Algorithms for NLP

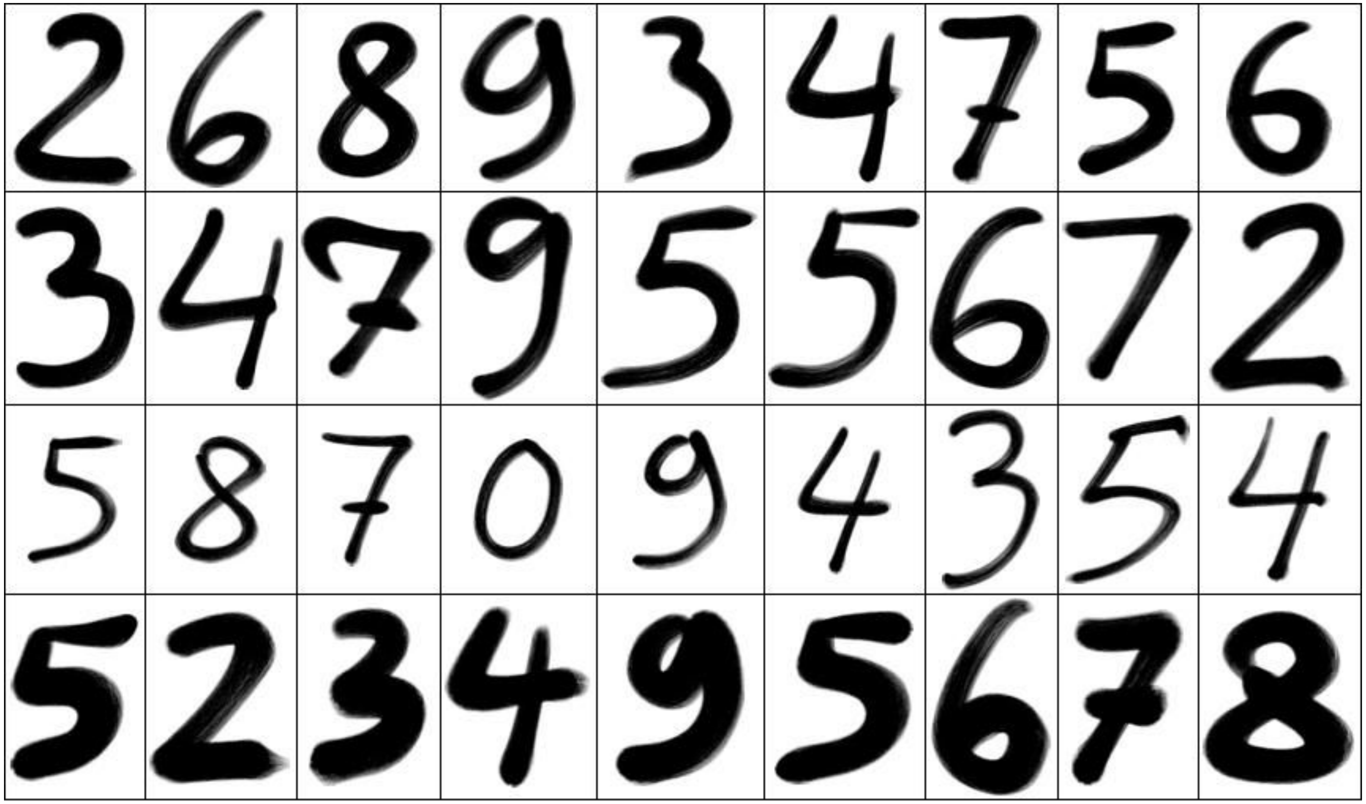
Classification I

Sachin Kumar - CMU

Slides: Dan Klein – UC Berkeley, Taylor Berg-Kirkpatrick, Yulia Tsvetkov – CMU

Classification

Image → Digit



Classification

Document → Category

Classification

Query + Web Pages → Best Match

“Apple Computers”

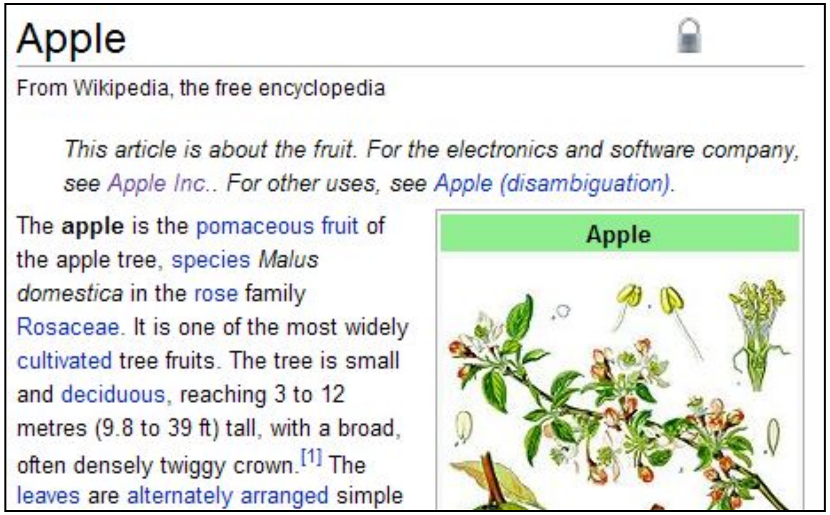
Apple Inc.

From Wikipedia, the free encyclopedia

(Redirected from Apple Computer)

Apple Inc.,

Apple Inc.



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).

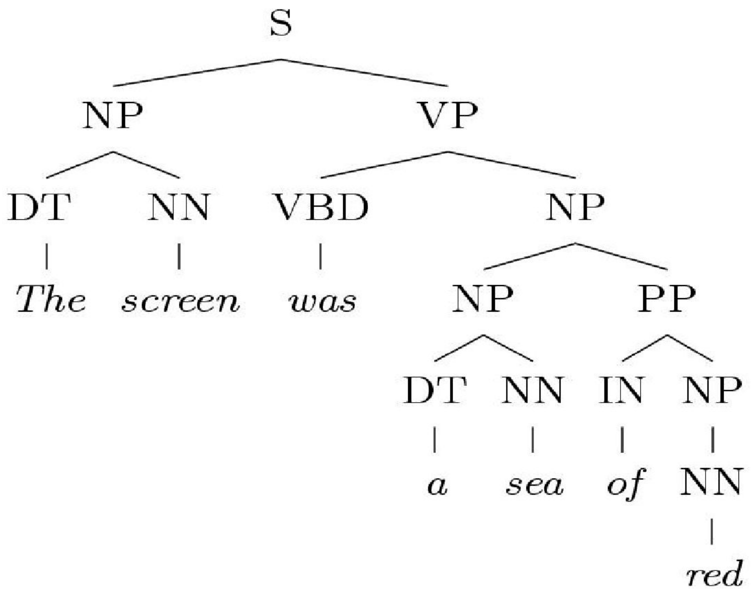
The **apple** is the pomaceous fruit of the apple tree, species *Malus domestica* in the rose family Rosaceae. It is one of the most widely cultivated tree fruits. The tree is small and deciduous, reaching 3 to 12 metres (9.8 to 39 ft) tall, with a broad, often densely twiggy crown.[1] The leaves are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

Classification

Sentence → Parse Tree

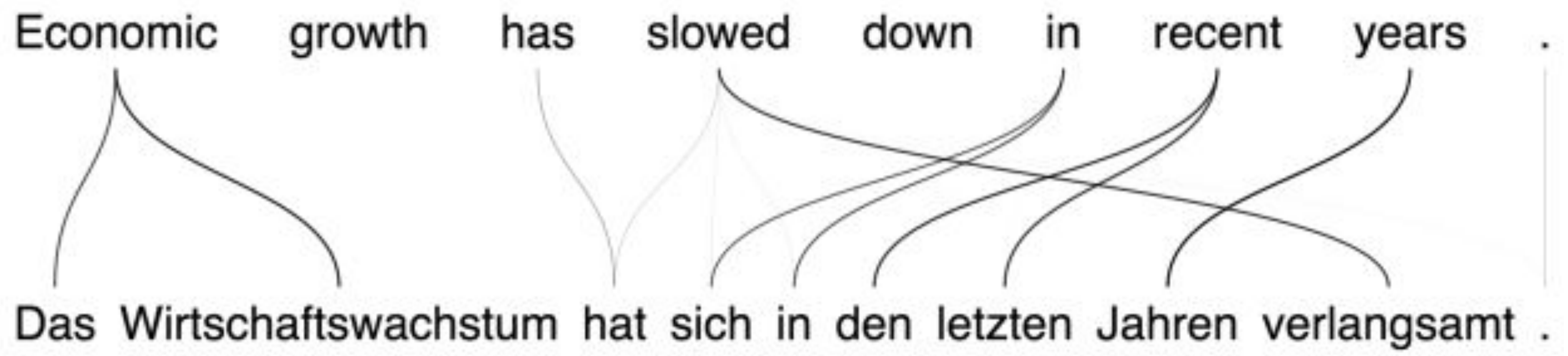
x **y**

The screen was a sea of red



Classification

Sentence ⟶ Translation

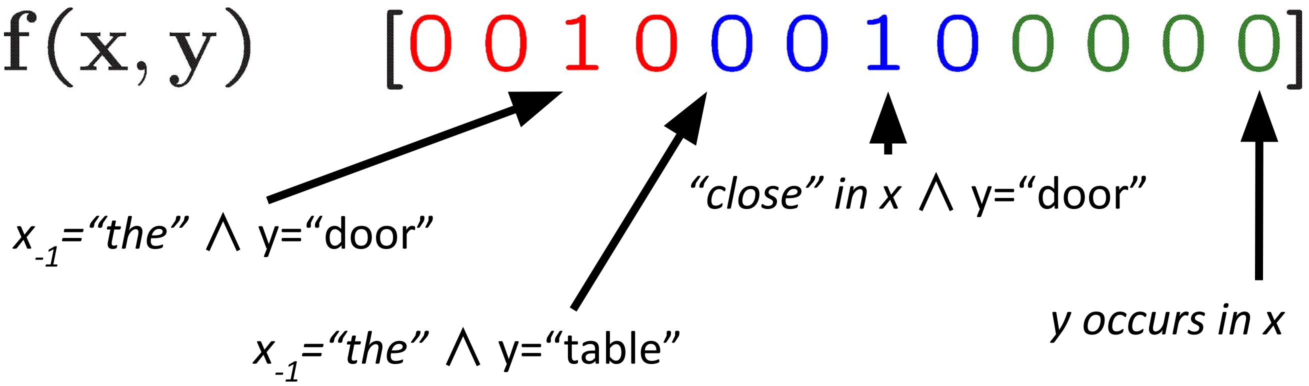


Classification

- Three main ideas
    - Representation as feature vectors
    - Scoring by linear functions
    - Learning (the scoring functions) by optimization

Some Definitions

INPUTS	$\mathbf{x}_i$	*close the ____*
CANDIDATE SET	$\mathcal{Y}(\mathbf{x})$	*{table, door, … }*
CANDIDATE	$\mathbf{y}$	*table*
TRUE OUTPUT	$\mathbf{y}_i^*$	*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

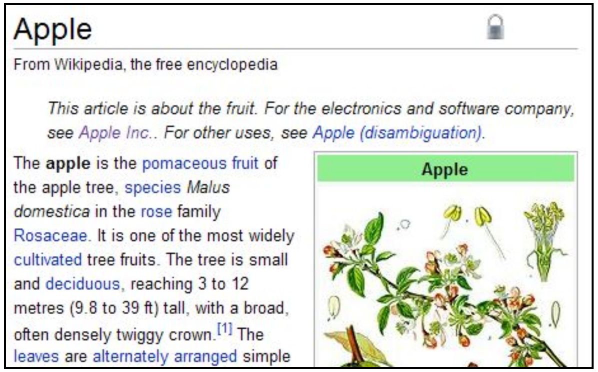
Features

Feature Vectors

- Example: web page ranking (not actually classification)

x_i = “Apple Computers”

$\mathbf{f}_i($



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

$) = [0.3\ 5\ 0\ 0\ \ldots]$

$\mathbf{f}_i($

Apple Inc.

From Wikipedia, the free encyclopedia

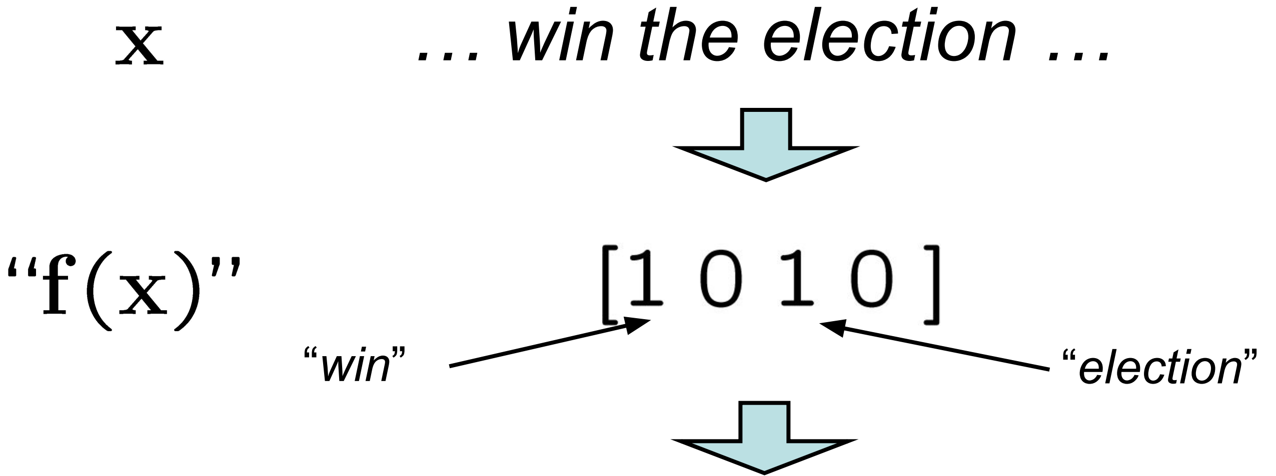
(Redirected from Apple Computer)

Apple Inc.,

$) = [0.8\ 4\ 2\ 1\ \ldots]$

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 …

$$\mathbf{f}(SPORTS) = [1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$$

… win the election …

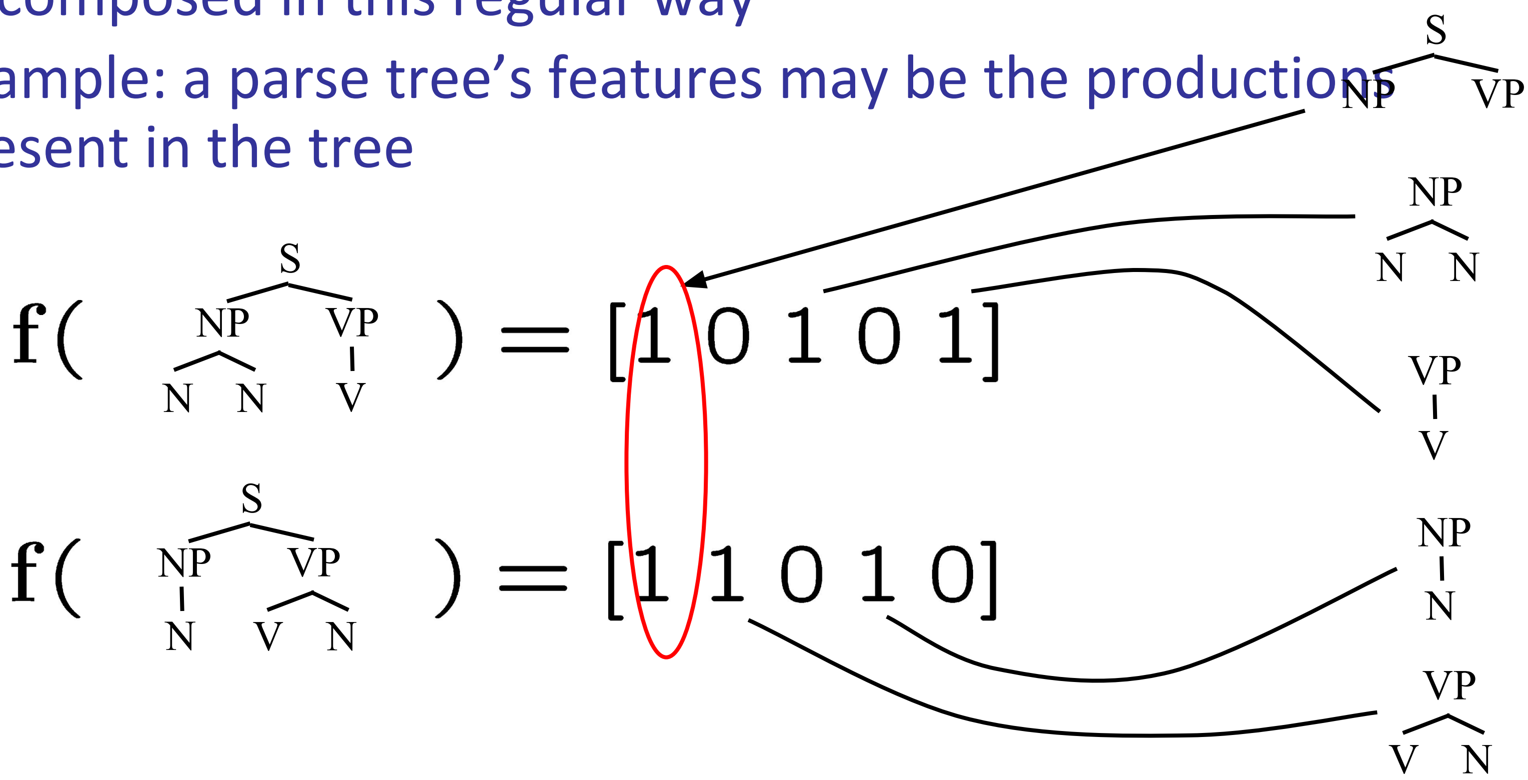
$$\mathbf{f}(POLITICS) = [0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0]$$

… win the election …

$$\mathbf{f}(OTHER) = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0]$$

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

… win the election …

$$\mathbf{f}(POLITICS) = [\ 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$$

… win the election …

$$\mathbf{f}(SPORTS) = [\ 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$$

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

- We score hypotheses by multiplying features and weights:

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

… win the election …

$$\mathbf{f}(POLITICS) = [\ 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$$

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

… win the election …

$$score(POLITICS, \mathbf{w}) = 1 \times 1 + 1 \times 1 = 2$$

Linear Models: Decision Rule

- The linear decision rule:

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

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

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

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

⇓

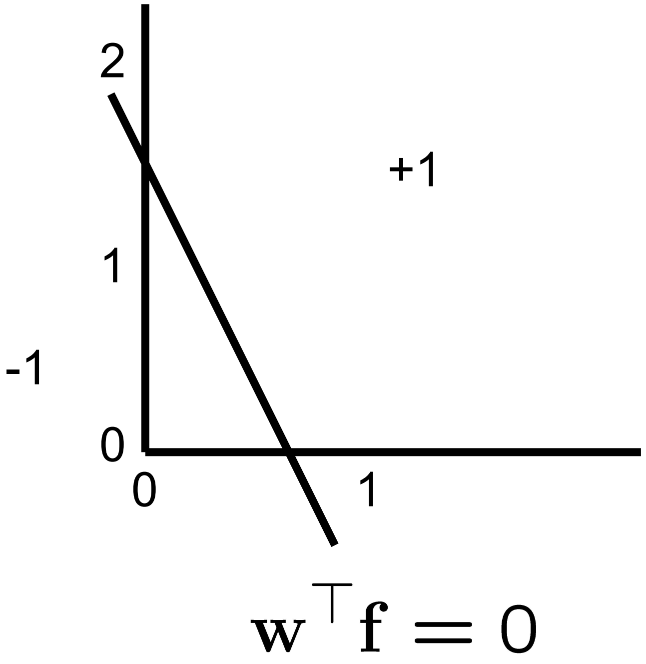
$$prediction(\text{... win the election ...}, \mathbf{w}) = \overset{\text{... win the election ...}}{POLITICS}$$

- We’ve said nothing about where weights come from

Binary Classification

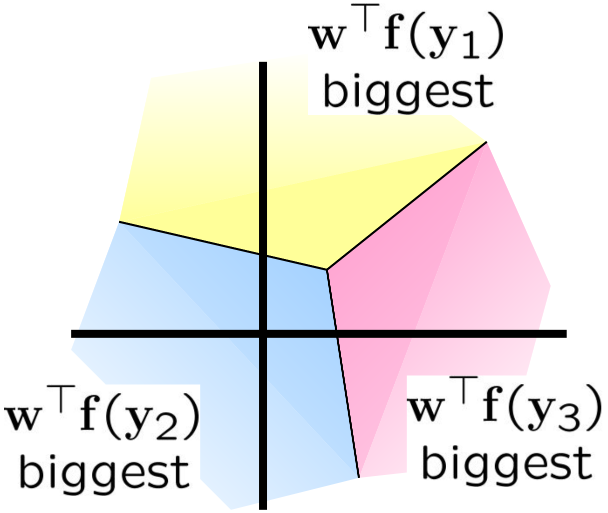
- Important special case: binary classification
  - Classes are y=+1/-1
  - Decision boundary is a hyperplane

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



Multiclass Decision Rule

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



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

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?

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

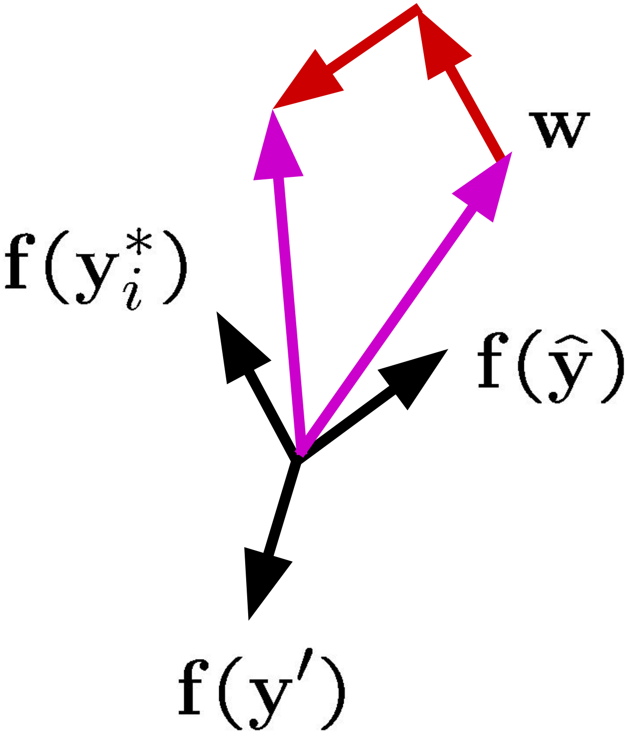
Linear Models: Perceptron

- 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”

$\mathbf{f}_i($

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

$) = [0.3\ 5\ 0\ 0\ \ldots]$

$$\mathbf{w}^\top \mathbf{f} = 10.3 \qquad \hat{\mathbf{y}}$$

$\mathbf{f}_i($

Apple Inc.

From Wikipedia, the free encyclopedia

(Redirected from Apple Computer)

Apple Inc.,

Apple Inc.

$) = [0.8\ 4\ 2\ 1\ \ldots]$

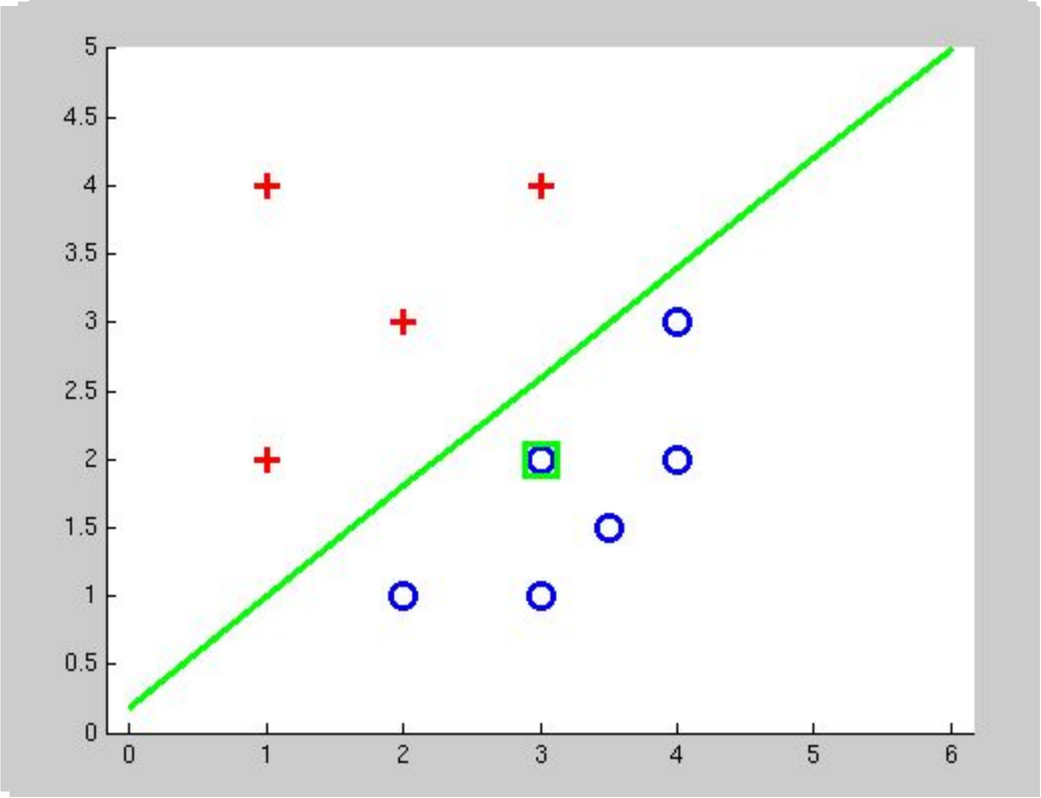
$$\mathbf{w}^\top \mathbf{f} = 8.8 \qquad \mathbf{y}_i^*$$

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

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

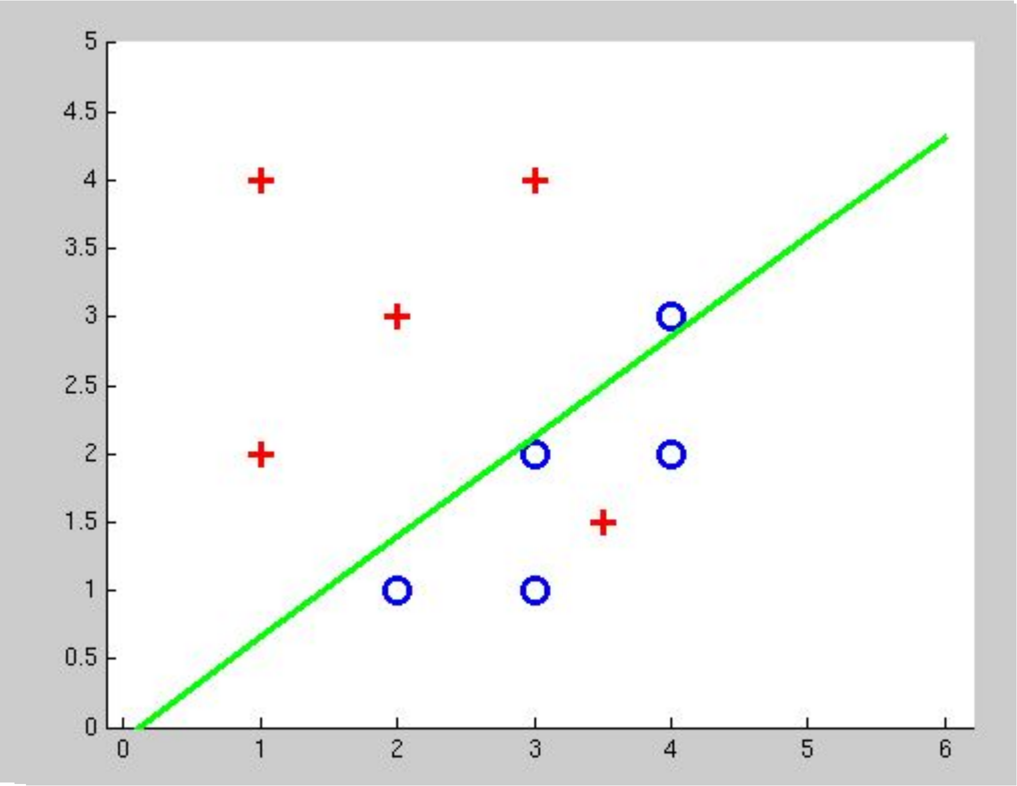
Examples: Perceptron

- Separable Case



Examples: Perceptron

- Non-Separable Case

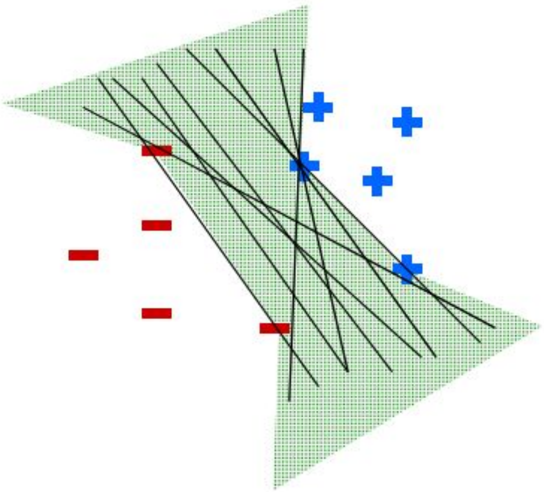


Problems with Perceptron

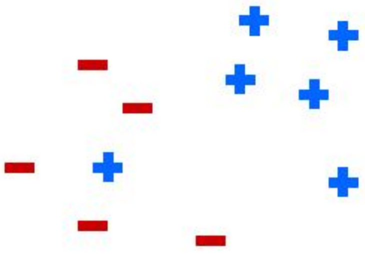
- Perceptron “Goal”: Seperate the training data

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

1. This may be an entire feasible space



2. Or it may be impossible



Objective Functions

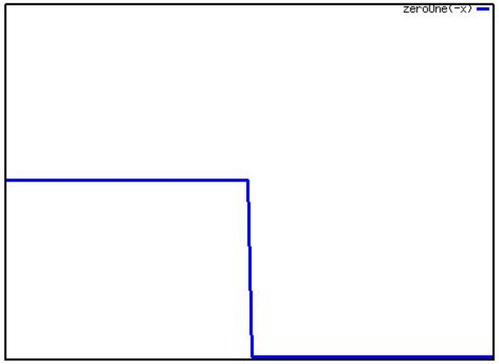
- What do we want from our weights?
  - So far: minimize (training) errors:

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

or

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

  - This is the “zero-one loss”
    - Discontinuous, minimizing is NP-complete
  - Maximum entropy and SVMs have other objectives related to zero-one loss

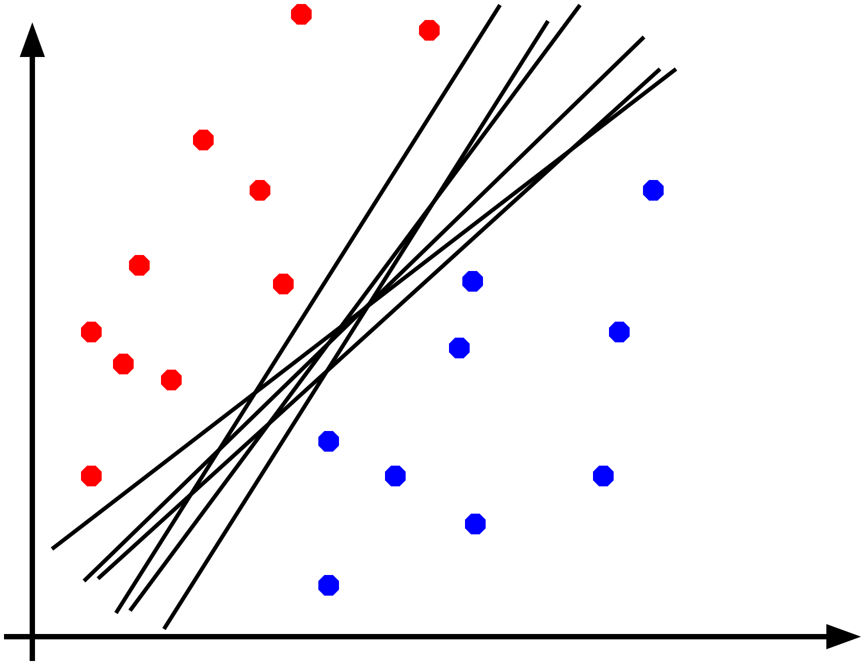


$$\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y} \neq \mathbf{y}_i^*} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$$

Margin

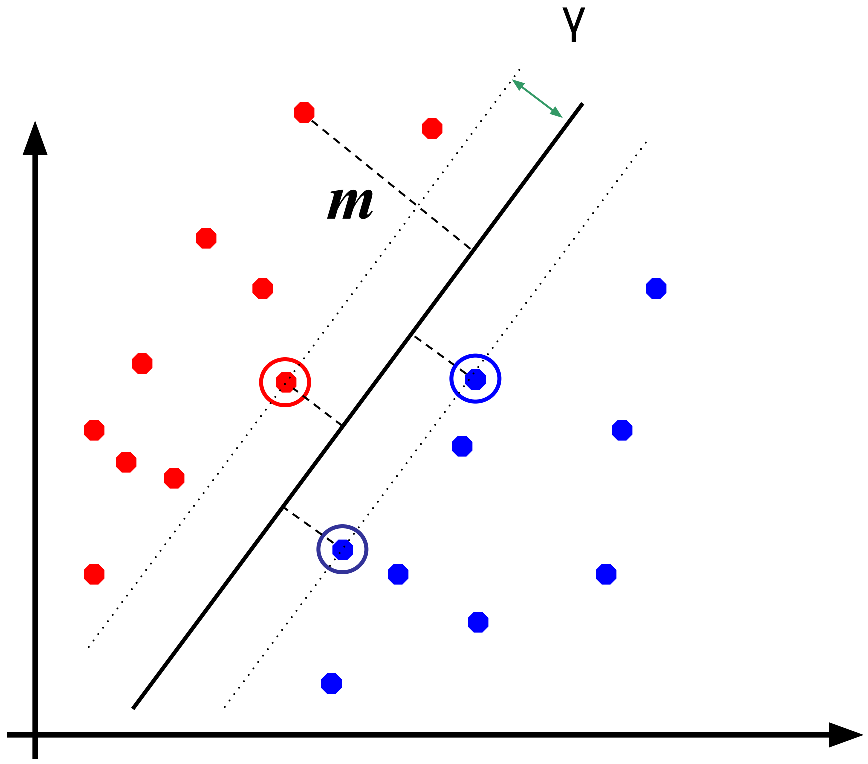
Linear Separators

- 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

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

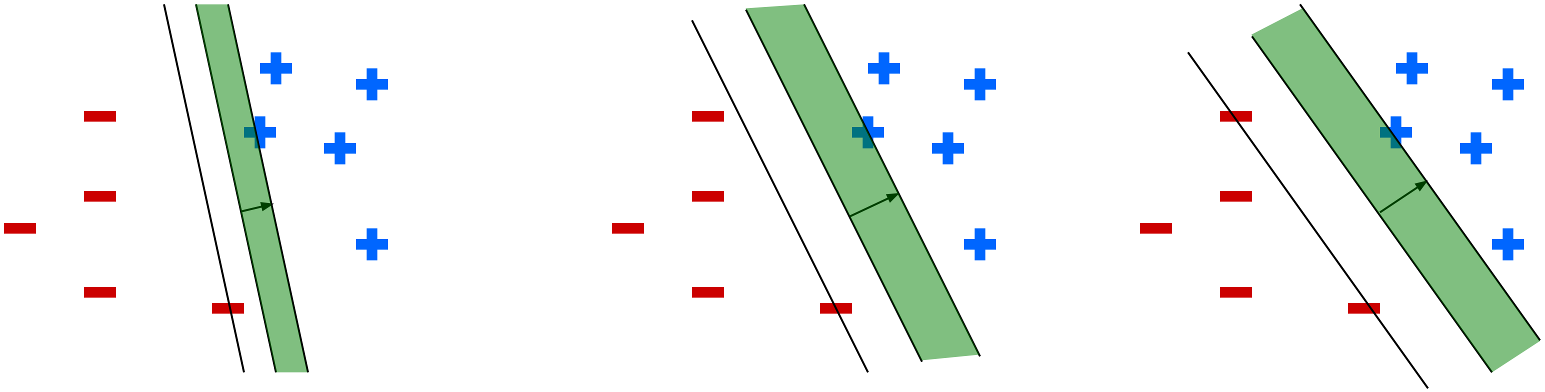
Maximum Margin

- Separable SVMs: find the max-margin w

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

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

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



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

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

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

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

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

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

Gamma to w

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

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

$$\mathbf{w} = \gamma u$$

$$\gamma = 1/||u||$$

$$\max_{||\gamma u||=1} 1/||u||^2$$

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

$$\max_{||\gamma u||=1} 1/||u||^2$$

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

$$\min_{||\gamma u||=1} ||u||^2$$

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

$$\min_u ||u||^2$$

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

$$\min_u \frac{1}{2}||u||^2$$

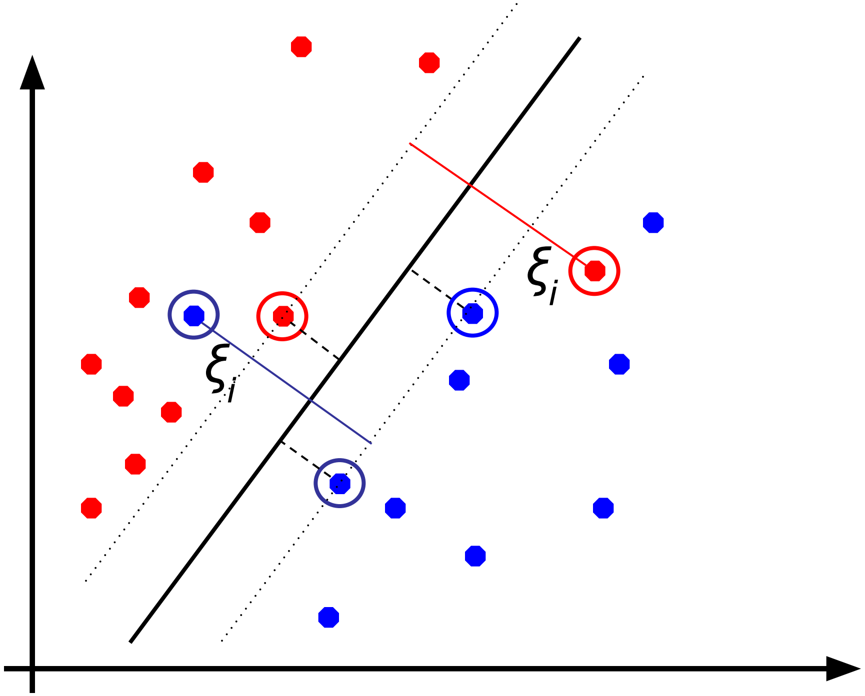
$$\forall i, \mathbf{y} \quad u^\top \mathbf{f}_i(\mathbf{y}_i^*) \geq u^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$$

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

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

Soft Margin Classification

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



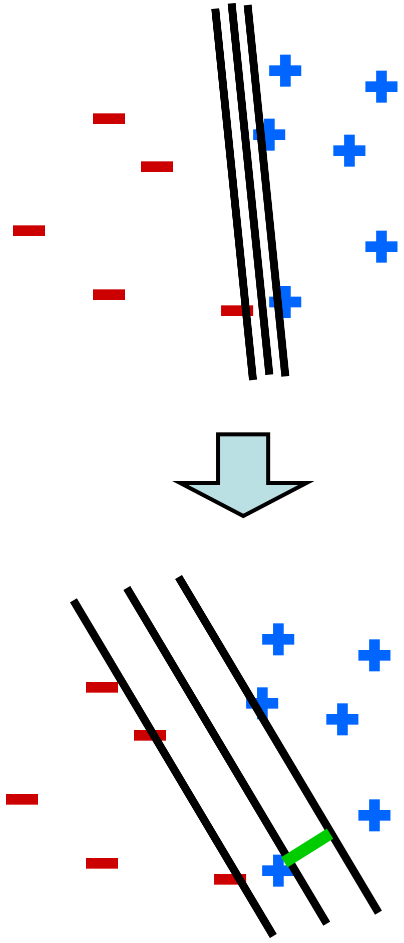
Maximum Margin

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!

Hinge Loss

- We have 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

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

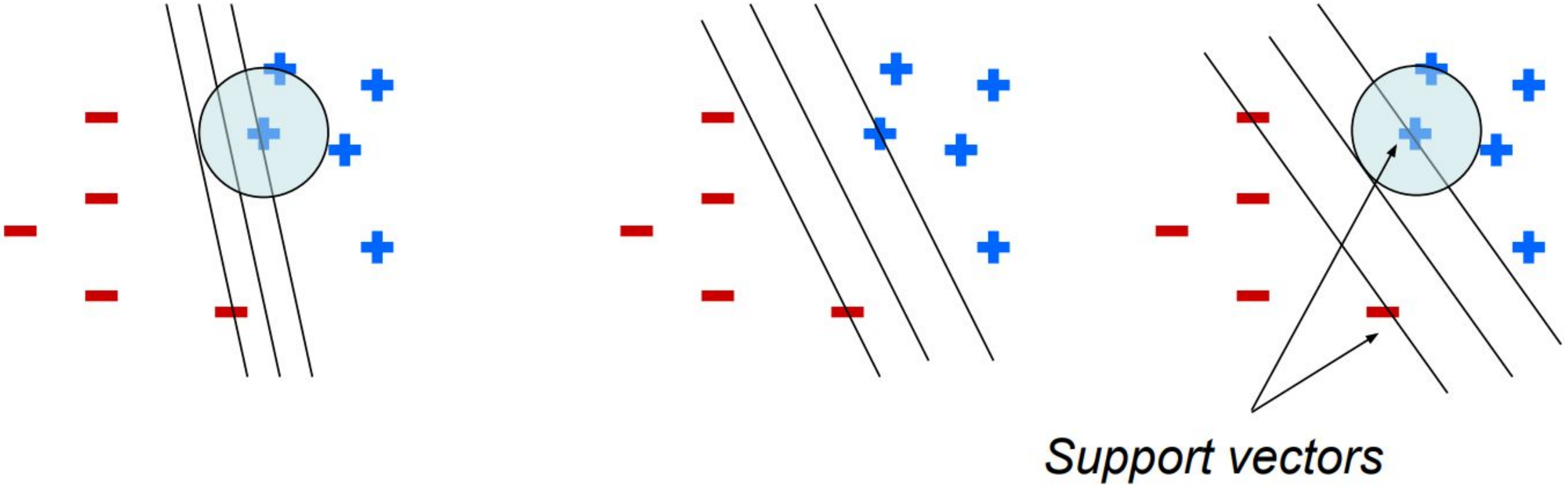
$$\forall i, \quad \xi_i = \max_{\mathbf{y}} \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})\right) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*)$$

- Giving

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

Why Max Margin?

- Why do this? Various arguments:
  - Solution depends only on the boundary cases, or *support vectors*
  - 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



Likelihood

Linear Models: Maximum Entropy

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

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

Make positive

Normalize

  - Maximize the (log) conditional likelihood of training data

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

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

Maximum Entropy II

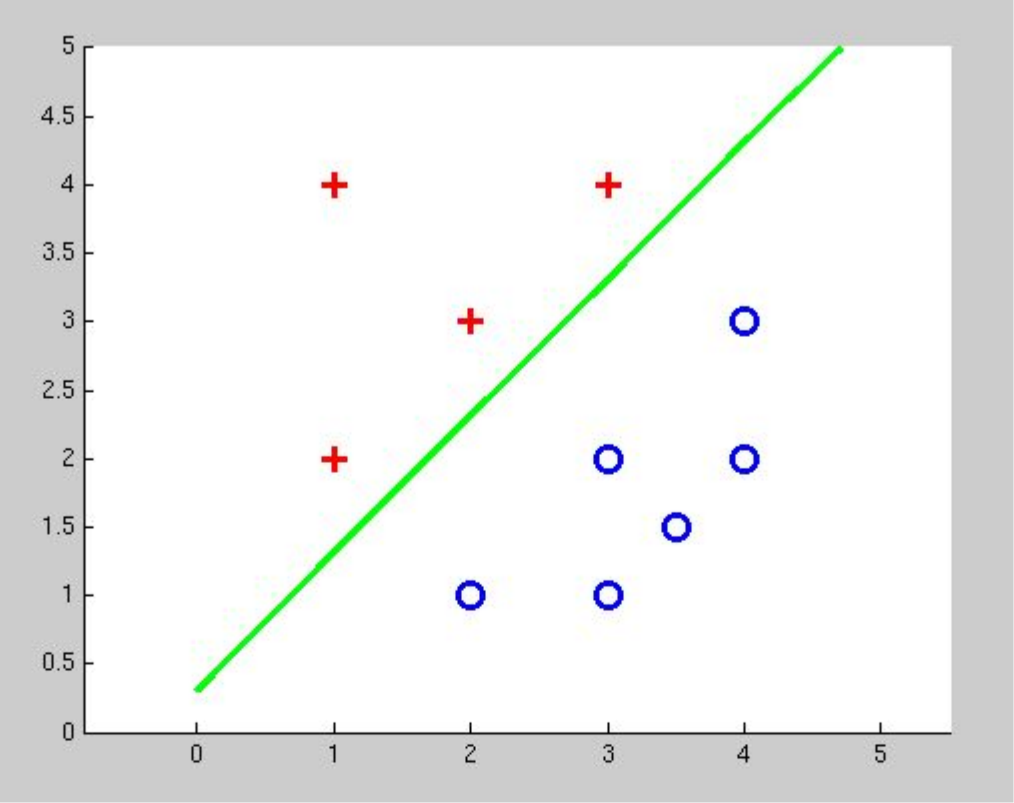
- Motivation for maximum entropy:
  - Connection to maximum entropy principle (sort of)
  - Might want to do a good job of being uncertain on noisy cases…
  - … in practice, though, posteriors are pretty peaked

- 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:

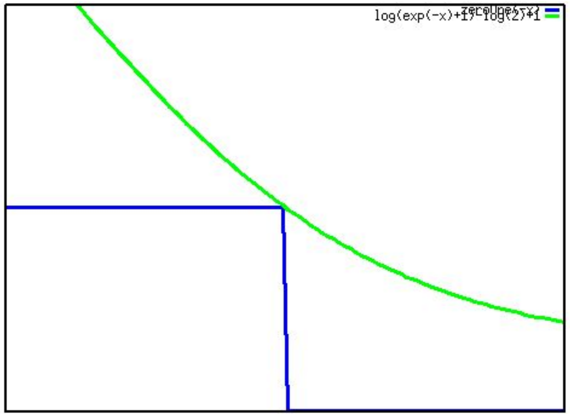
$$\min_{\mathbf{w}} \; k||\mathbf{w}||^2 + \sum_i - \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right)$$

- This minimizes the "log loss" on each example

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

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

- Or *d* on z



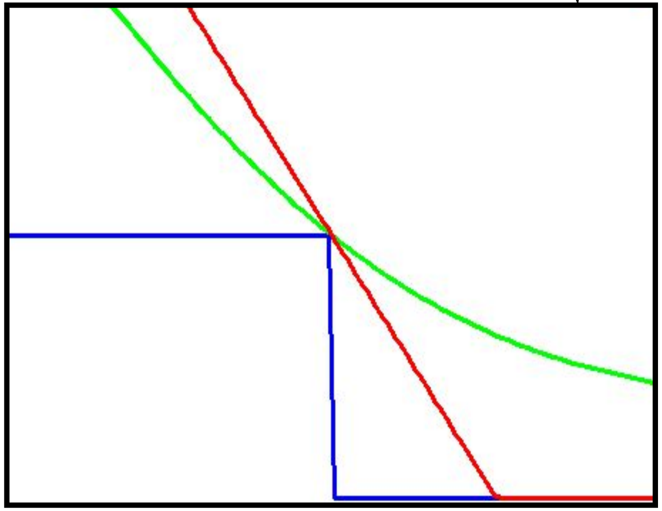
Remember SVMs - Hinge Loss

- Consider the per-instance objective:

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

Plot really only right in binary case

- 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:

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

- Ver, 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

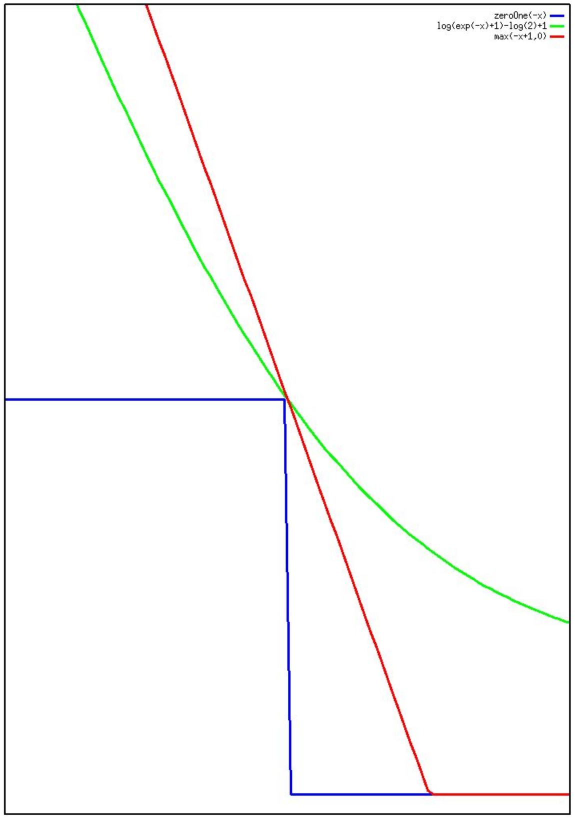
$$\sum_i \mathit{step}\left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \max_{\mathbf{y} \neq \mathbf{y}_i^*} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})\right)$$

- Hinge

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

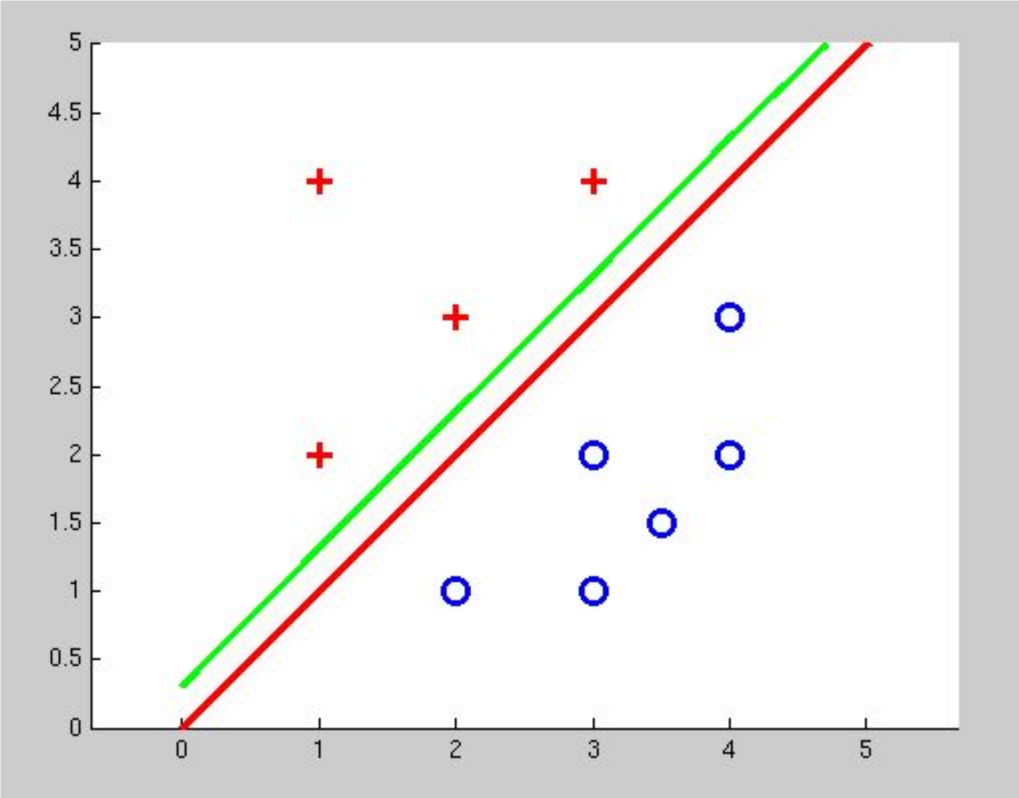
- Log

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



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

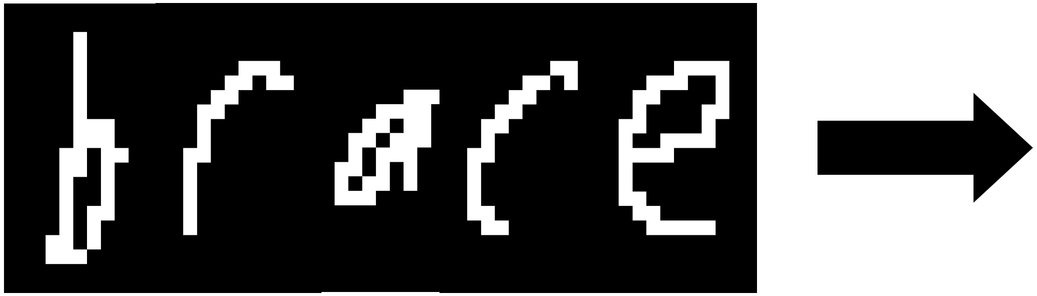
Separators: Comparison



Structure

Handwriting recognition

x **y**



brace

Sequential structure

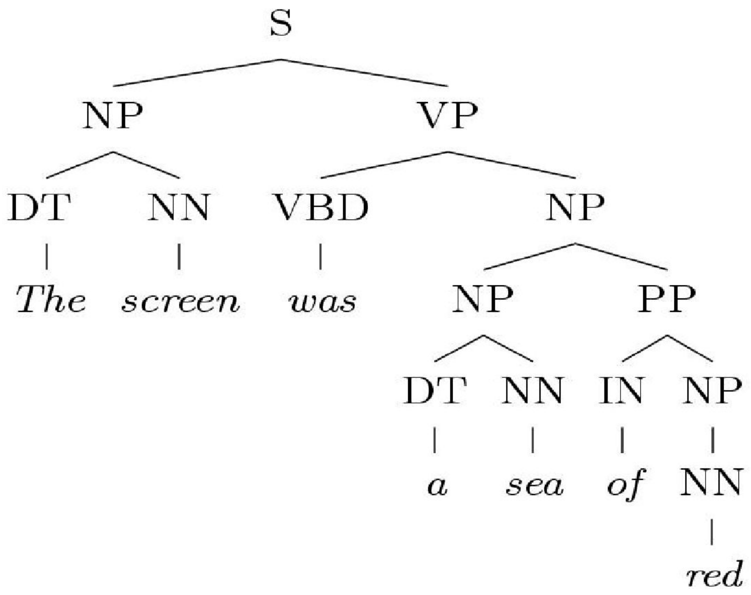
[Slides: Taskar and Klein 05]

CFG Parsing

x

y

The screen was a sea of red



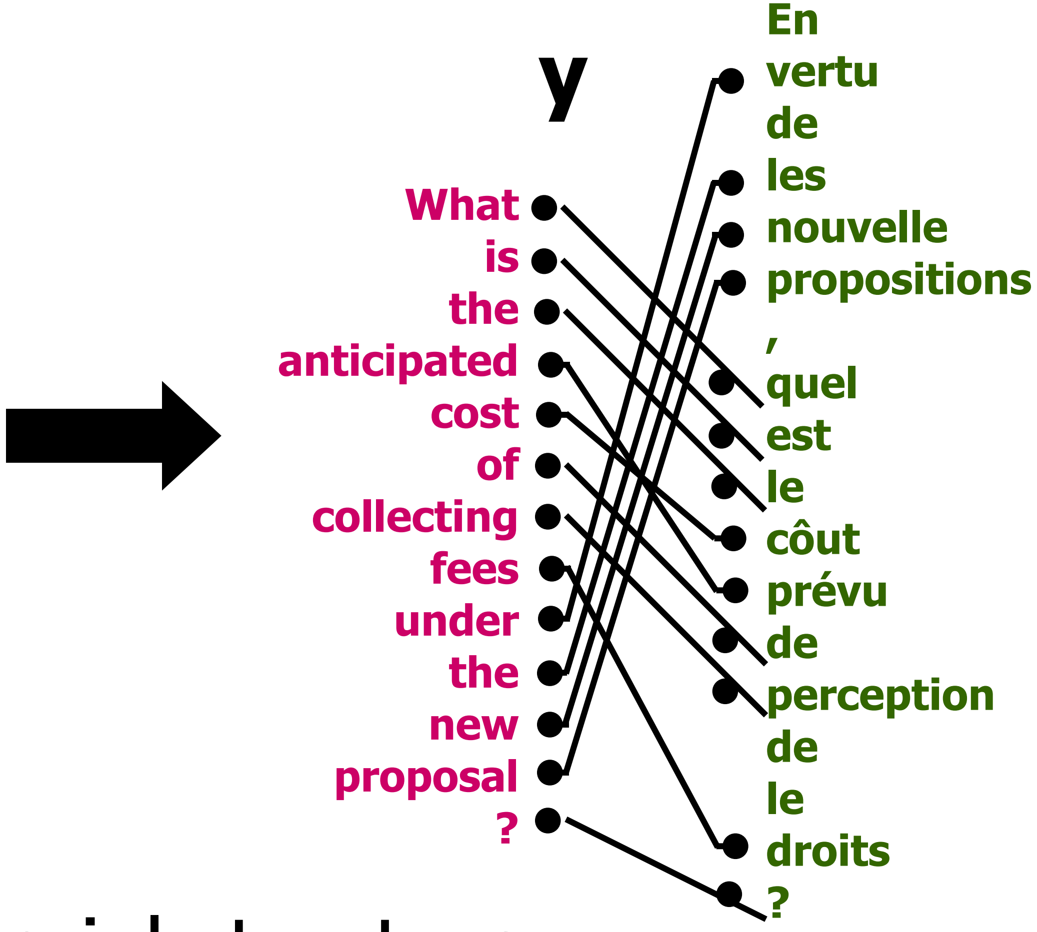
Recursive structure

Bilingual Word Alignment

x

What is the anticipated cost of collecting fees under the new proposal?

En vertu de nouvelle propositions, quel est le côut prévu de perception de les droits?



y

What
is
the
anticipated
cost
of
collecting
fees
under
the
new
proposal
?

En
vertu
de
les
nouvelle
propositions
,
quel
est
le
côut
prévu
de
perception
de
le
droits
?

Combinatorial structure

Definitions

INPUTS	$\mathbf{x}_i$
CANDIDATE SET	$\mathcal{Y}(\mathbf{x})$
CANDIDATES	$\mathbf{y}$
TRUE OUTPUTS	$\mathbf{y}_i^*$
FEATURE VECTORS	$\mathbf{f}(\mathbf{x}, \mathbf{y})$

Structured Models

$$\mathit{prediction}(\mathbf{x},\mathbf{w}) = \operatorname*{arg\,max}_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \mathit{score}(\mathbf{y},\mathbf{w})$$

space of feasible outputs

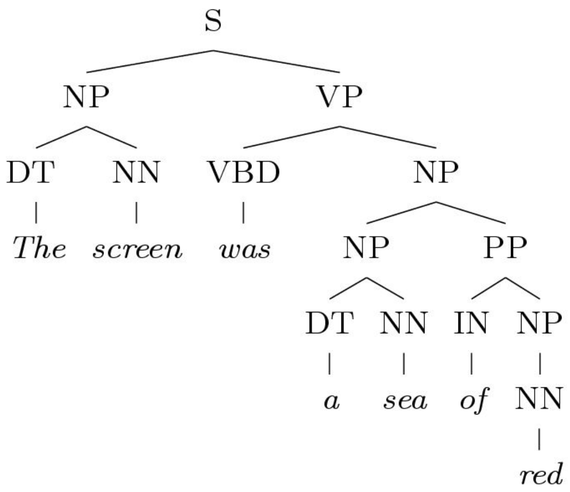
Assumption:

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

Score is a sum of local “part” scores

Parts = nodes, edges, productions

CFG Parsing



$$\mathbf{f} : \mathcal{X} \times \mathcal{Y} \rightarrow \Re^d$$

#(NP → DT NN)

…

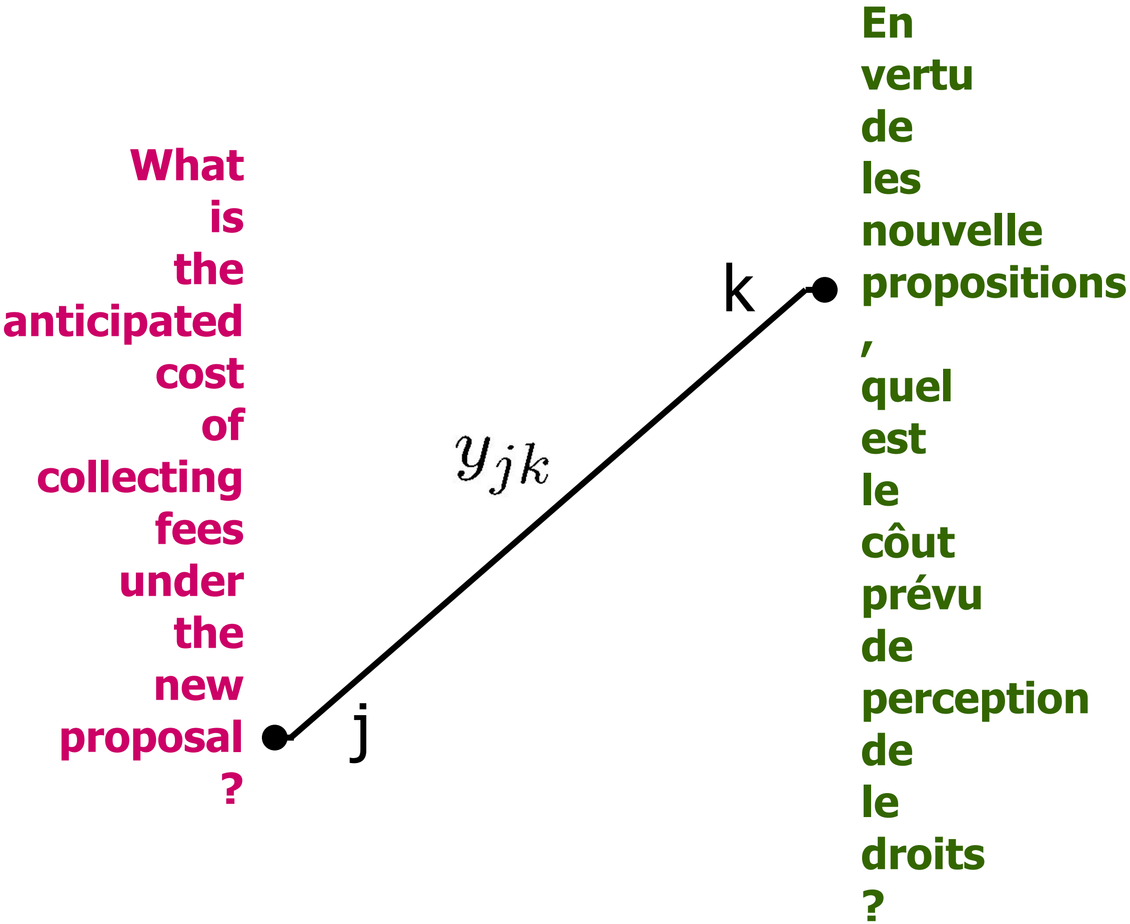
#(PP → IN NP)

…

#(NN → ‘sea’)

Bilingual word alignment

$$\sum_{y_{jk} \in \mathbf{y}} \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk}) = \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})$$



What
is
the
anticipated
cost
of
collecting
fees
under
the
new
proposal
?

En
vertu
de
les
nouvelle
propositions
,
quel
est
le
côut
prévu
de
perception
de
le
droits
?

$\mathbf{f}(\mathbf{x}_{jk})$

- association
- position
- orthography

Efficient Decoding

- Common case: you have a black box which computes

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

at least approximately, and you want to learn w

- Easiest option is the structured perceptron [Collins 01]
  - Structure enters here in that the search for the best y is typically a combinatorial algorithm (dynamic programming, matchings, ILPs, A*…)
  - Prediction is structured, learning update is not

Structured Margin (Primal)

Remember our primal margin objective?

$$\min_{w} \quad \frac{1}{2}\|w\|_2^2 + C\sum_i \left(\max_y \left(w^\top f_i(y) + \ell_i(y)\right) - w^\top f_i(y_i^*)\right)$$

Still applies with structured output space!

Structured Margin (Primal)

Just need efficient loss-augmented decode:

$$\bar{y} = \operatorname{argmax}_y \left(w^\top f_i(y) + \ell_i(y) \right)$$

$$\min_w \quad \frac{1}{2}\|w\|_2^2 + C \sum_i \left(w^\top f_i(\bar{y}) + \ell_i(\bar{y}) - w^\top f_i(y_i^*) \right)$$

$$\nabla_w = w + C \sum_i \left(f_i(\bar{y}) - f_i(y_i^*) \right)$$

Still use general subgradient descent methods! (Adagrad)

Structured Margin

- Remember the constrained version of primal:

$$\begin{array}{ll} \min_{\mathbf{w},\xi} & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \\ \forall i, \mathbf{y} & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y}) - \xi_i \end{array}$$

Full Margin: OCR

- We want:

$\arg\max_{\mathbf{y}} \ \mathbf{w}^\top \mathbf{f}($  $, \mathbf{y}) \ = \ $ "brace"

- Equivalently:

$\mathbf{w}^\top \mathbf{f}($  $,$ "brace"$) \ > \ \mathbf{w}^\top \mathbf{f}($  $,$ "aaaaa"$)$

$\mathbf{w}^\top \mathbf{f}($  $,$ "brace"$) \ > \ \mathbf{w}^\top \mathbf{f}($  $,$ "aaaab"$)$

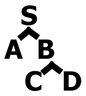
$\ldots$

$\mathbf{w}^\top \mathbf{f}($  $,$ "brace"$) \ > \ \mathbf{w}^\top \mathbf{f}($  $,$ "zzzzz"$)$

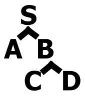
$\Bigg\}$ **a lot!**

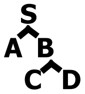
Parsing example

- We want:

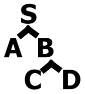
$\arg\max_{\mathrm{y}} \; \mathrm{w}^\top \mathrm{f}($ **'It was red'** $, \mathrm{y}) \; = $ 

- Equivalently:

$\mathrm{w}^\top \mathrm{f}($**'It was red',**  $) \; > \; \mathrm{w}^\top \mathrm{f}($**'It was red',**  $)$

$\mathrm{w}^\top \mathrm{f}($**'It was red',**  $) \; > \; \mathrm{w}^\top \mathrm{f}($**'It was red',**  $)$

…

$\mathrm{w}^\top \mathrm{f}($**'It was red',**  $) \; > \; \mathrm{w}^\top \mathrm{f}($**'It was red',**  $)$

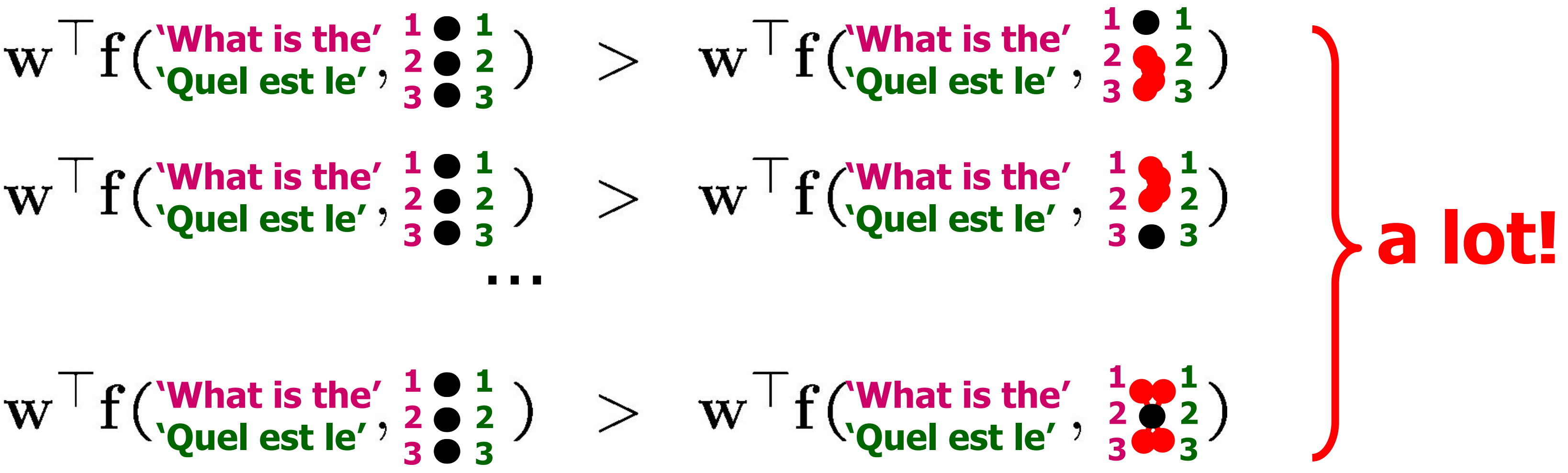
} **a lot!**

Alignment example

- We want:

$$\arg\max_y \; w^\top f\left(\begin{array}{c}\text{'What is the'}\\ \text{'Quel est le'}\end{array}, y\right) = \begin{array}{ccc}1 & \bullet & 1\\ 2 & \bullet & 2\\ 3 & \bullet & 3\end{array}$$

- Equivalently:



Cutting Plane

- A constraint induction method [Joachims et al 09]
  - Exploits that the number of constraints you actually need per instance is typically very small
  - Requires (loss-augmented) primal-decode only

- Repeat:
  - Find the most violated constraint for an instance:

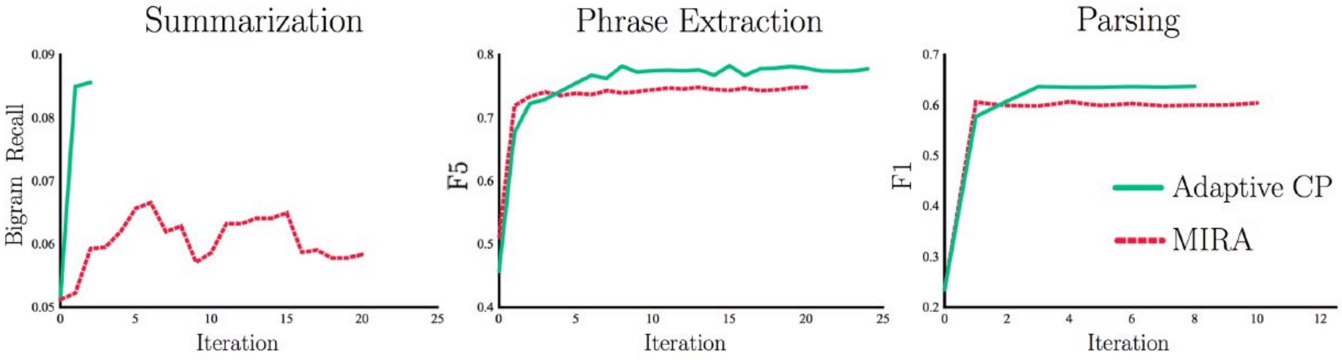
$$\forall \mathbf{y} \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$$

$$\arg\max_{\mathbf{y}} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$$

  - Add this constraint and resolve the (non-structured) QP (e.g. with SMO or other QP solver)

Cutting Plane (Dual)

- Some issues:
  - Can easily spend too much time solving QPs
  - Doesn't exploit shared constraint structure
  - In practice, works pretty well; fast like perceptron/MIRA, more stable, no averaging



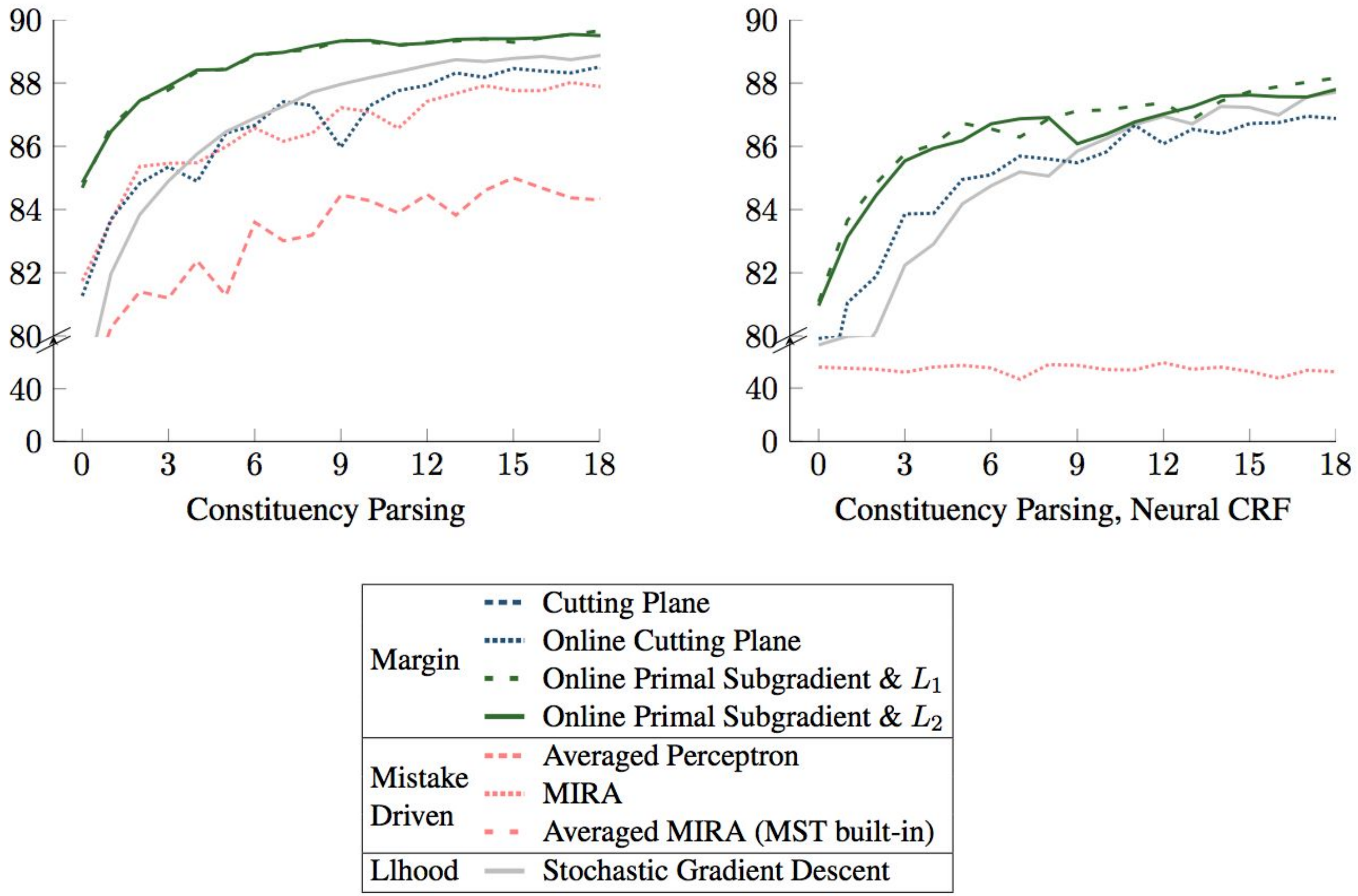
Likelihood, Structured

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

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = -2k\mathbf{w} + \sum_i \left(\mathbf{f}_i(\mathbf{y}_i^*) - \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}_i)\mathbf{f}_i(\mathbf{y}) \right)$$

- Structure needed to compute:
  - Log-normalizer
  - Expected feature counts
    - E.g. if a feature is an indicator of DT-NN then we need to compute posterior marginals P(DT-NN|sentence) for each position and sum
- Also works with latent variables (more later)

Comparison

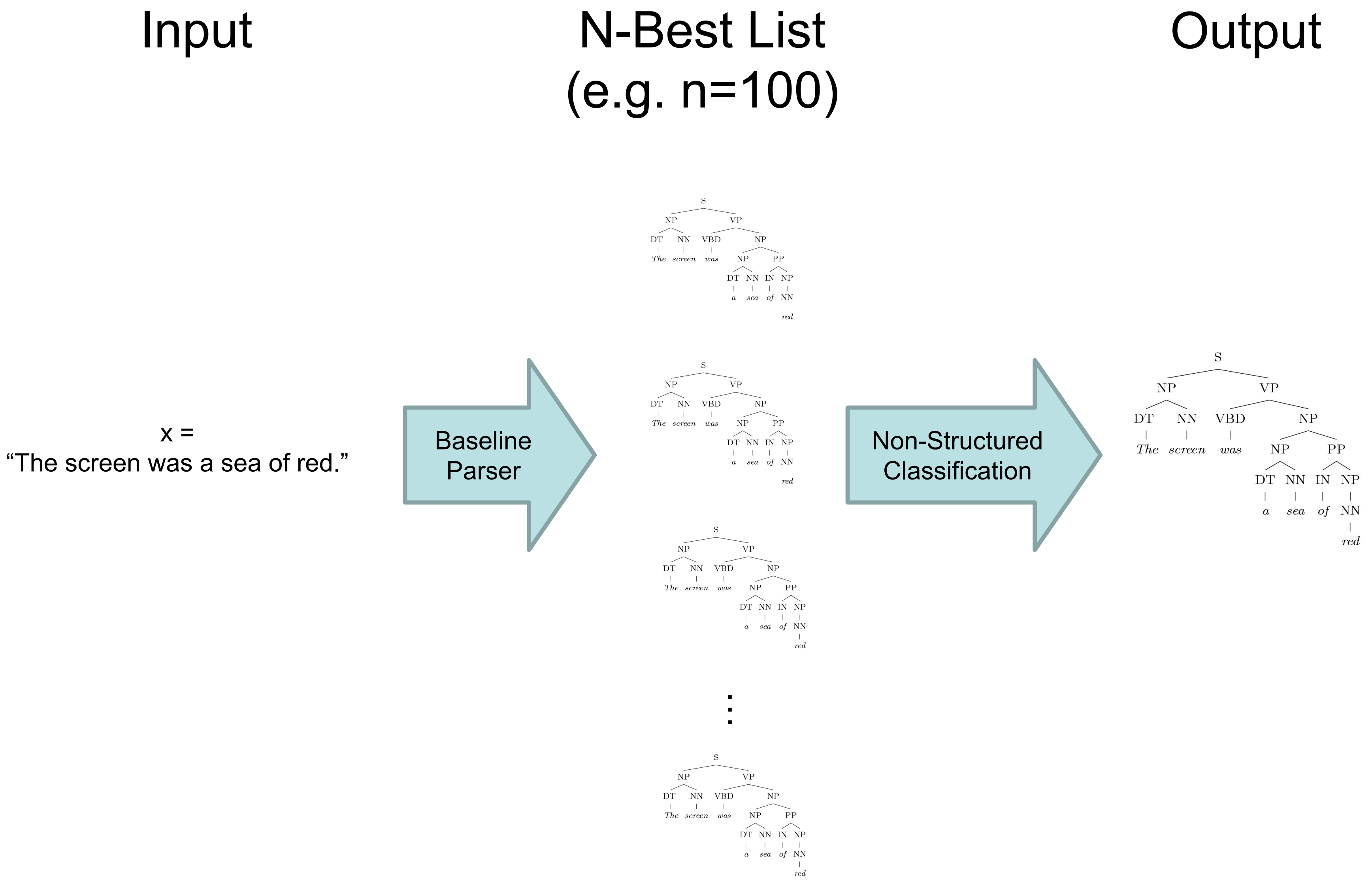


Constituency Parsing

Constituency Parsing, Neural CRF

Option 0: Reranking

[e.g. Charniak and Johnson 05]



Input

N-Best List
(e.g. n=100)

Output

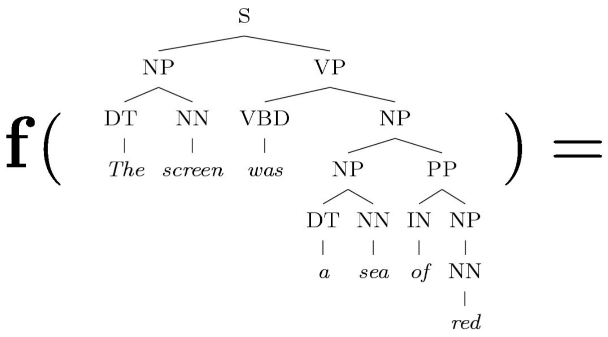
x =
“The screen was a sea of red.”

Baseline Parser

Non-Structured Classification

Reranking

- Advantages:
    - Directly reduce to non-structured case



- Disadvantages:
    - Stuck with errors of baseline parser
    - Baseline system must produce n-best lists
    - But, feedback is possible [McCloskey, Charniak, Johnson 2006]

M3Ns

- Another option: express all constraints in a packed form
  - Maximum margin Markov networks [Taskar et al 03]
  - Integrates solution structure deeply into the problem structure

- Steps
  - Express inference over constraints as an LP
  - Use duality to transform minimax formulation into min-min
  - Constraints factor in the dual along the same structure as the primal; alphas essentially act as a dual “distribution”
  - Various optimization possibilities in the dual

Example: Kernels

- Quadratic kernels

$$K(x, x') = (x \cdot x' + 1)^2$$

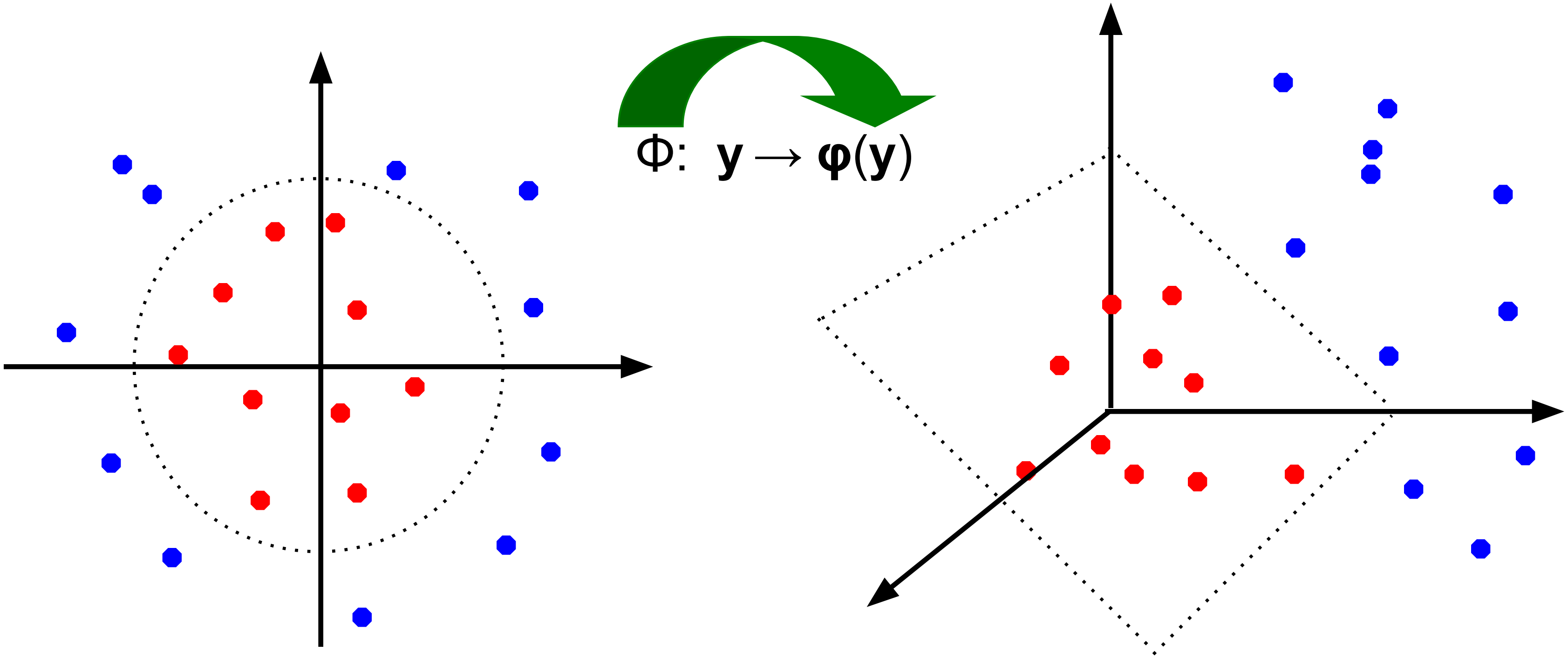
$$= \sum_{i,j} x_i x_j \, x'_i x'_j + 2 \sum_i x_i \, x'_i + 1$$

$$\Downarrow$$

$$K(\mathbf{y}, \mathbf{y}') = (\mathbf{f}(\mathbf{y})^\top \mathbf{f}(\mathbf{y}') + 1)^2$$

Non-Linear Separators

- Another view: kernels map an original feature space to some higher-dimensional feature space where the training set is (more) separable



Why Kernels?

- Can’t you just add these features on your own (e.g. add all pairs of features instead of using the quadratic kernel)?
  - Yes, in principle, just compute them
  - No need to modify any algorithms
  - But, number of features can get large (or infinite)
  - Some kernels not as usefully thought of in their expanded representation, e.g. RBF or data-defined kernels [Henderson and Titov 05]

- Kernels let us compute with these features implicitly
  - Example: implicit dot product in quadratic kernel takes much less space and time per dot product
  - Of course, there’s the cost for using the pure dual algorithms…