Algorithms for NLP

Classification II

Taylor Berg-Kirkpatrick – CMU

Slides: Dan Klein – UC Berkeley

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)

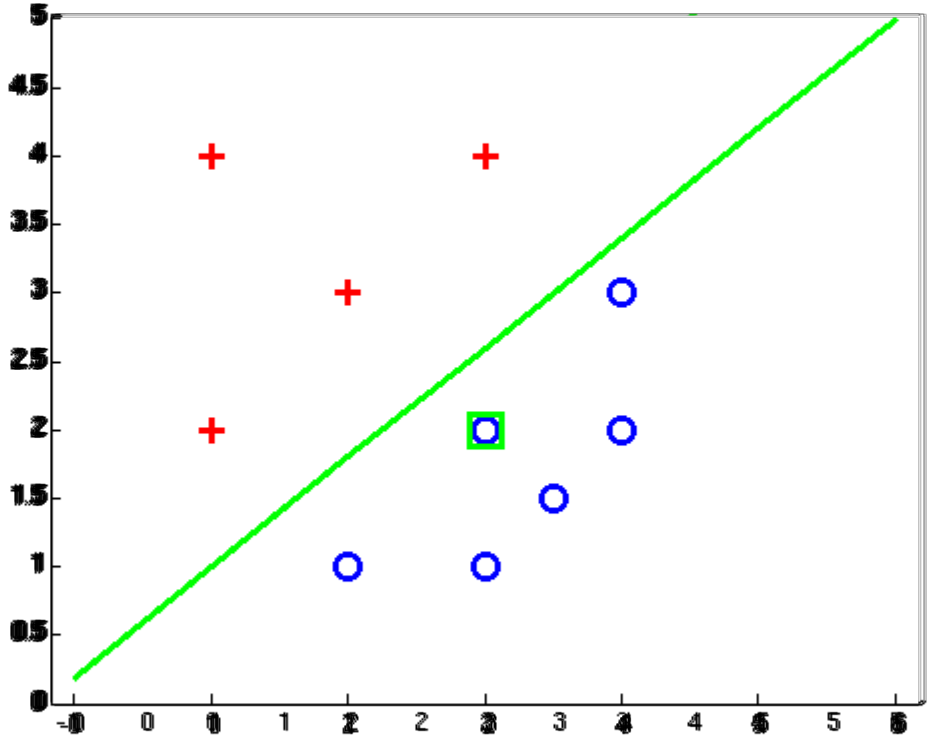
- We could, in principle, minimize training loss:

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

- This is a hard, discontinuous optimization problem

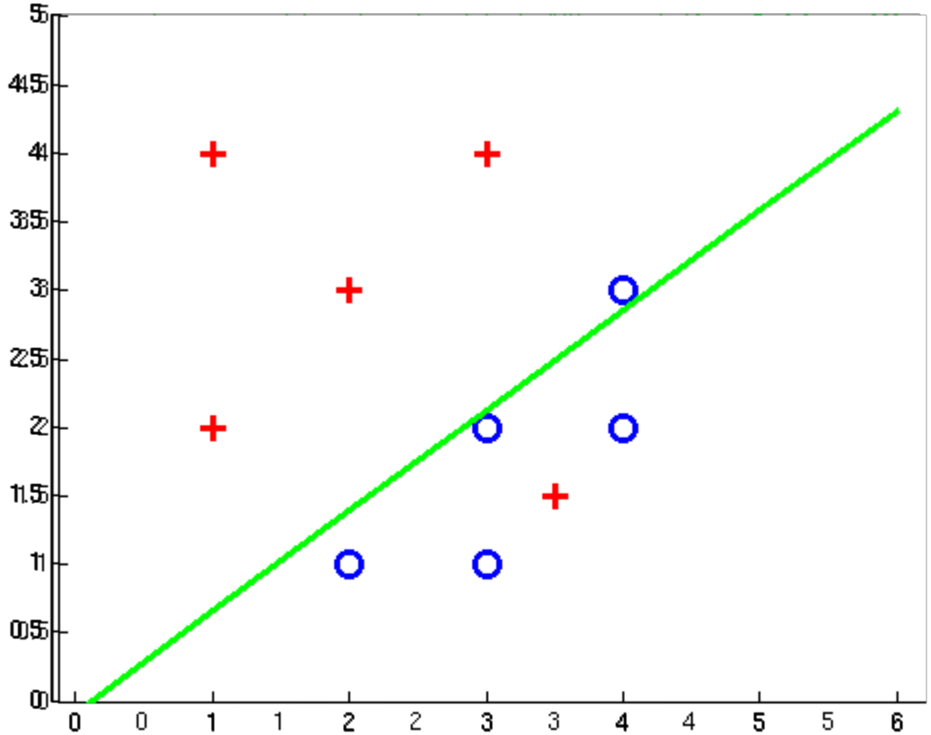
Examples: Perceptron

- Separable Case



Examples: Perceptron

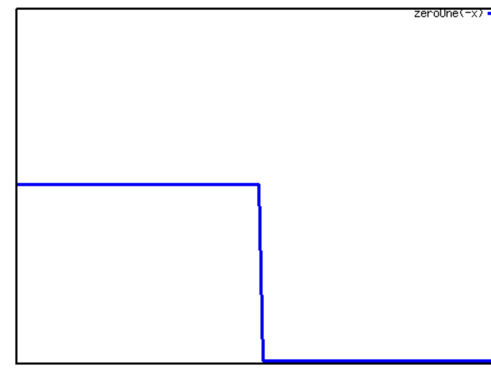
- Non-Separable Case



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
  - Maximum entropy and SVMs have other objectives related to zero-one loss

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)

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

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

Log-Loss

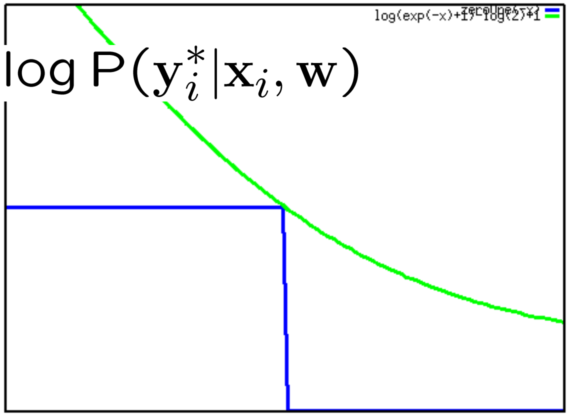
- If we view maxent as a minimization problem:

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

- This minimizes the "log loss" on each example

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

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



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

Maximum Margin

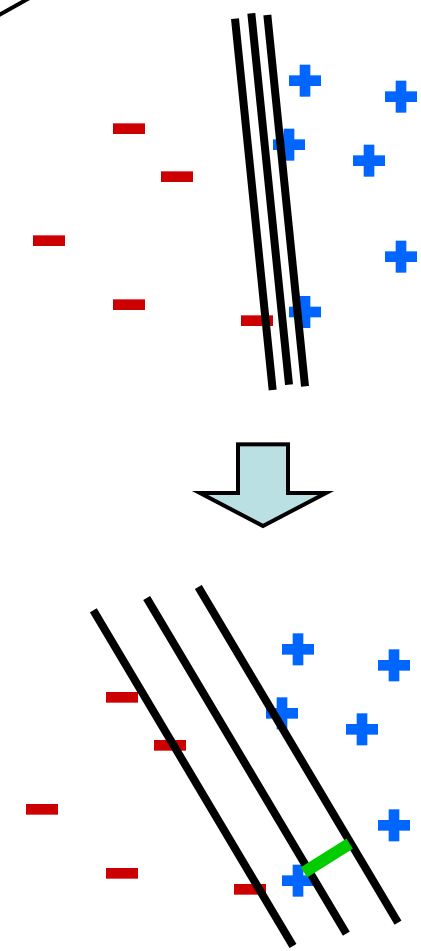
Note: exist other choices of how to penalize slacks!

- 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

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

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

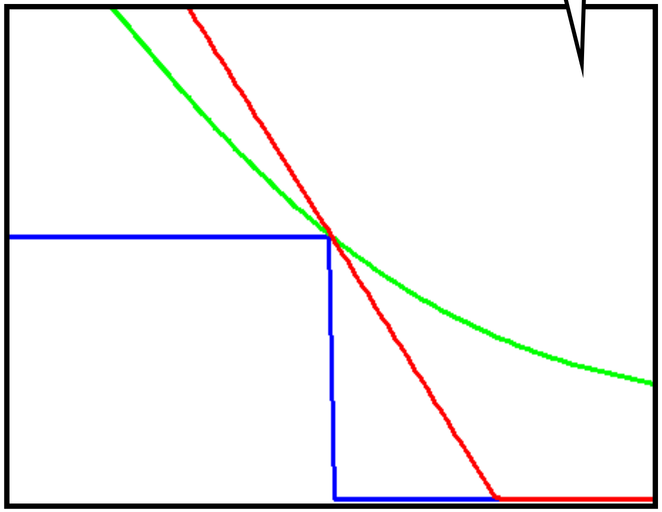
Hinge Loss

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:

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

- Maxent:

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

- 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

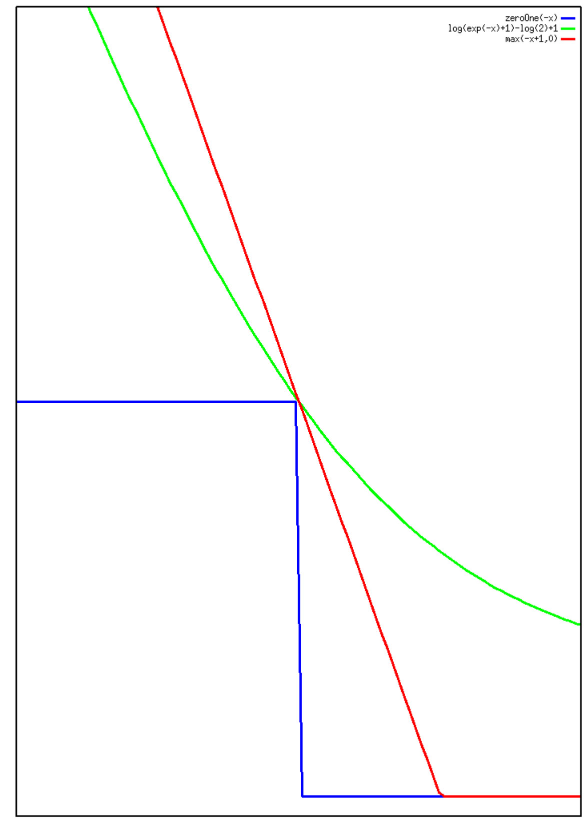
$$\sum_i step\left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \max_{\mathbf{y} \neq \mathbf{y}_i^*} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})\right)$$

- 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

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

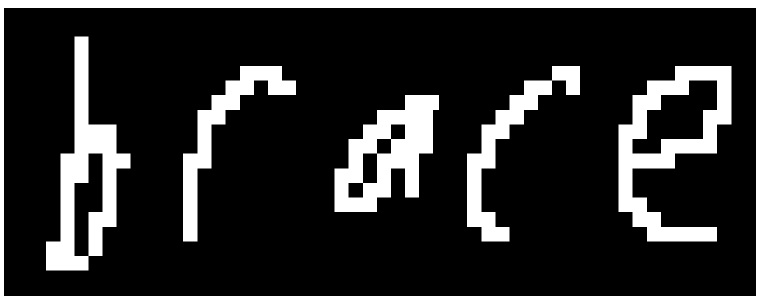


$$\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) - \max_{\mathbf{y} \neq \mathbf{y}_i^*} \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})\right)$$

Structure

Handwriting recognition

x **y**

 → **brace**

Sequential structure

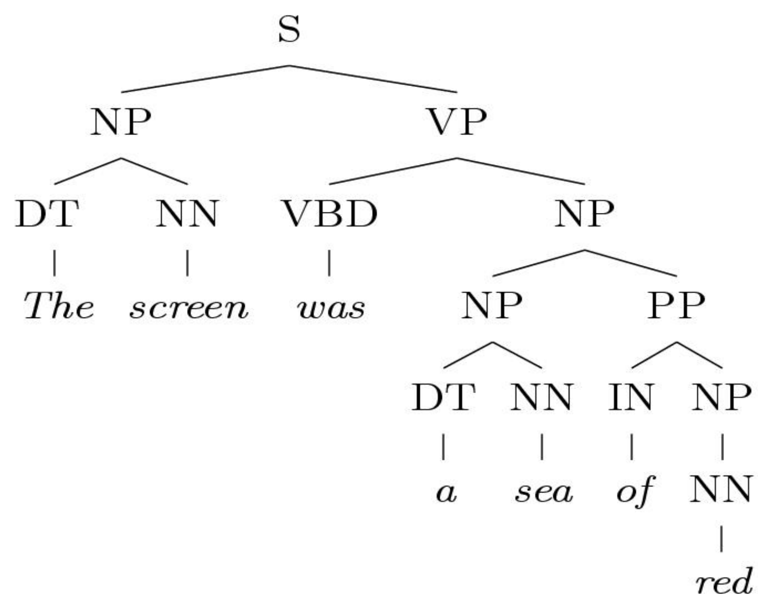
[Slides: Taskar and Klein 05]

CFG Parsing

x

The screen was a sea of red

y



Recursive structure

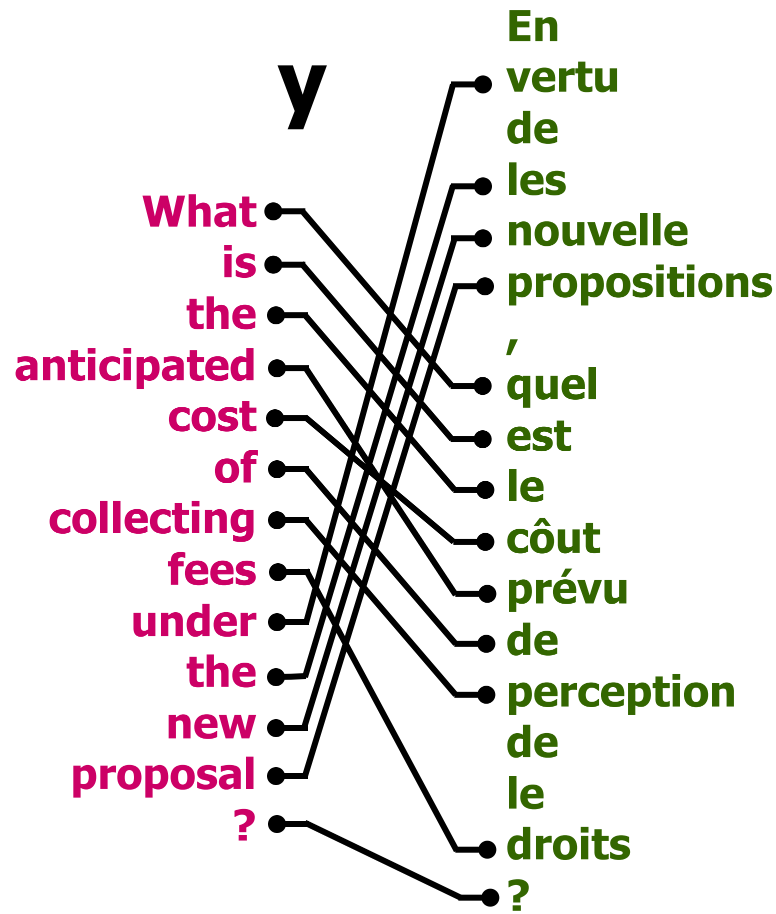
Bilingual Word Alignment

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



Combinatorial structure

Structured Models

$$prediction(\mathbf{x}, \mathbf{w}) = \underset{\mathbf{y} \in \mathcal{Y}(\mathbf{x})}{\arg\max}\; score(\mathbf{y}, \mathbf{w})$$

space of feasible outputs

Assumption:

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

Score is a sum of local “part” scores

Parts = nodes, edges, productions

Named Entity Recognition

$$f(x, y) = \sum_{(y_{i-1}, y_i) \in y} f(y_{i-1}, y_i) + \sum_{(x_i, y_i)} f(x_i, y_i)$$

$f(y_{i-1}, y_i)$

y_{i-1} y_i

ORG ORG --- ORG — ORG ORG --- --- LOC

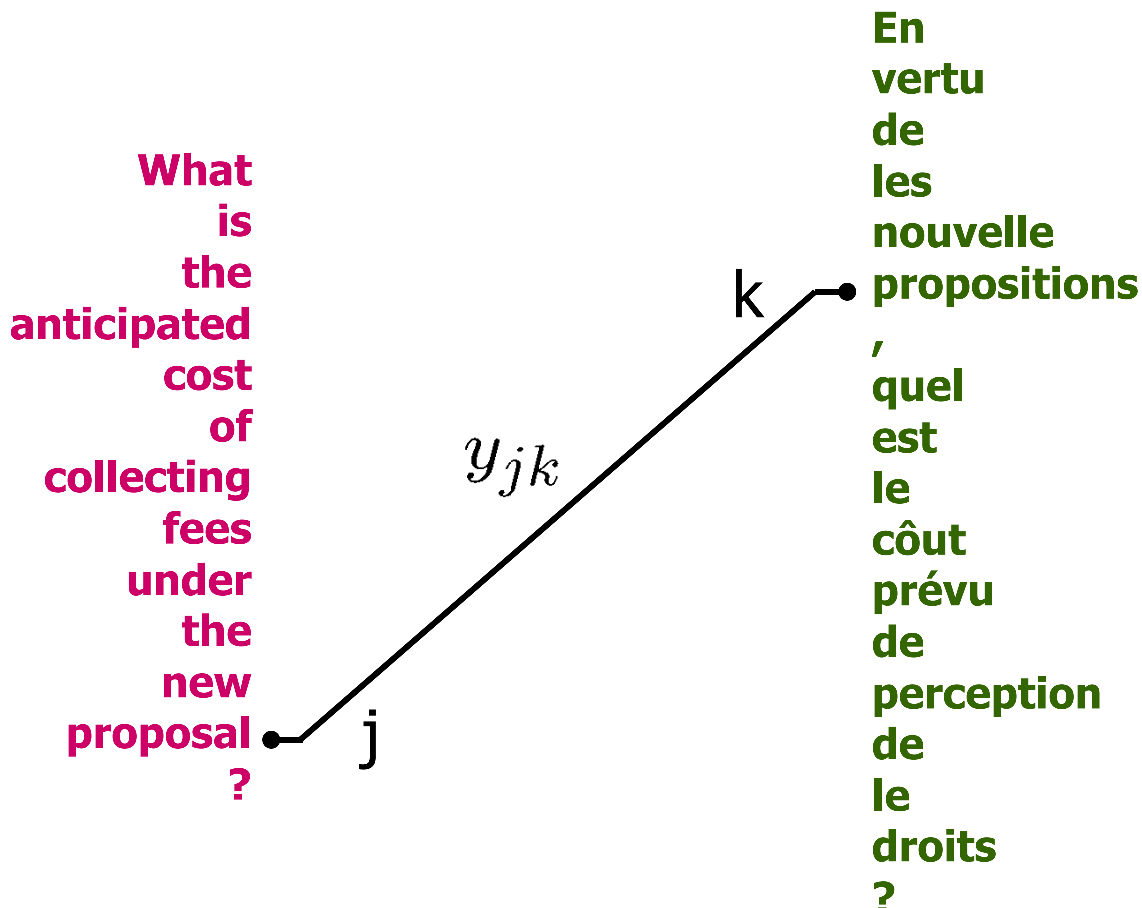
$f(x_i, y_i)$

Apple Computer bought Smart Systems Inc. located in Arkansas.

x_i

Bilingual word alignment

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



$$f(x, y_{jk})$$

- association
- position
- orthography

Efficient Decoding

- Common case: you have a black box which computes

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

  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 (Dual)

- Remember the constrained version of primal:

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

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

- Dual has a variable for every constraint here

Full Margin: OCR

- We want:

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

- Equivalently:

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

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

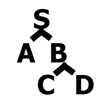
$$\cdots$$

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

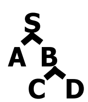
a lot!

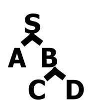
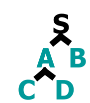
Parsing example

- We want:

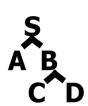
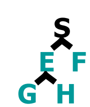
$\arg\max_{y}\ w^\top f($ **'It was red'** $, y) =$ 

- Equivalently:

$w^\top f($**'It was red',**  $) > w^\top f($**'It was red',**  $)$

$w^\top f($**'It was red',**  $) > w^\top f($**'It was red',**  $)$

...

$w^\top f($**'It was red',**  $) > w^\top f($**'It was red',**  $)$

a lot!

Alignment example

- We want:

$$\arg\max_{\mathbf{y}} \ \mathbf{w}^\top \mathbf{f}\left(\begin{matrix}\text{'What is the'}\\\text{'Quel est le'}\end{matrix}, \mathbf{y}\right) = \begin{matrix}1 \leftrightarrow 1\\2 \leftrightarrow 2\\3 \leftrightarrow 3\end{matrix}$$

- Equivalently:

$$\mathbf{w}^\top \mathbf{f}\left(\begin{matrix}\text{'What is the'}\\\text{'Quel est le'}\end{matrix}, \begin{matrix}1 \leftrightarrow 1\\2 \leftrightarrow 2\\3 \leftrightarrow 3\end{matrix}\right) > \mathbf{w}^\top \mathbf{f}\left(\begin{matrix}\text{'What is the'}\\\text{'Quel est le'}\end{matrix}, \begin{matrix}1 \leftrightarrow 1\\2 \leftrightarrow 3\\3 \leftrightarrow 2\end{matrix}\right)$$

$$\mathbf{w}^\top \mathbf{f}\left(\begin{matrix}\text{'What is the'}\\\text{'Quel est le'}\end{matrix}, \begin{matrix}1 \leftrightarrow 1\\2 \leftrightarrow 2\\3 \leftrightarrow 3\end{matrix}\right) > \mathbf{w}^\top \mathbf{f}\left(\begin{matrix}\text{'What is the'}\\\text{'Quel est le'}\end{matrix}, \begin{matrix}1 \leftrightarrow 2\\2 \leftrightarrow 1\\3 \leftrightarrow 3\end{matrix}\right)$$

$$\cdots$$

$$\mathbf{w}^\top \mathbf{f}\left(\begin{matrix}\text{'What is the'}\\\text{'Quel est le'}\end{matrix}, \begin{matrix}1 \leftrightarrow 1\\2 \leftrightarrow 2\\3 \leftrightarrow 3\end{matrix}\right) > \mathbf{w}^\top \mathbf{f}\left(\begin{matrix}\text{'What is the'}\\\text{'Quel est le'}\end{matrix}, \begin{matrix}1 \leftrightarrow 3\\2 \leftrightarrow 2\\3 \leftrightarrow 1\end{matrix}\right)$$

a lot!

Cutting Plane (Dual)

- 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:

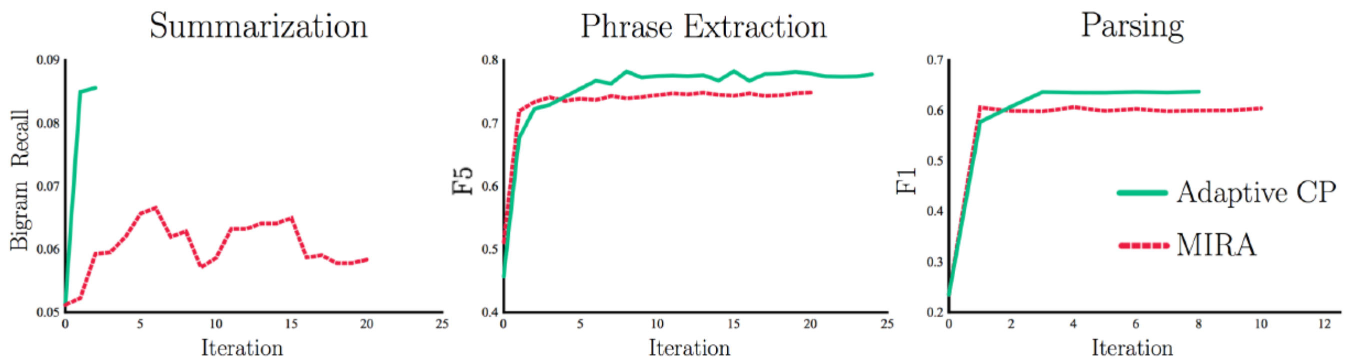
$$\forall \mathbf{y} \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i^*) \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$$

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

  - 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



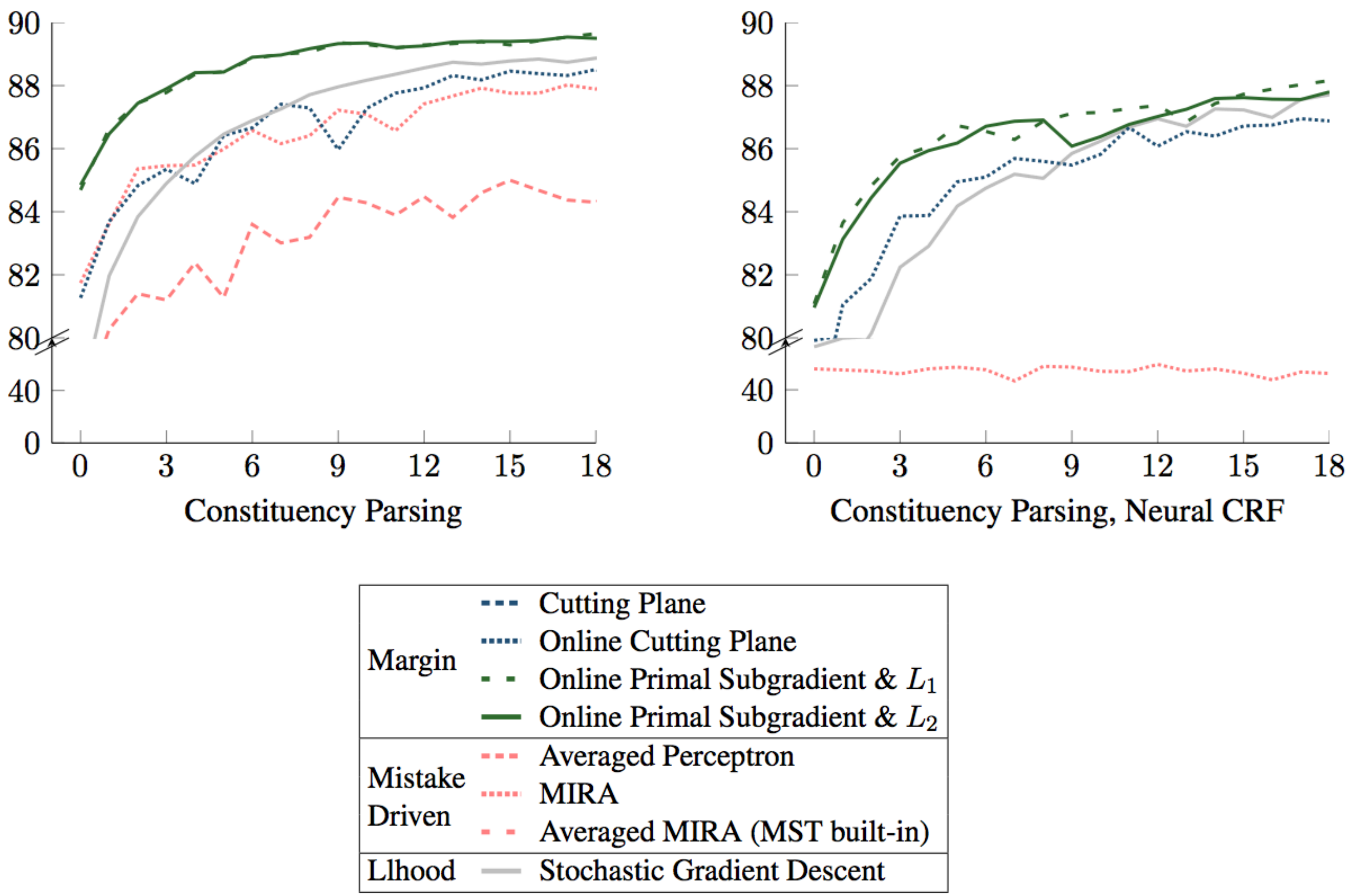
Likelihood, Structured

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

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

- 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



90
88
86
84
82
80
40
0
0
3
6
9
12
15
18
Constituency Parsing
Constituency Parsing, Neural CRF
Margin
Cutting Plane
Online Cutting Plane
Online Primal Subgradient & L_1
Online Primal Subgradient & L_2
Mistake Driven
Averaged Perceptron
MIRA
Averaged MIRA (MST built-in)
Llhood
Stochastic Gradient Descent

Option 0: Reranking

[e.g. Charniak and Johnson 05]

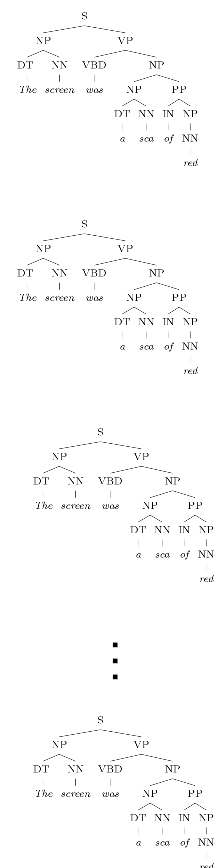
Input

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

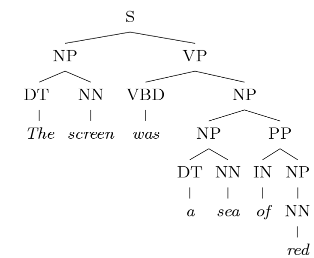
Output

x =
"The screen was a sea of red."

Baseline Parser

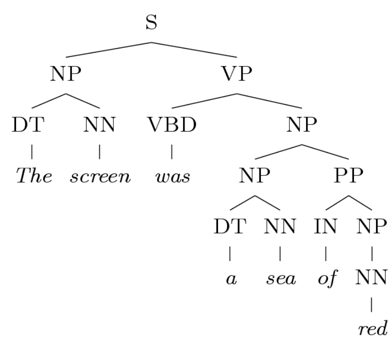


Non-Structured Classification



Reranking

- Advantages:
  - Directly reduce to non-structured case
  - No locality restriction on features

$\mathbf{f}($  $) =$

- 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

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

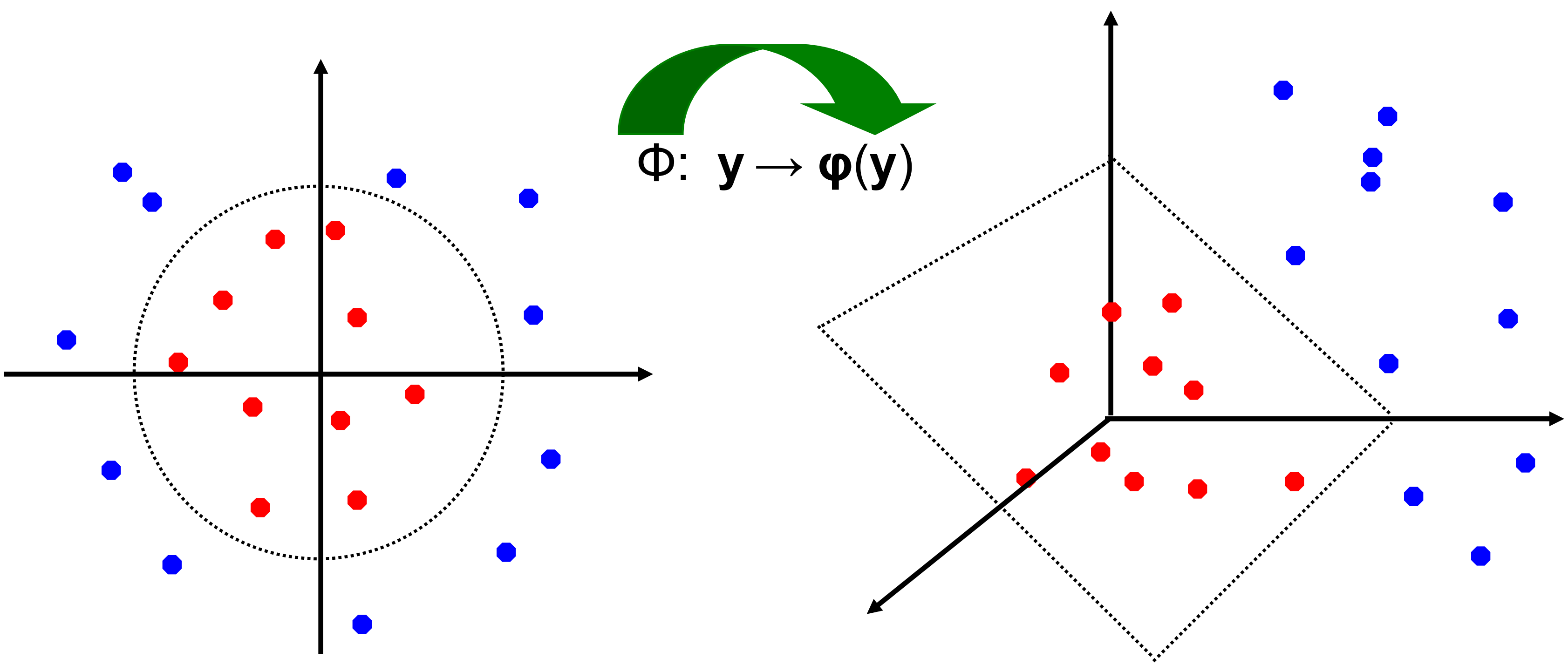
$$= \sum_{i,j} x_i x_j \, x'_i x'_j + 2 \sum_i x_i \, x'_i + 1$$

$$\Downarrow$$

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

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…