
Apple Inc.

$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{f}(\mathbf{y}_i^*) - \mathbf{f}(\hat{\mathbf{y}})$$

$$\mathbf{w} = [1.5 \quad 1 \quad 2 \quad 1 \quad \ldots]$$

Examples: Perceptron

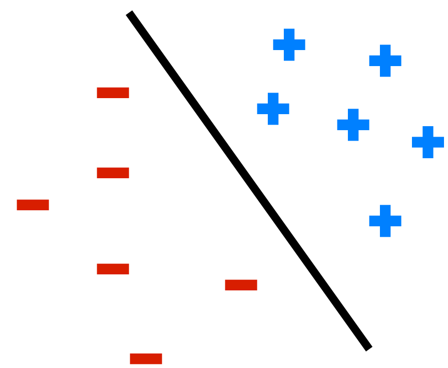
- Separable Case



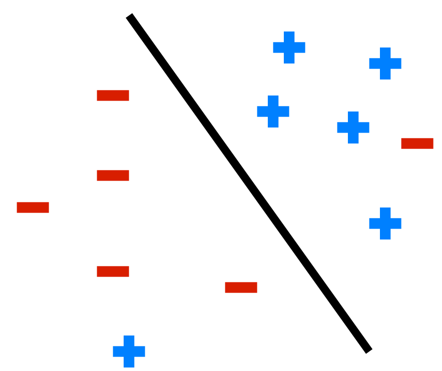
Perceptrons and Separability

- A data set is separable if some parameters classify it perfectly
- Convergence: if training data separable, perceptron will separate (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

Separable

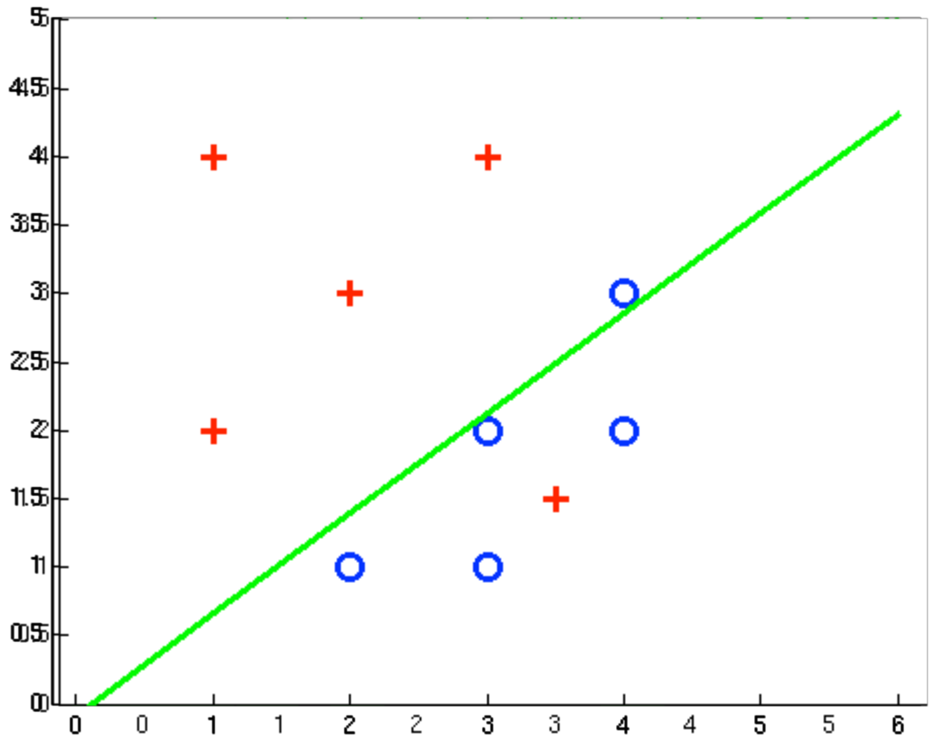


Non-Separable



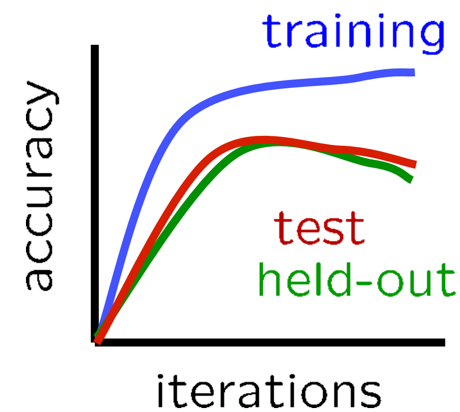
Examples: Perceptron

- Non-Separable Case

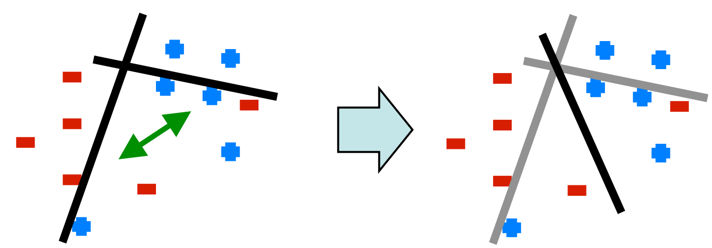


Issues with Perceptrons

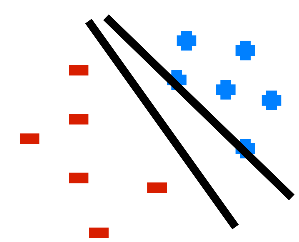
- Overtraining: test / held-out accuracy usually rises, then falls
  - Overtraining isn't the typically discussed source of overfitting, but it can be important



- Regularization: if the data isn't separable, weights often thrash around
  - Averaging weight vectors over time can help (averaged perceptron)
  - [Freund & Schapire 99, Collins 02]



- Mediocre generalization: finds a "barely" separating solution

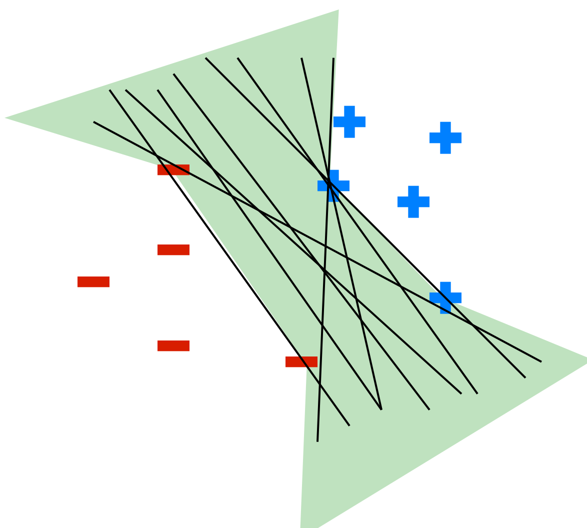


Problems with Perceptrons

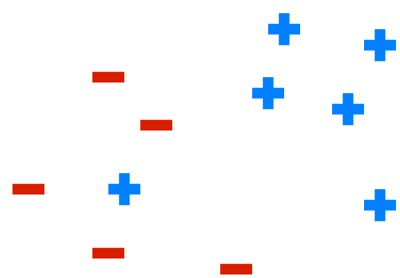
- Perceptron "goal": separate the training data

$$\forall i, \forall \mathbf{y} \neq \mathbf{y}^i \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$$

1. This may be an entire feasible space



2. Or it may be impossible

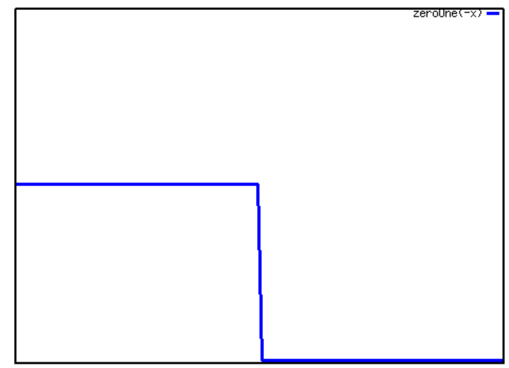


Margin

Objective Functions

- What do we want from our weights?
  - Depends!
  - So far: minimize (training) errors:

$$\sum_i step\left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \max_{\mathbf{y}\neq\mathbf{y}_i^*} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})\right)$$

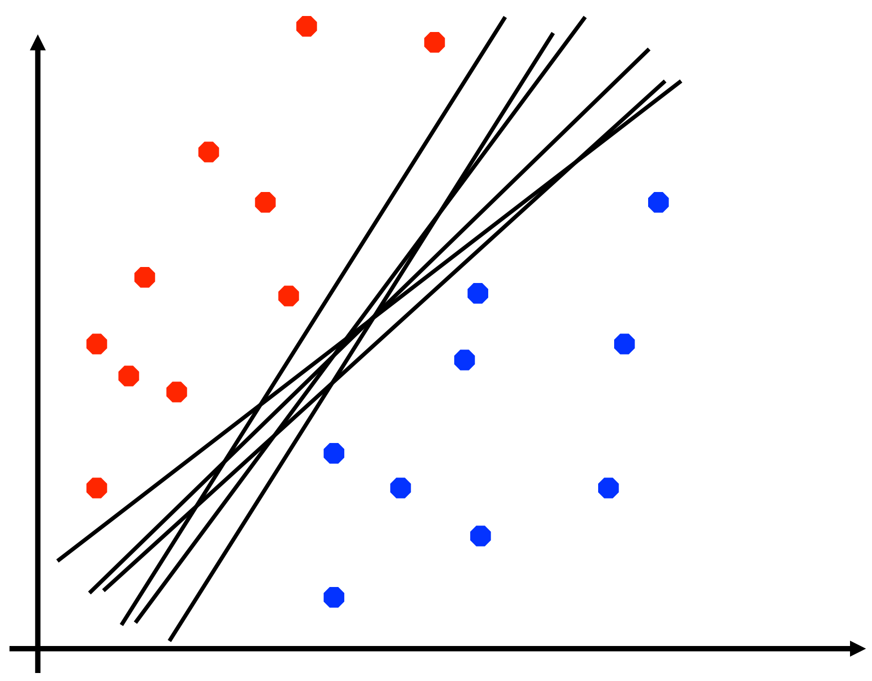


$$\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y}\neq\mathbf{y}_i^*} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$$

  - This is the "zero-one loss"
    - Discontinuous, minimizing is NP-complete
    - Not really what we want anyway
  - Maximum entropy and SVMs have other objectives related to zero-one loss

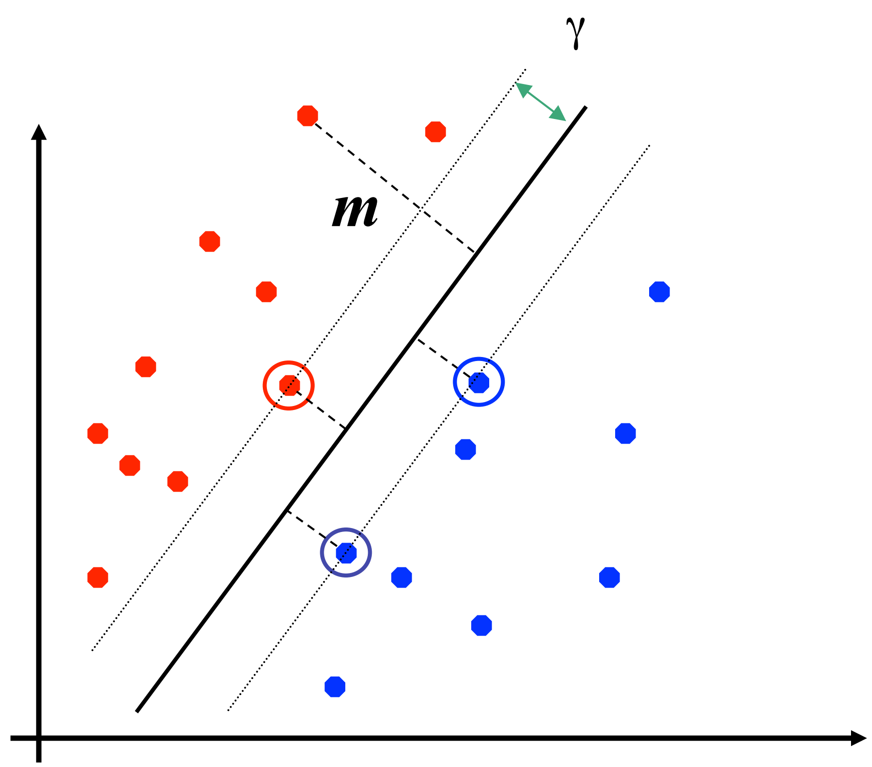
Linear Separators

- Which of these linear separators is optimal?



Classification Margin (Binary)

- Distance of $\mathbf{x}_i$ to separator is its margin, m_i
- Examples closest to the hyperplane are support vectors
- Margin γ of the separator is the minimum m



Classification Margin

- For each example $\mathbf{x}_i$ and possible mistaken candidate $\mathbf{y}$, we avoid that mistake by a margin $m_i(\mathbf{y})$ (with zero-one loss)

$$m_i(\mathbf{y}) = \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$$

- Margin γ of the entire separator is the minimum m

$$\gamma = \min_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \max_{\mathbf{y} \neq \mathbf{y}_i^*} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) \right)$$

- It is also the largest γ for which the following constraints hold

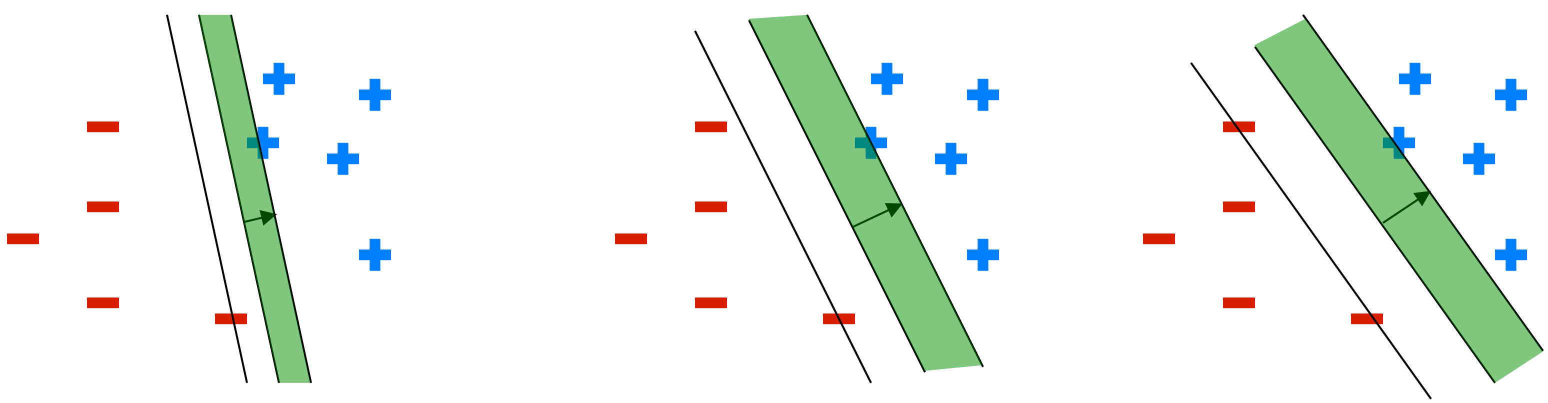
$$\forall i, \forall \mathbf{y} \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \gamma \ell_i(\mathbf{y})$$

Maximum Margin

- Separable SVMs: find the max-margin w

$$\max_{||\mathbf{w}||=1} \gamma \qquad \ell_i(\mathbf{y}) = \begin{cases} 0 & \text{if } \mathbf{y} = \mathbf{y}_i^* \\ 1 & \text{if } \mathbf{y} \neq \mathbf{y}_i^* \end{cases}$$

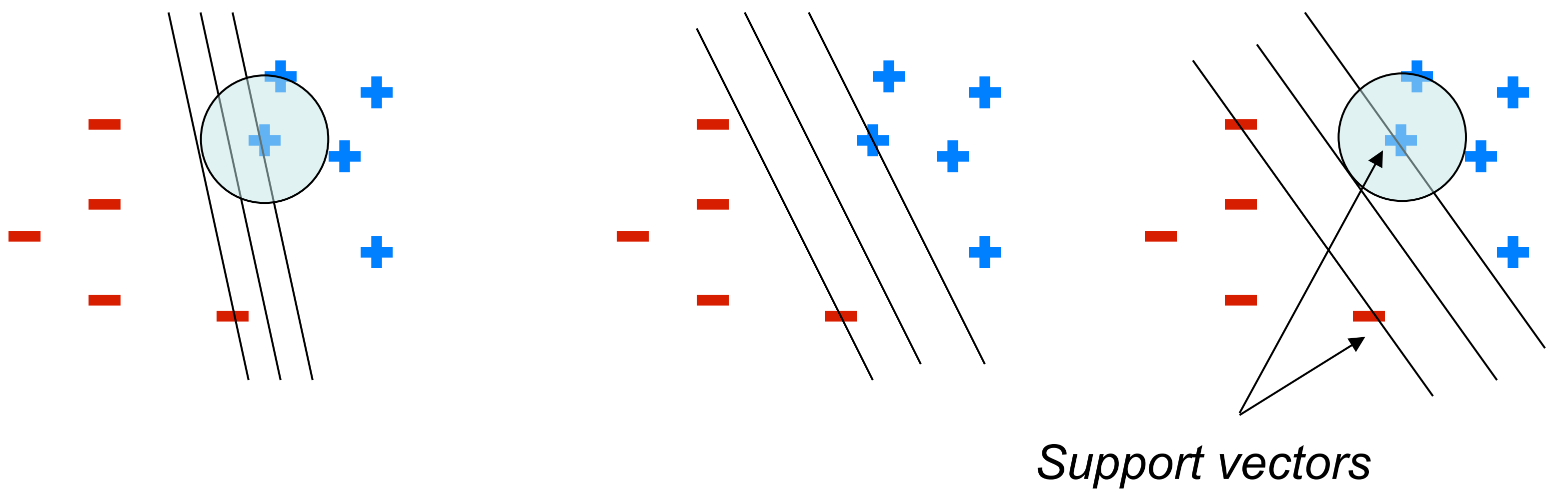
$$\forall i, \forall \mathbf{y} \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \gamma \ell_i(\mathbf{y})$$



- Can stick this into Matlab and (slowly) get an SVM
- Won’t work (well) if non-separable

Why Max Margin?

- Why do this? Various arguments:
  - Solution depends only on the boundary cases, or *support vectors* (but remember how this diagram is broken!)
  - Solution robust to movement of support vectors
  - Sparse solutions (features not in support vectors get zero weight)
  - Generalization bound arguments
  - Works well in practice for many problems



Max Margin / Small Norm

- Reformulation: find the smallest w which separates data

Remember this condition?

$$\max_{||\mathbf{w}||=1} \gamma$$

$$\forall i, \mathbf{y} \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \gamma \ell_i(\mathbf{y})$$

- γ scales linearly in w, so if ||w|| isn't constrained, we can take any separating w and scale up our margin

$$\gamma = \min_{i,\mathbf{y} \neq \mathbf{y}_i^*} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})] / \ell_i(\mathbf{y})$$

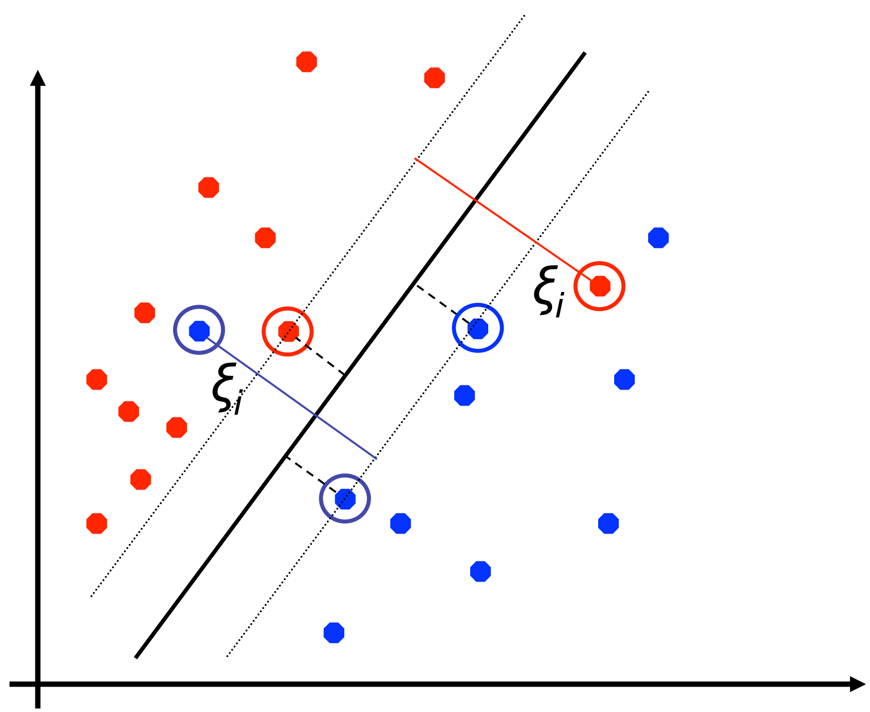
- Instead of fixing the scale of w, we can fix $\gamma = 1$

$$\min_{\mathbf{w}} \frac{1}{2} ||\mathbf{w}||^2$$

$$\forall i, \mathbf{y} \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + 1 \ell_i(\mathbf{y})$$

Soft Margin Classification

- What if the training set is not linearly separable?
- *Slack variables* $\boldsymbol{\xi_i}$ can be added to allow misclassification of difficult or noisy examples, resulting in a *soft margin* classifier



Maximum Margin

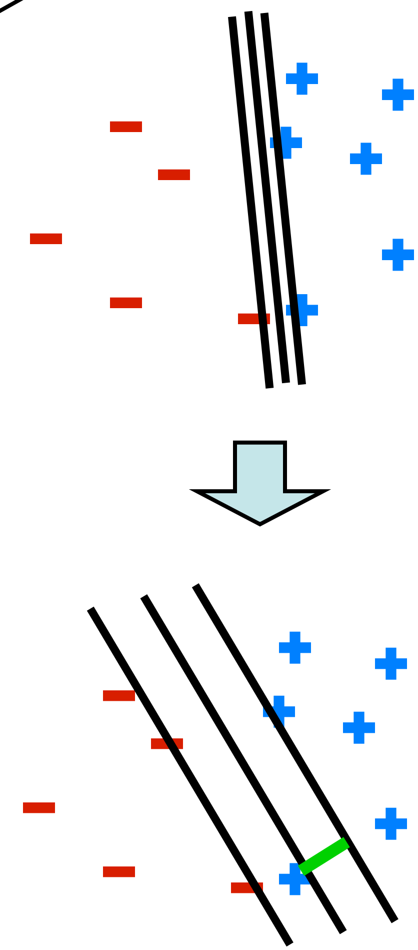
Note: exist other choices of how to penalize slacks!

- Non-separable SVMs
  - Add slack to the constraints
  - Make objective pay (linearly) for slack:

$$\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i$$

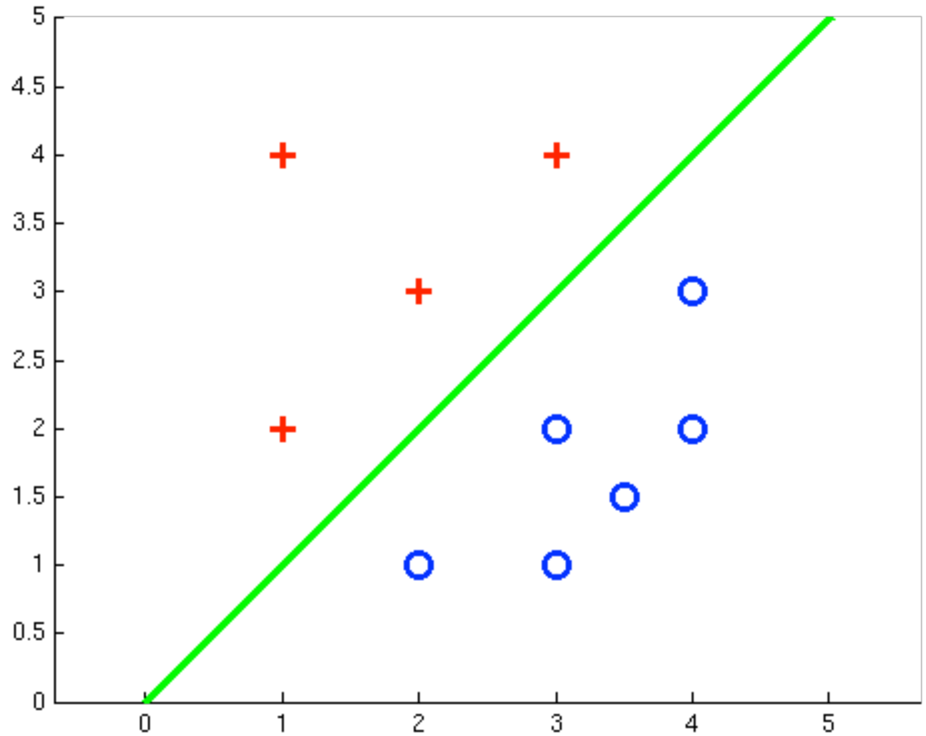
$$\forall i, \mathbf{y}, \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) + \xi_i \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$$

  - C is called the *capacity* of the SVM – the smoothing knob



- Learning:
  - Can still stick this into Matlab if you want
  - Constrained optimization is hard; better methods!
  - We'll come back to this later

Maximum Margin



Likelihood

Linear Models: Maximum Entropy

- Maximum entropy (logistic regression)
  - Use the scores as probabilities:

$$P(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y}))}{\sum_{\mathbf{y}'} \exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y}'))}$$

Make positive

Normalize

  - Maximize the (log) conditional likelihood of training data

$$L(\mathbf{w}) = \log \prod_i P(\mathbf{y}_i^*|\mathbf{x}_i, \mathbf{w}) = \sum_i \log \left(\frac{\exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*))}{\sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}))} \right)$$

$$= \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right)$$

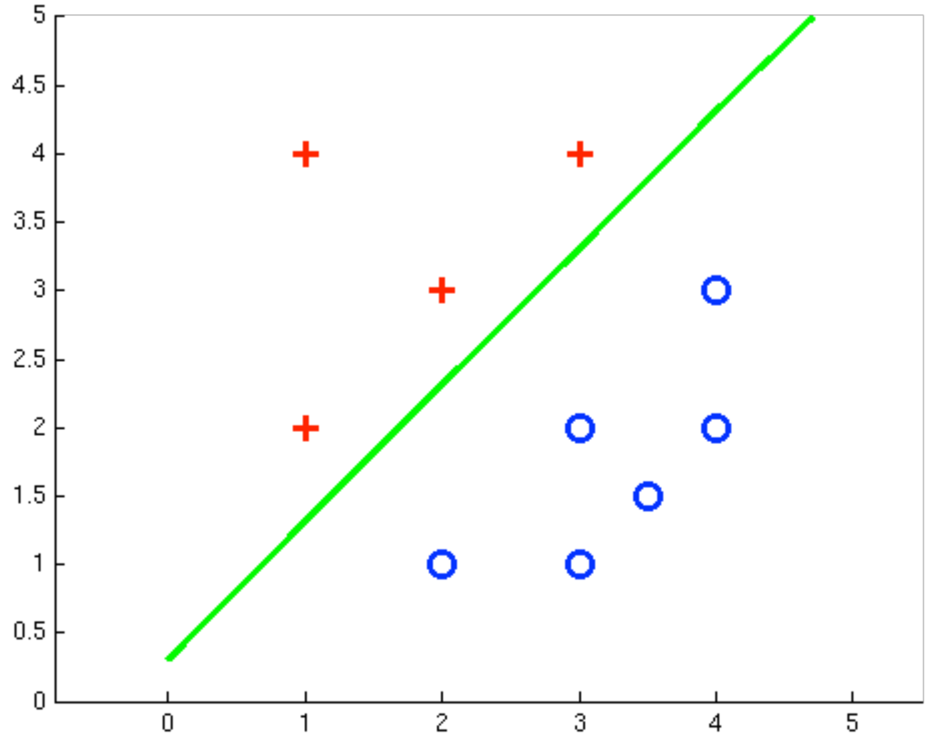
Maximum Entropy II

- Motivation for maximum entropy:
  - Connection to maximum entropy principle (sort of)
  - Might want to do a good job of being uncertain on noisy cases…
  - … in practice, though, posteriors are pretty peaked
- Regularization (smoothing)

$$\max_{\mathbf{w}} \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right) - k||\mathbf{w}||^2$$

$$\min_{\mathbf{w}} \; k||\mathbf{w}||^2 - \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right)$$

Maximum Entropy



Loss Comparison

Log-Loss

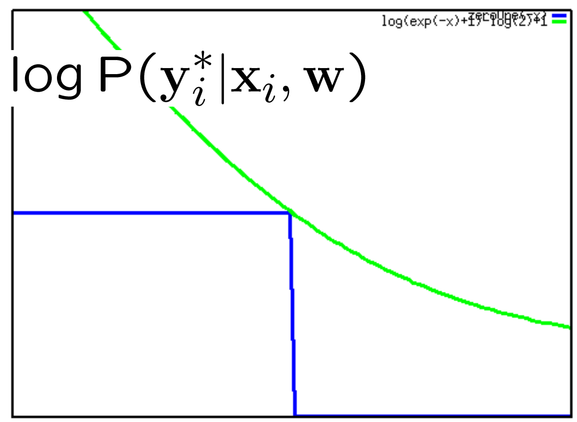
- If we view maxent as a minimization problem:

$$\min_{\mathbf{w}} \quad k||\mathbf{w}||^2 + \sum_i - \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right)$$

- This minimizes the "log loss" on each example

$$- \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right) = - \log \mathsf{P}(\mathbf{y}_i^* | \mathbf{x}_i, \mathbf{w})$$

$$\mathit{step} \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \max_{\mathbf{y} \neq \mathbf{y}_i^*} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) \right)$$



- One view: log loss is an *upper bound* on zero-one loss

Remember SVMs…

- We had a constrained minimization

$$\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i$$

$$\forall i, \mathbf{y}, \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) + \xi_i \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$$

- …but we can solve for ξ_i

$$\forall i, \mathbf{y}, \quad \xi_i \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y}) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*)$$

$$\forall i, \quad \xi_i = \max_{\mathbf{y}} \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})\right) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*)$$

- Giving

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \left(\max_{\mathbf{y}} \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})\right) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*)\right)$$

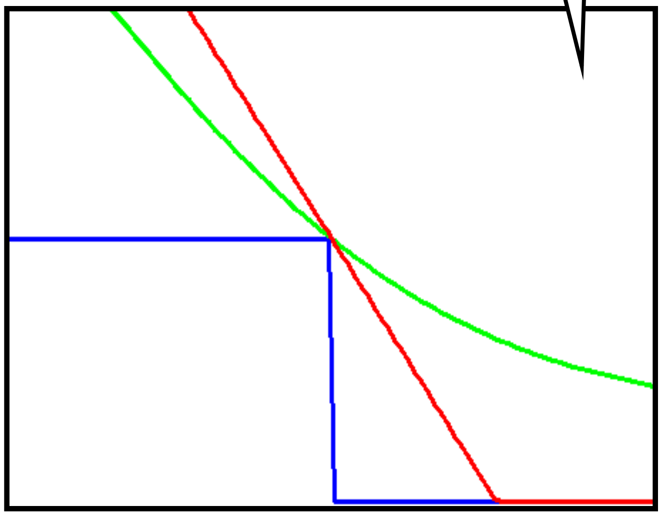
Hinge Loss

Plot really only right in binary case

- Consider the per-instance objective:

$$\min_{\mathbf{w}} \quad k||\mathbf{w}||^2 + \sum_i \left(\max_{\mathbf{y}} \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(y) \right) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) \right)$$

- This is called the "hinge loss"
  - Unlike maxent / log loss, you stop gaining objective once the true label wins by enough
  - You can start from here and derive the SVM objective
  - Can solve directly with sub-gradient decent (e.g. Pegasos: Shalev-Shwartz et al 07)



$$\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \max_{\mathbf{y} \neq \mathbf{y}_i^*} \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) \right)$$

Max vs "Soft-Max" Margin

- SVMs:

$$\min_{\mathbf{w}} \quad k||\mathbf{w}||^2 - \sum_i \left(\underbrace{\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \max_{\mathbf{y}} \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(y) \right)}_{\text{You can make this zero}} \right)$$

- Maxent:

$$\min_{\mathbf{w}} \quad k||\mathbf{w}||^2 - \sum_i \left(\underbrace{\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \log \sum_{\mathbf{y}} \exp \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) \right)}_{\text{… but not this one}} \right)$$

- Very similar! Both try to make the true score better than a function of the other scores
  - The SVM tries to beat the augmented runner-up
  - The Maxent classifier tries to beat the "soft-max"

Loss Functions: Comparison

- Zero-One Loss

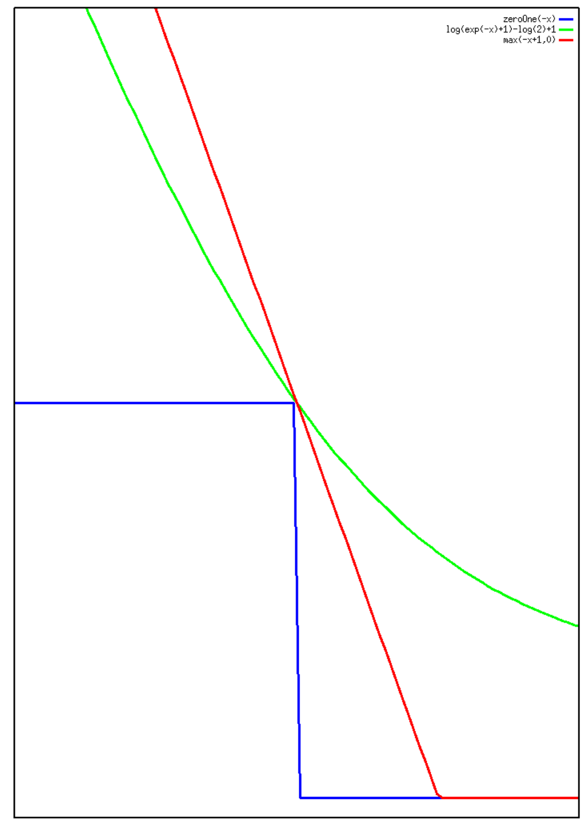
$$\sum_i step\left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \max_{\mathbf{y} \neq \mathbf{y}_i^*} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})\right)$$

- Hinge

$$\sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \max_{\mathbf{y}} \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(y)\right)\right)$$

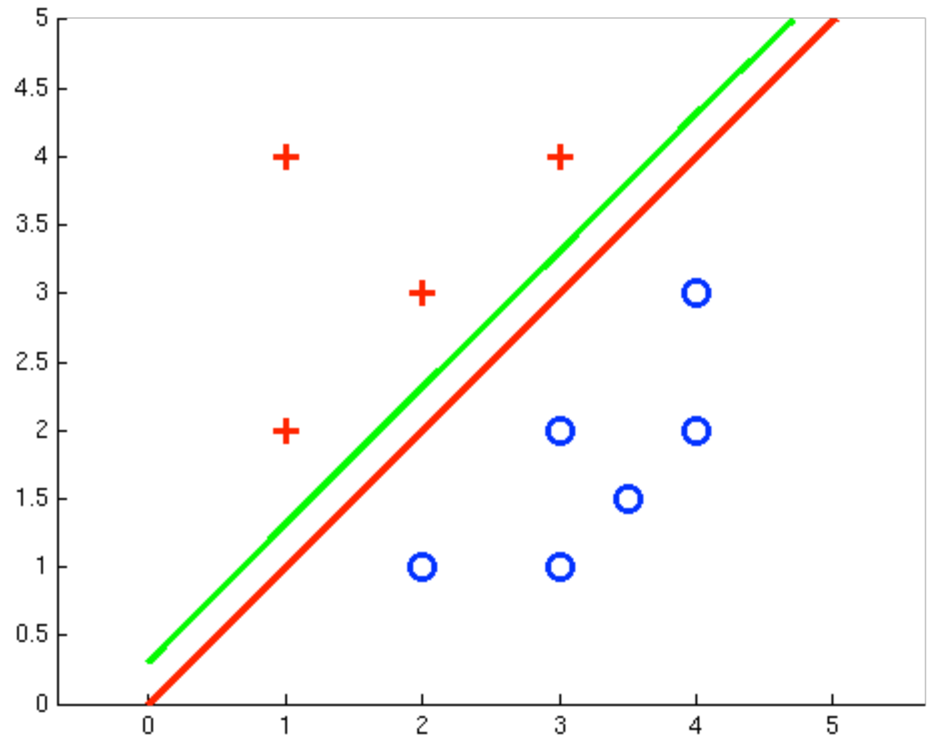
- Log

$$\sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \log \sum_{\mathbf{y}} \exp\left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})\right)\right)$$



$$\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \max_{\mathbf{y} \neq \mathbf{y}_i^*} \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})\right)$$

Separators: Comparison



Conditional vs Joint Likelihood

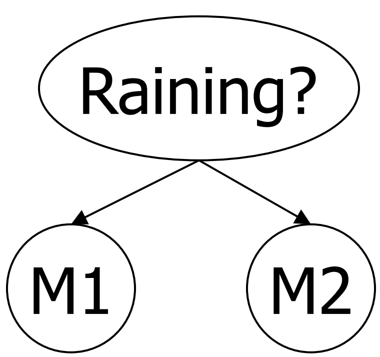
Example: Sensors

Reality

Raining | Sunny

P(+,+,r) = 3/8   P(-,-,r) = 1/8   P(+,+,s) = 1/8   P(-,-,s) = 3/8

NB Model



NB FACTORS:

- P(s) = 1/2
- P(+|s) = 1/4
- P(+|r) = 3/4

PREDICTIONS:

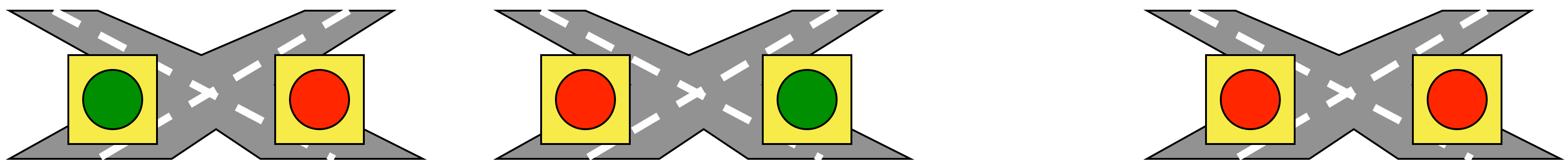
- P(r,+,+) = (½)(¾)(¾)
- P(s,+,+) = (½)(¼)(¼)
- P(r|+,+) = 9/10
- P(s|+,+) = 1/10

Example: Stoplights

Reality

Lights Working

Lights Broken

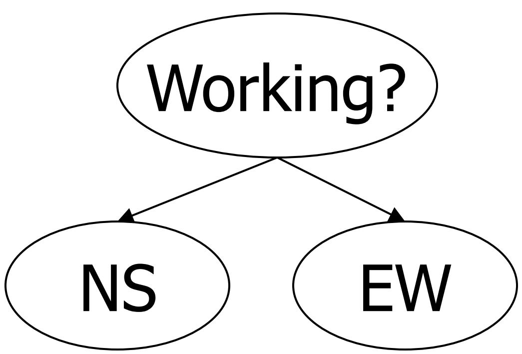


$P(g,r,w) = 3/7$

$P(r,g,w) = 3/7$

$P(r,r,b) = 1/7$

NB Model



NB FACTORS:

- $P(w) = 6/7$
- $P(r|w) = 1/2$
- $P(g|w) = 1/2$

- $P(b) = 1/7$
- $P(r|b) = 1$
- $P(g|b) = 0$

Example: Stoplights

- What does the model say when both lights are red?
  - P(b,r,r) = (1/7)(1)(1) = 1/7 = 4/28
  - P(w,r,r) = (6/7)(1/2)(1/2) = 6/28 = 6/28
  - P(w|r,r) = 6/10!
- We'll guess that (r,r) indicates lights are working!
- Imagine if P(b) were boosted higher, to 1/2:
  - P(b,r,r) = (1/2)(1)(1) = 1/2 = 4/8
  - P(w,r,r) = (1/2)(1/2)(1/2) = 1/8 = 1/8
  - P(w|r,r) = 1/5!
- Changing the parameters bought accuracy at the expense of data likelihood