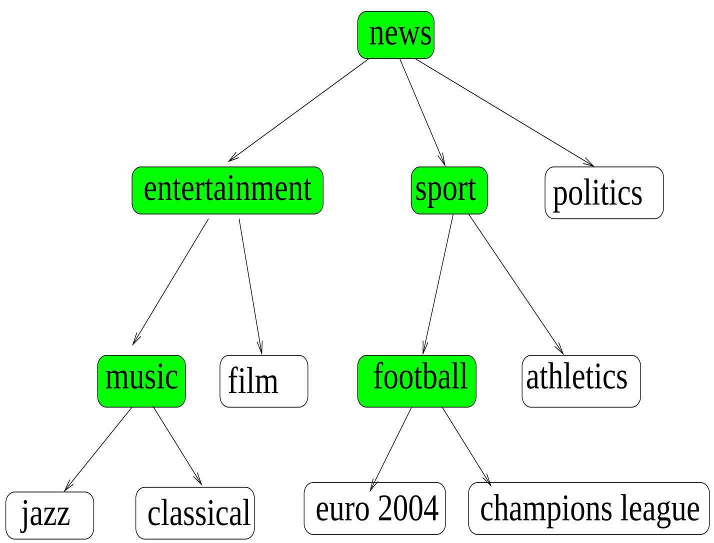
On Max-Margin Markov Networks in Hierarchical Document Classification

Juho Rousu
Department of Computer Science
University of Helsinki, Finland

J. Rousu, C. Saunders, S. Szedmak and J. Shawe-Taylor: Kernel-based Learning of Hierarchical Multilabel Classification Models, Journal of Machine Learning Research, 2006, in press

Hierarchical Multilabel Classification: union of partial paths model

Goal: Given document x, and hierachy $T = (V, E)$, predict multilabel $\mathbf{y} \in \{+1, -1\}^k$ where the positive microlabels y_i form a union of partial paths in T



Frequently used learning strategies for hierarchies

- **Flatten the hierarchy**: Learn each microlabel independently with classification learner of your choice
  - Computationally relatively inexpensive
  - Does not make use of the dependencies between the microlabels
- **Hierarchical training**: Train a node j with examples $(x, \mathbf{y})$ that belong to the parent, i.e. $y_{pa(j)} = 1$.
  - Some of the microlabel dependencies are learned.
  - However, training data fragments towards the leaves, hence estimation becomes less reliable
  - Model is not explicitly trained in terms of a loss function for the hierarchy.

We wish to improve on these approaches...

The classification model

Make the hierarchy a Conditional Random Field (aka Markov Network) $T = (V, E)$ with the exponential family.

$$P(\mathbf{y}|x, \mathbf{w}) = Z(x, \mathbf{w})^{-1} \prod_{e \in E} \exp\left(\mathbf{w}_e^T \boldsymbol{\phi}_e(x, \mathbf{y}_e)\right) = \exp\left(\mathbf{w}^T \boldsymbol{\phi}(x, \mathbf{y})\right)$$

- $\mathbf{y}_e = (y_i, y_j)$ is an edge-labeling, i.e. a restriction of the whole multilabel $\mathbf{y}$ into the edge $e = (i, j)$
- $\boldsymbol{\phi}_e(x, \mathbf{y}_e)$ is a joint feature map for the pair $(x, \mathbf{y}_e)$
- $\mathbf{w} = (\mathbf{w}_e)_{e \in E}$ is the weight vector to be learned
- $Z(x, \mathbf{w}) = \sum_{\mathbf{y} \in \{+1,-1\}^k} \exp\left(\mathbf{w}^T \boldsymbol{\phi}(x, \mathbf{y})\right)$ is a normalization factor (aka partition function).

Feature vectors

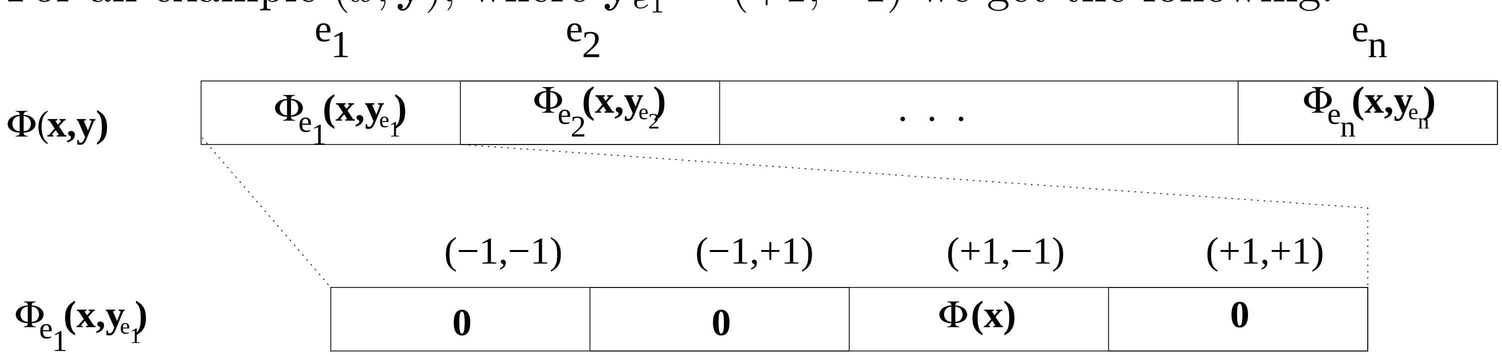
The joint feature vector $\boldsymbol{\phi}(x, \mathbf{y})$ is composed of blocks

$$\boldsymbol{\phi}_e^{\mathbf{u}_e}(x, \mathbf{y}_e) = [\![\mathbf{y}_e = \mathbf{u}_e]\!]\boldsymbol{\phi}(x), e \in E, \mathbf{u}_e \in \{+1, -1\}^2$$

where $\boldsymbol{\phi}(x)$ is some feature representation of x (e.g. bag of words, substring spectrum,...)

- This representation allows us to learn different feature weights for different contexts.
- The special structure of repeating $\boldsymbol{\phi}(x)$ can be utilized to save memory

For an example $(x, \mathbf{y})$, where $\mathbf{y}_{e_1} = (+1, -1)$ we get the following:



Loss functions for hierarchies

Consider a true multilabel $\mathbf{y} = (y_1, \ldots, y_k) \in \{+1,-1\}^k$, and a predicted one $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_k)$. Many choices:

- **Zero-one loss**: $\ell_{0/1}(\mathbf{y}, \hat{\mathbf{y}}) = [\![\mathbf{y} \neq \hat{\mathbf{y}}]\!]$; treats all incorrect multilabels alike
- **Hamming loss**: $\ell_\Delta(\mathbf{y}, \hat{\mathbf{y}}) = \sum_j [\![y_j \neq \hat{y}_j]\!]$; counts incorrect microlabels.

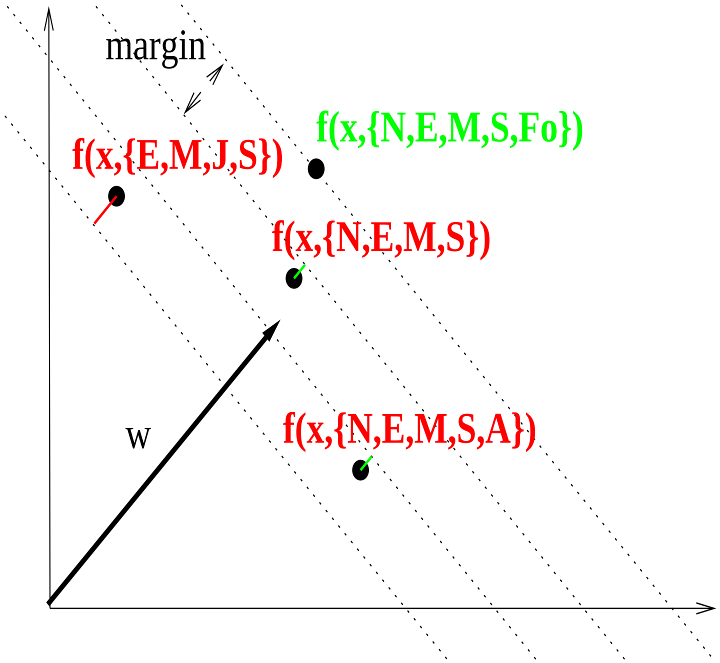
Neither of the above takes the hierarchy into account. These do:

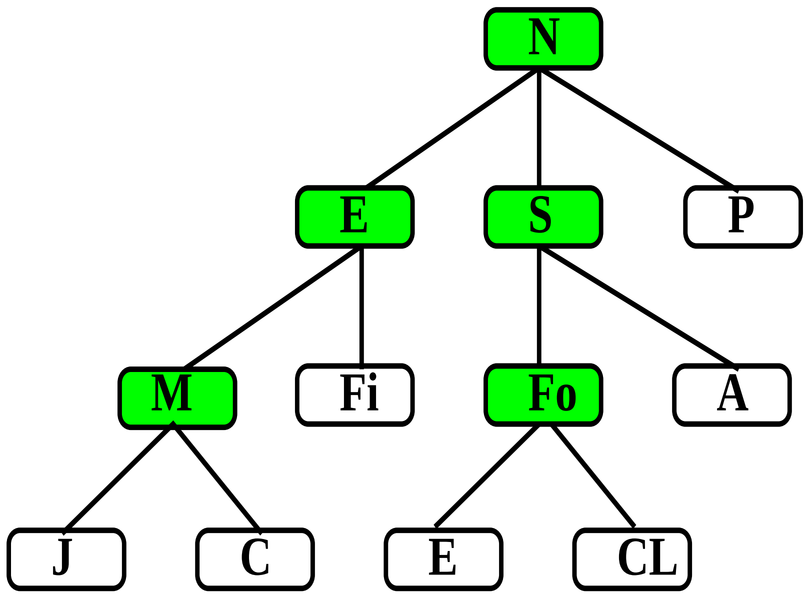
- **Path loss** (Cesa-Bianchi et al. 2004): $\ell_H(\mathbf{y}, \hat{\mathbf{y}}) = \sum_j c_j [\![y_j \neq \hat{y}_j \;\;\&\;\; y_k = \hat{y}_k \forall k \in ancestors(j)]\!]$; the first mistake along a path is penalized
- **Edge loss**: $\ell_{\tilde{H}}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_j c_j [\![y_j \neq \hat{y}_j \;\;\&\;\; y_{parent(j)} = \hat{y}_{parent(j)}]\!]$; mistake in the child is penalized if the parent was correct.

Max-margin Structured output learning
(Taskar et al., 2004; Tsochantaridis et al., 2004; ...)

Goal:

- Separate the correct multilabel from the incorrect ones by a large margin.
- Let the targeted margin scale proportionally to the loss of the multilabel
- Allow slack for non-separability of data





Optimization problem

Primal form:

$$\min_{\mathbf{w},\boldsymbol{\xi}\geq 0} \frac{1}{2}\left|\left|\mathbf{w}\right|\right|^2 + C\sum_{i=1}^{m}\xi_i$$

$$\text{s.t. } \mathbf{w}^T\left(\phi(\mathbf{x}_i,\mathbf{y}_i) - \phi(\mathbf{x}_i,\mathbf{y})\right) \geq \boldsymbol{\ell}(\mathbf{y}_i,\mathbf{y}) - \xi_i, \forall i, \mathbf{y} \in \{+1,-1\}^k$$

Dual:

$$\max_{\boldsymbol{\alpha}>0} \sum_{i,\mathbf{y}}\alpha(x_i,\mathbf{y})\ell(\mathbf{y}_i,\mathbf{y}) - \frac{1}{2}\sum_{x_i,\mathbf{y}}\sum_{x'_i,\mathbf{y}'}\alpha(x_i,\mathbf{y})^T K(x_i,\mathbf{y};x'_i,\mathbf{y}')\alpha(x'_i,\mathbf{y}')$$

$$\text{s.t.} \sum_{\mathbf{y}}\alpha(x_i,\mathbf{y}) \leq C, \forall i$$

- Exponential number (in size of the hierarchy) of primal constraints and dual variables, one per pseudo-example $(x_i,\mathbf{y})$
- Cannot be solved in this form for realistic-sized datasets, many approaches to make the model tractable (Taskar et al., 2004, 2005; Tshochantaridis et al. 2004)

Marginalized problem

A polynomial-sized problem can be obtained by marginalization (c.f. Taskar *et al.* , 2004), if the loss function and the feature representation is chosen suitably.

Our choices:

- Edge-marginals of dual variables : $\mu_e(x, \mathbf{y}_e) = \sum_{\mathbf{u}|\mathbf{u}_e=\mathbf{y}_e} \alpha(x, \mathbf{u})$
- Loss function decomposable by the edges: $\ell(\mathbf{y}, \mathbf{y}') = \sum_{e \in E} \ell(\mathbf{y}_e, \mathbf{y}'_e)$; Hamming loss and edge loss apply
- Kernel decomposable by the edges: $K(x, \mathbf{y}; x', \mathbf{y}') = \sum_{e \in E} K_e(x, \mathbf{y}_e; x, \mathbf{y}'_e)$;

Marginalized problem

b

$$\max_{\boldsymbol{\mu}>0} \sum_{e\in E} \boldsymbol{\mu}_e^T \boldsymbol{\ell}_e - \frac{1}{2}\sum_{e\in E} \boldsymbol{\mu}_e^T K_e \boldsymbol{\mu}_e$$
$$\text{s.t } B_{ie}\boldsymbol{\mu}_{ie} \leq C, \forall\ i, e \in E,$$
$$A_i \boldsymbol{\mu}_i = 0, \forall\ i$$

- The matrices B_{ie} encode box constraints $\sum_{y,y'} \mu_e(i,y,y') \leq C$
- The matrices A_i encode marginal consistency constraints $\sum_{y'} \mu_e(i,y',y) = \sum_{y'} \mu_{e'}(i,y,y'),\ \forall y,(e,e') : e = parent(e')$; these need to be inserted to make the problem correspond to the original dual problem.
- The number of marginal dual variables μ_e is $O(m|E|)$, the edge-kernels K_e take $O(m^2|E|)$ space, which is too much even for medium-sized datasets
- e.g. optimizing 1372 examples by 188 microlabels will consume $> 10Gb$ memory!

Decomposing the model

$$\max_{\boldsymbol{\mu}>0} \sum_{e\in E} \boldsymbol{\mu}_e^T \boldsymbol{\ell}_e - \frac{1}{2}\sum_{e\in E} \boldsymbol{\mu}_e^T K_e \boldsymbol{\mu}_e$$

$$\text{s.t } B_{ie}\boldsymbol{\mu}_{ie} \leq C, \forall\ i, e \in E,$$

$$A_i\boldsymbol{\mu}_i = 0, \forall\ i$$

- Consistency constraints $A_i\boldsymbol{\mu}_i = 0$ tie the edges together
- Kernels K_e tie training examples together
- But the gradient of the objective $\mathbf{g} = \boldsymbol{\ell} - (K_e\boldsymbol{\mu}_e)_{e\in E}$ does not the contain example interactions

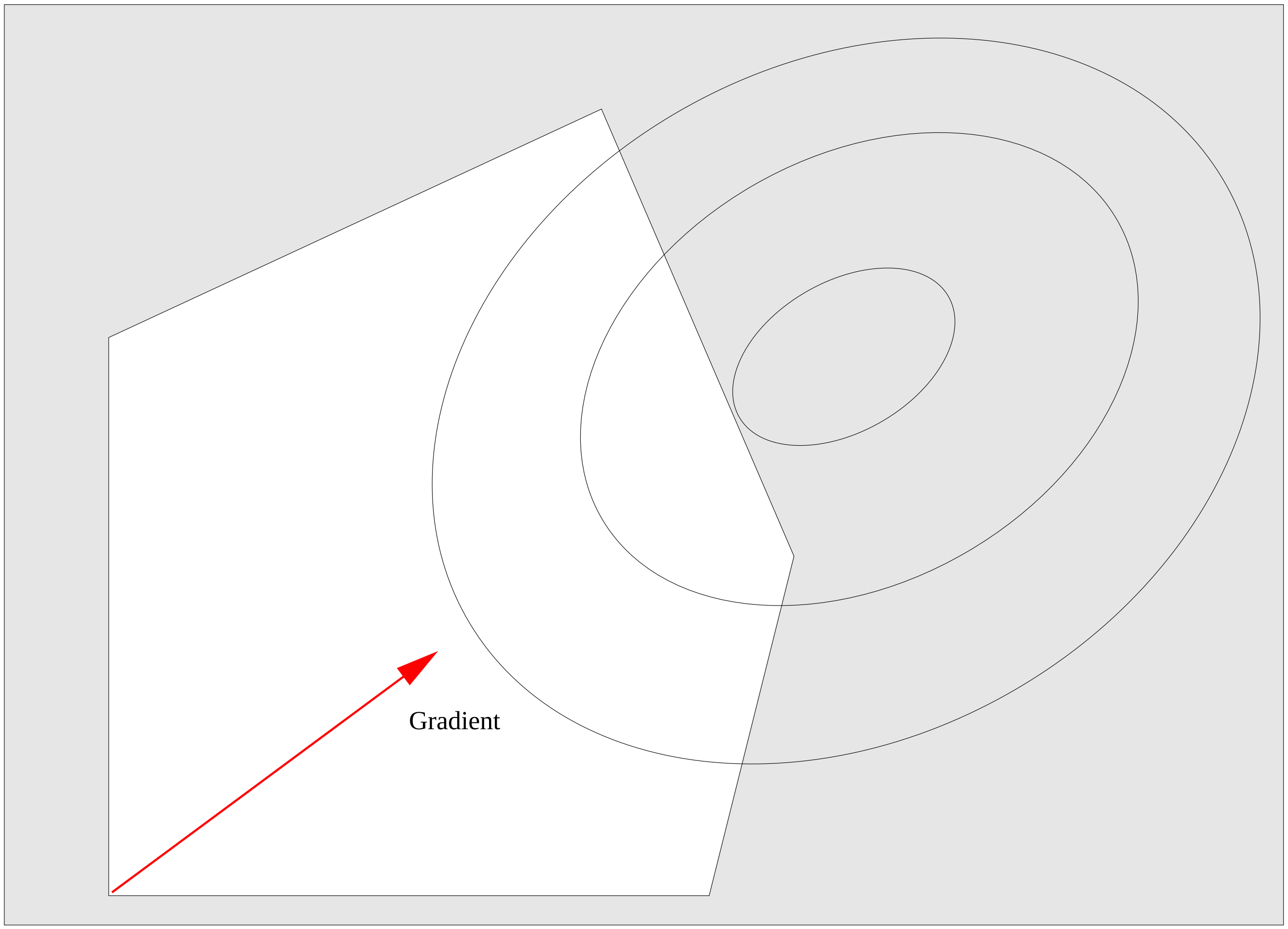
$\Rightarrow$ Iterative, gradient-based methods allow decomposed training, one example at a time

Conditional Gradient method

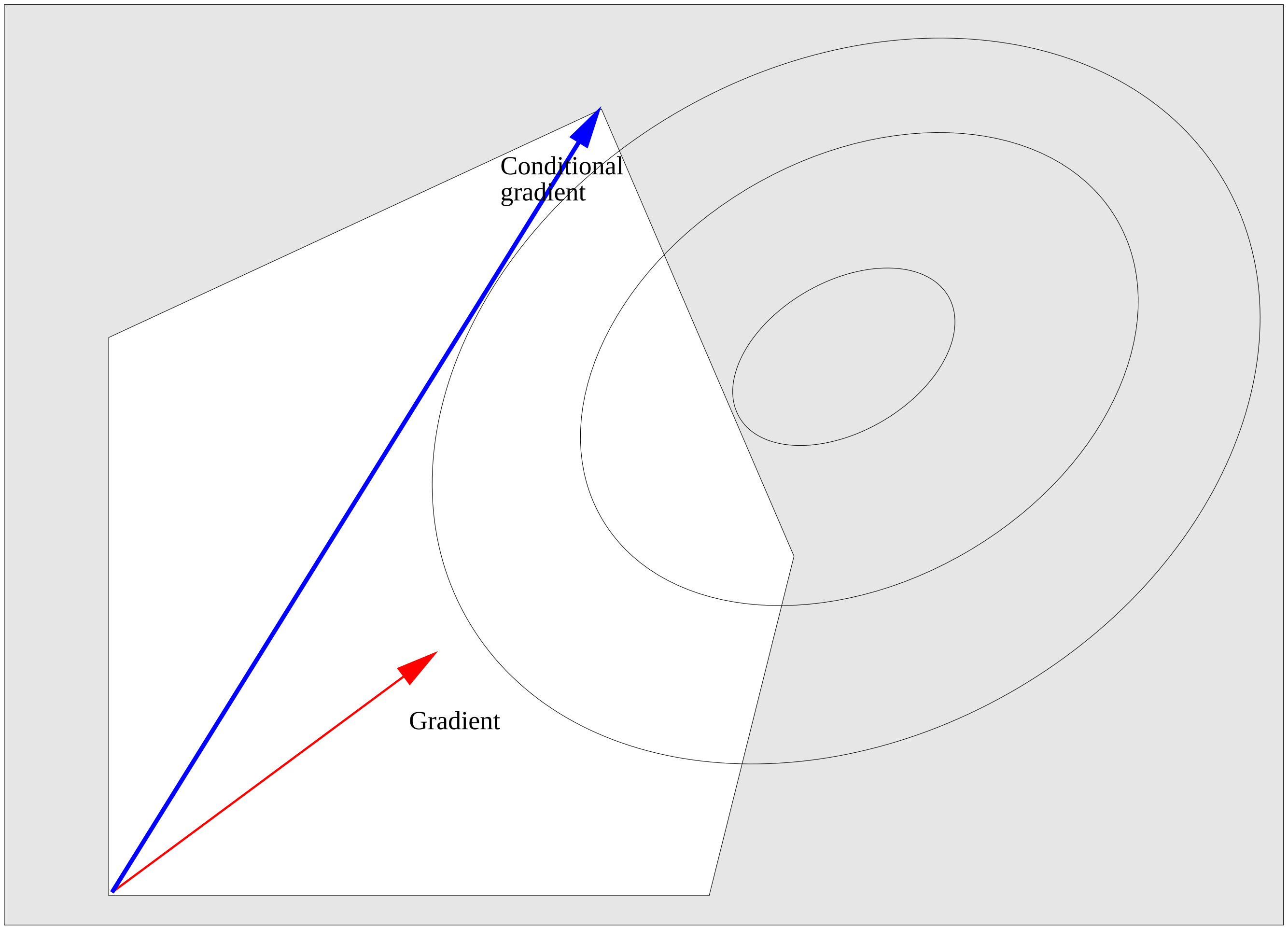
We use Conditional Gradient Descent (c.f. Bertsekas, 1999) to optimize the marginalized dual problem Ingredients:

- Iterative gradient search in the feasible set
- Update direction is the highest feasible point assuming current gradient; found by solving a constrained linear program: $\max_{\boldsymbol{\mu} \in \mathcal{F}} (\boldsymbol{\ell} - K\boldsymbol{\mu}_0)^T \boldsymbol{\mu}$
- updates within single-example subspaces can be done independently, after obtaining an initial gradient.

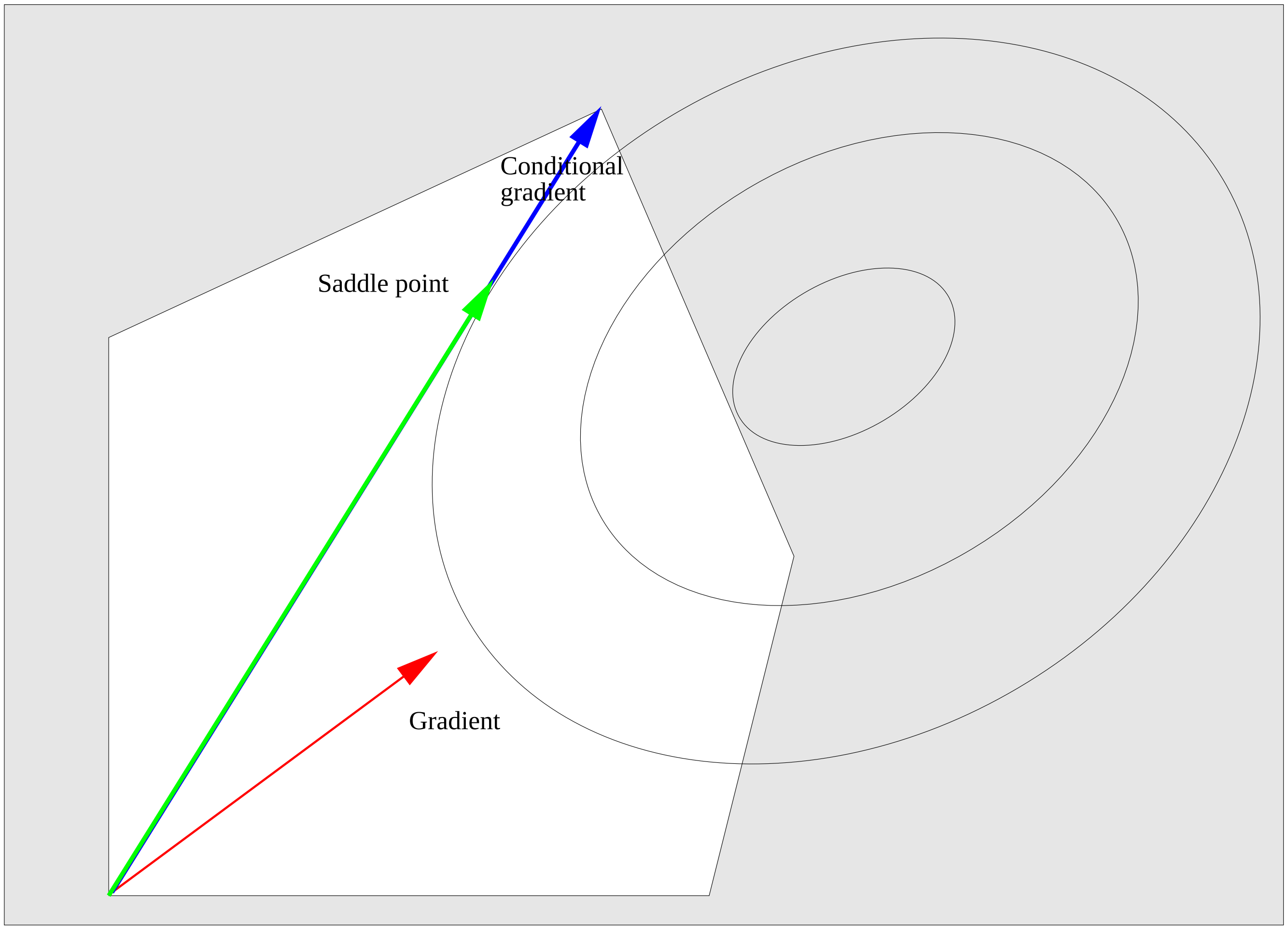
Conditional Gradient Ascent



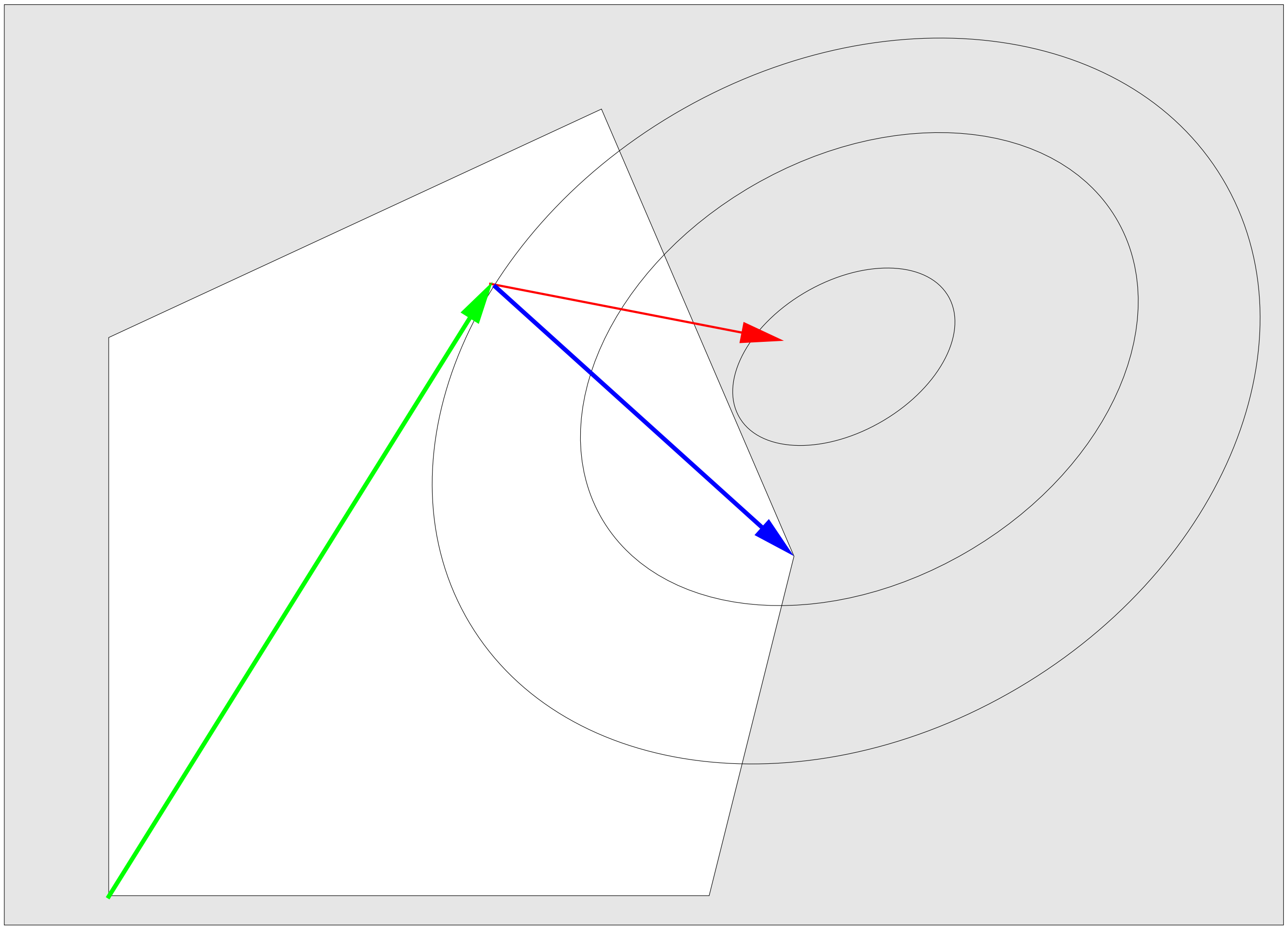
Conditional Gradient Ascent



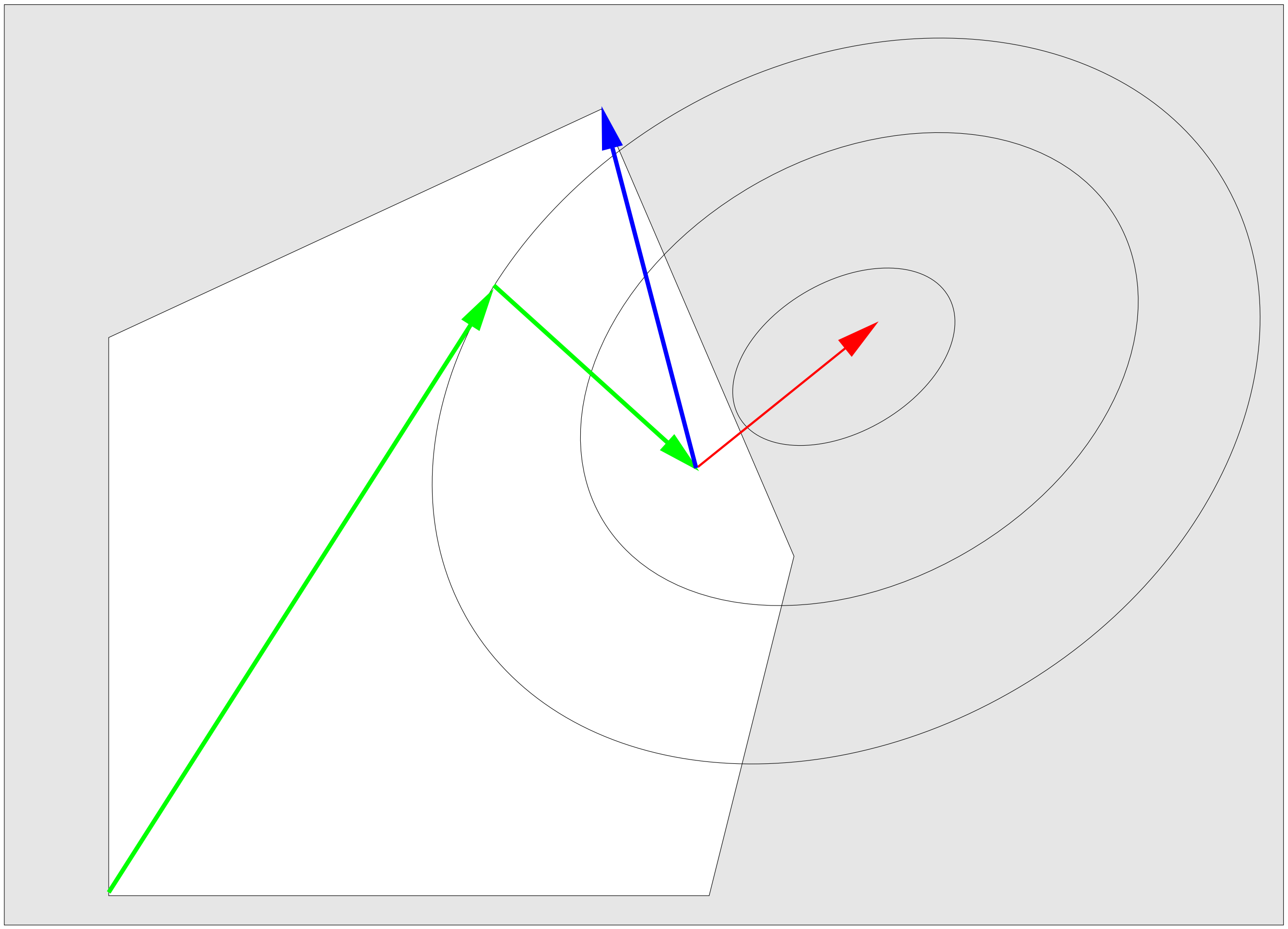
Conditional Gradient Ascent



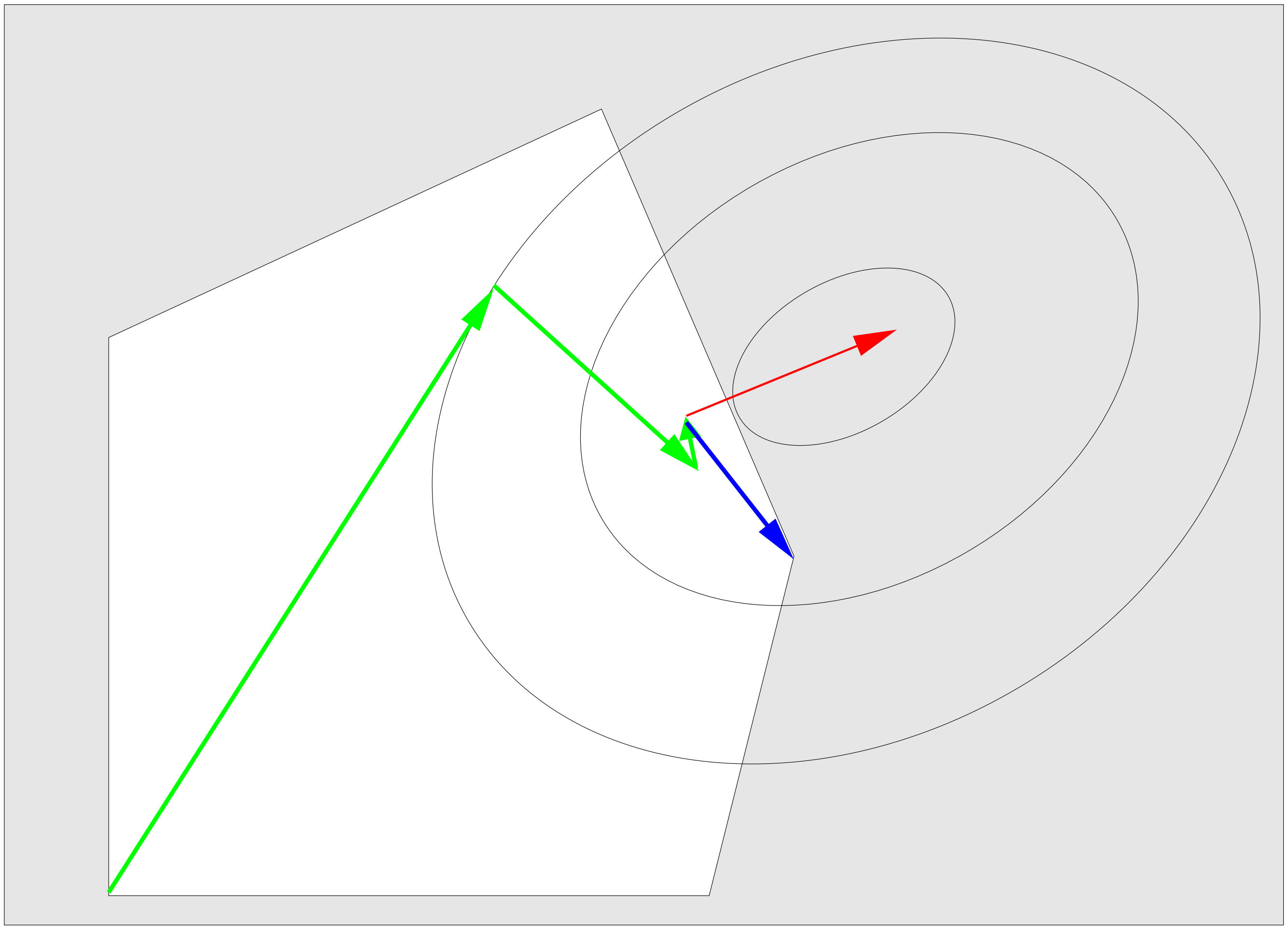
Conditional Gradient Ascent



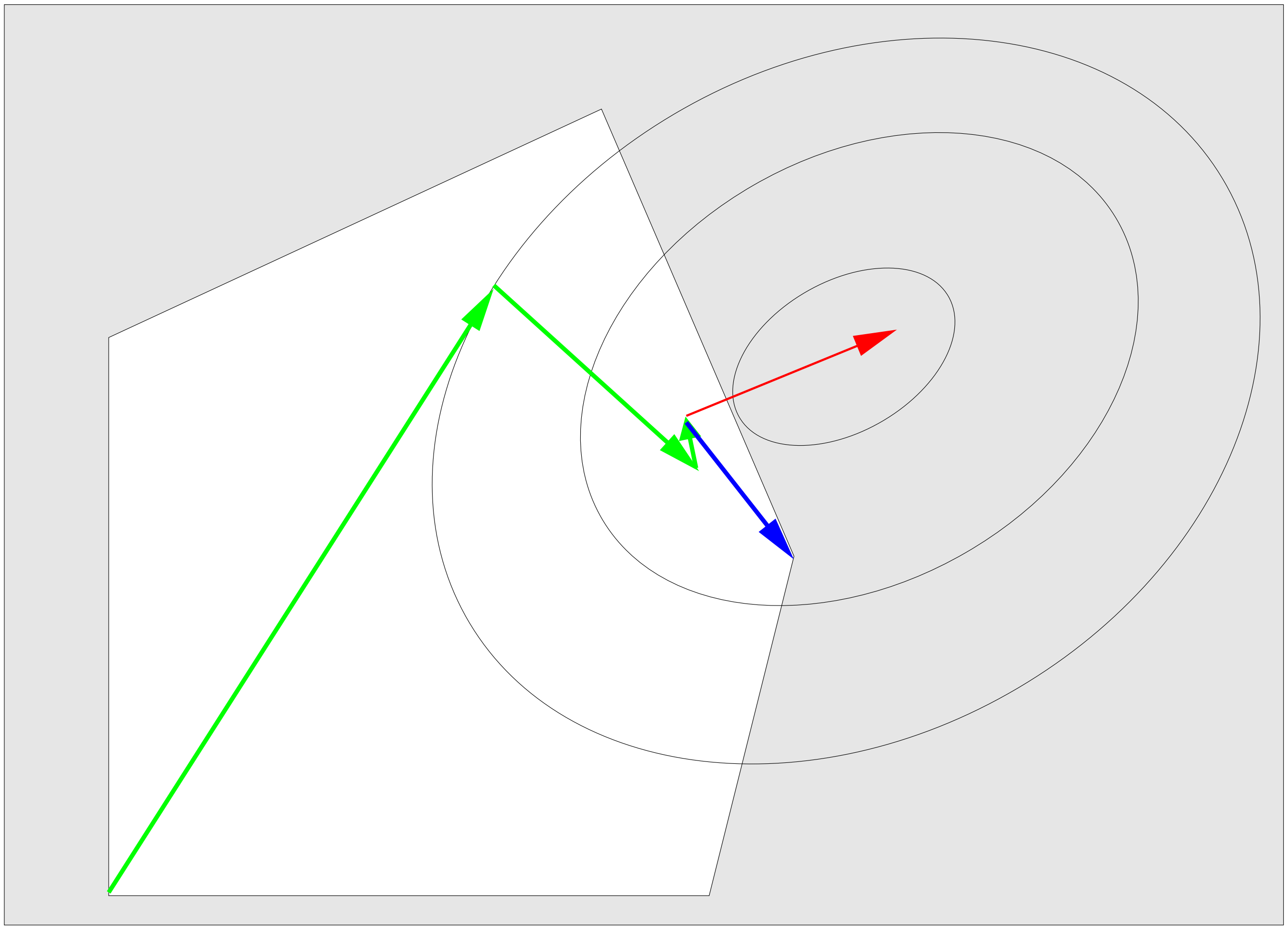
Conditional Gradient Ascent



Conditional Gradient Ascent



Conditional Gradient Ascent



Using inference to find update directions

- Solving the update direction $\max_{\boldsymbol{\mu} \in \mathcal{F}} (\boldsymbol{\ell} - K\boldsymbol{\mu}_0)^T \boldsymbol{\mu}$ with an LP solver will constitute a bottleneck for scalability
- By utilizing the hierarchical structure, we solve the problem efficiently
- **Theorem**: if $\boldsymbol{\mu}$ is a vertex of $\mathcal{F}$ there is a unique multilabel $\mathbf{y}$ that corresponds to that vertex.
- We can solve the update direction by finding multilabel $\mathbf{y}^*$ that maximizes the gradient
- Message-passing over the hierarchy T, dynamic programming implementaiton works in linear time.

Experiments

Datasets:

- Reuters Corpus Volume 1 ('CCAT' family), 34 microlabels, maximum tree depth 3, bag-of-words with TFIDF wieghting, 2500 documents were used for training and 5000 for testing.
- WIPO-alpha patent dataset (D section), 188 microlabels, maximum tree depth 4, 1372 documents for training, 358 for testing.

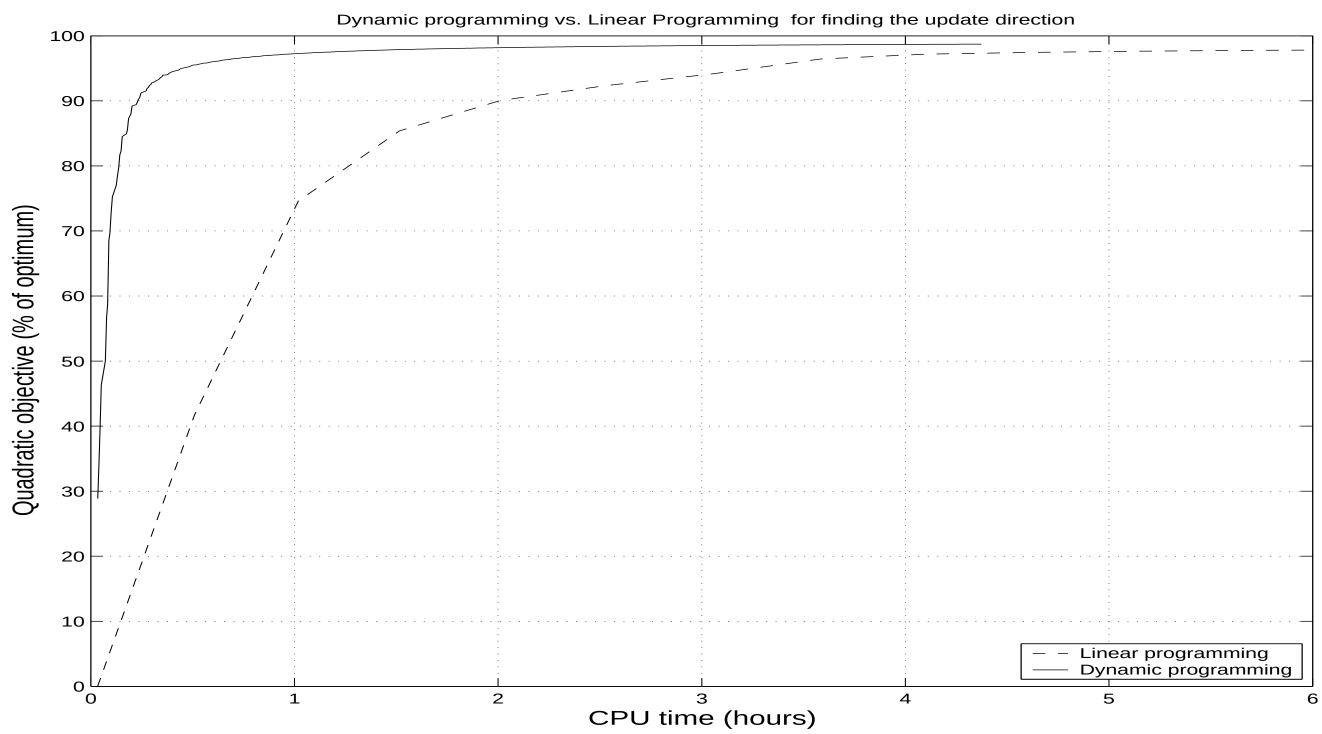
Algorithms:

- Our algorithm: H-M^3 ('Hierarchical Maximum Margin Markov')
- Comparison: Flat SVM, hierarchically trained SVM, hierarchical regularized least squares algorithm (Cesa-Bianchi et al. 2004)
- Implementation in MATLAB 7, LIPSOL solver used in the gradient ascent
- Tests run on a high-end Pentium PC with 1GB RAM

Optimization efficiency

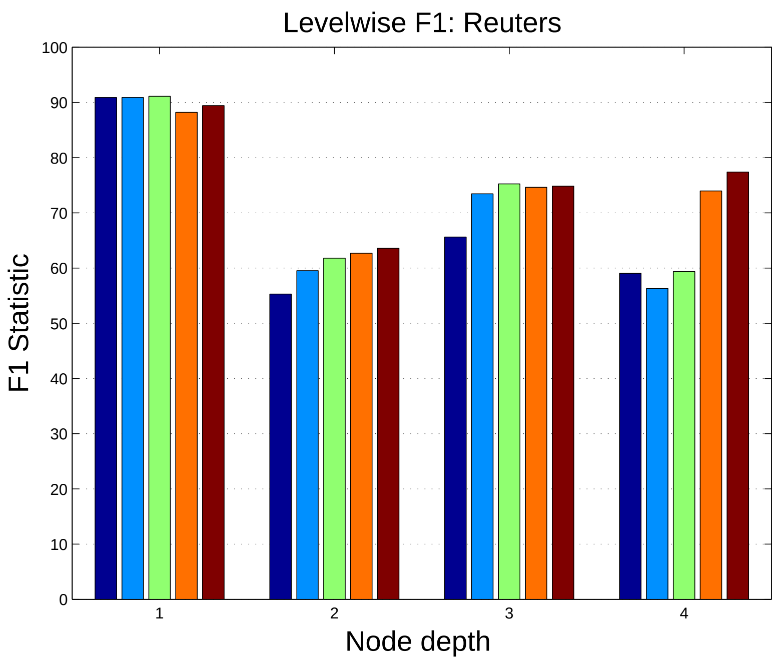
Optimization efficiency on WIPO dataset (1372 training examples, 188 microlabels) on a 3GHZ Pentium 4, 1GB main memory

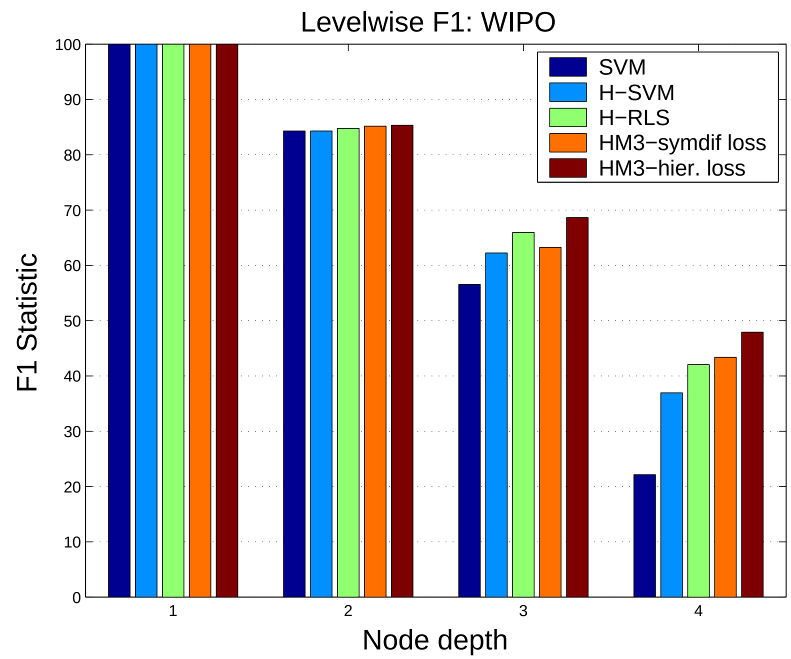
LP = update directions via linear programming DP = update directions via dynamic programmming



Prediction accuracy: Levelwise F1

F1 statistics computed for each node depth separately for Reuters (left) and WIPO (right)





Flat SVM is poor in recalling deep nodes, H-M^3-$\ell_{\tilde{H}}$ is the best prediction method in the leaves.

Scalability?

- Dual variables and the gradient require $O(m|E|)$ storage
- Kernel $K(x, x')$ requires $O(m^2)$ storage
- $\approx$ 10000 examples by 1000 microlabels fit to PC main memory, 100000 examples by 10000 microlabels will take up 100Gb hard disk!

Possibilities:

- Chunking to keep only a part of data in main memory at any given time
- Parallel implementation of conditional gradient algorithm is straight-forward.

Conclusions

- Kernel-based approach for hierarchical text classification when documents can belong to more than one category at a time
- Improved prediction accuracy on deep hierarchies
- Tractable optimization via decomposition into single-example subproblems, incremental conditional gradient search, and efficient inference algorithms to find update directions
- Tractable optimization for medium-sized datasets (thousands of examples $\times$ hundreds of microlabels)