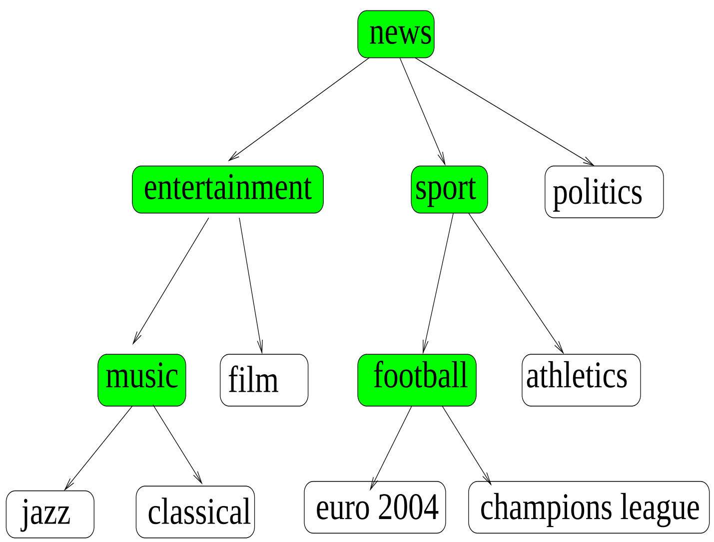
Kernel-based learning of hierarchial multilabel classification models

Juho Rousu
Department of Computer Science
Royal Holloway University of London, UK

Craig Saunders, Sandor Szedmak and John Shawe-Taylor
Electronics and Computer Science
University of Southampton, UK

Hierarchical Multilabel Classification: union of partial paths model

Goal: Learn to classify documents with respect to a classification hierarchy when a document can belong to more than one class at a time.



BBC News | ENTERTAINMENT | Football pundit accuses Posh

Front Page
World
UK
UK Politics
Business
Sci/Tech
Health
Education
Sport
Entertainment
New Music
Releases
Talking Point
In Depth
AudioVideo

Saturday, 8 January, 2000, 15:02 GMT

Football pundit accuses Posh

David and Victoria Beckham are permanently in the public eye

"No arrests were made because there was no written evidence"

"He lives a kind of pop star life"

BBC football pundit Mark Lawrenson has accused David Beckham and his pop star wife Victoria of 'courting publicity'.

SPECIAL REPORT World Club Championships

Lawrenson, an analyst on BBC1's Football Focus, spoke out during a discussion about Beckham's sending off in Thursday's World Club Championship match.

How to learn hierarchical multilabels?

Two simple strategies, both based on putting a classifier onto each internal node of the tree

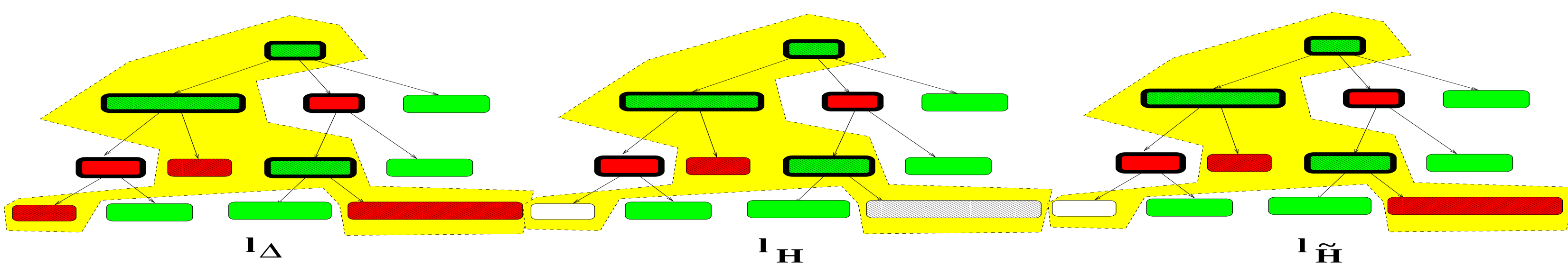
- Flatten the hierarchy: Decompose the multilabel into a set of binary classification problems which are learned independently. This approach does not utilize dependencies between the microlabels.
- Hierarchical training: Train a node j with examples $(x, \mathbf{y})$ that belong to the parent, i.e. $y_{pa(j)} = 1$. This approach utilizes some of the dependencies. However, it is not explicitly trained in terms of a loss function for the hierarchy.

We wish to improve on these approaches...

How to measure loss?

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
- **Symmetric difference loss**: $\ell_\Delta(\mathbf{y}, \hat{\mathbf{y}}) = \sum_j [\![y_j \neq \hat{y}_j]\!]$; counts incorrect microlabels.
- **Hierarchical 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
- **Simplified hierarchical 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.



Scaling the loss

It may also make sense to penalize mistakes made deep in the tree less than mistakes near the root. Two possible ways:

- Divide parent's scaling coefficent $c_{pa(j)}$ equally among the children (Cesa-Bianchi et al. 2004):

$$c_{root} = 1, c_j = c_{pa(j)}/|children(pa(j))|, j \neq root$$

- Scaling by the size of the subtree rooted by j:

$$c_{root} = 1, c_j = |T(j)|/|T(root)|, j \neq root$$

Coupled with the latter the hierarchical loss ℓ_H amounts to the proportion of the tree *not* reachable from the root when we stop before the first mistake along each path.

The classification model

We follow the approach of Hofmann et al. (2003) and Taskar et al. (2003).

Make the hierarchy a graphical model (Markov Tree) $T = (V, E)$ 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

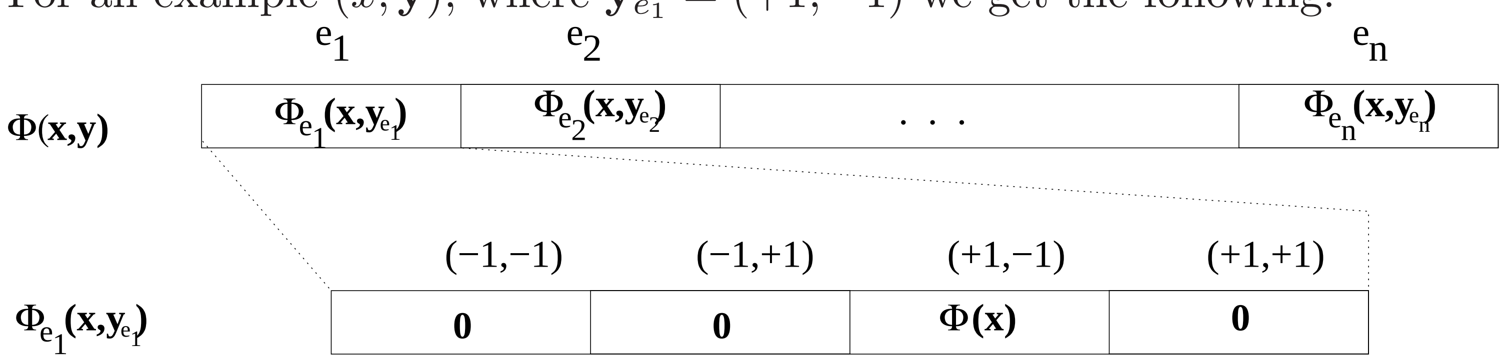
The joint feature vector $\boldsymbol{\phi}(x, \mathbf{y})$ is composed of blocks

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

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.
- In evaluating the kernel we can benefit from the special structure repeating $\boldsymbol{\phi}(x)$; the kernel does not need to be explicitly represented, which saves memory.

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



From Maximum Likelihood to Maximum Margin

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

To find the maximum likelihood assignment $\mathbf{w}* = \text{argmax}_w P(\mathbf{y}|x_i, \mathbf{w})$ we would need to compute the partition function $Z(x, \mathbf{w})$, but this is hard.

Examining the ratios of probabilities cancels out the partition function

$$\frac{P(\mathbf{y}_i|x_i, \mathbf{w})}{P(\mathbf{y}|x_i, \mathbf{w})} = exp\left(\mathbf{w}^T \boldsymbol{\phi}(x_i, \mathbf{y}_i) - \mathbf{w}^T \boldsymbol{\phi}(x_i, \mathbf{y}_i)\right)$$

Maximizing the ratio over all incorrect *pseudo-examples* $(x_i, \mathbf{y}) \neq (x_i, \mathbf{y}_i)$ is equivalent of maximizing the minimum margin

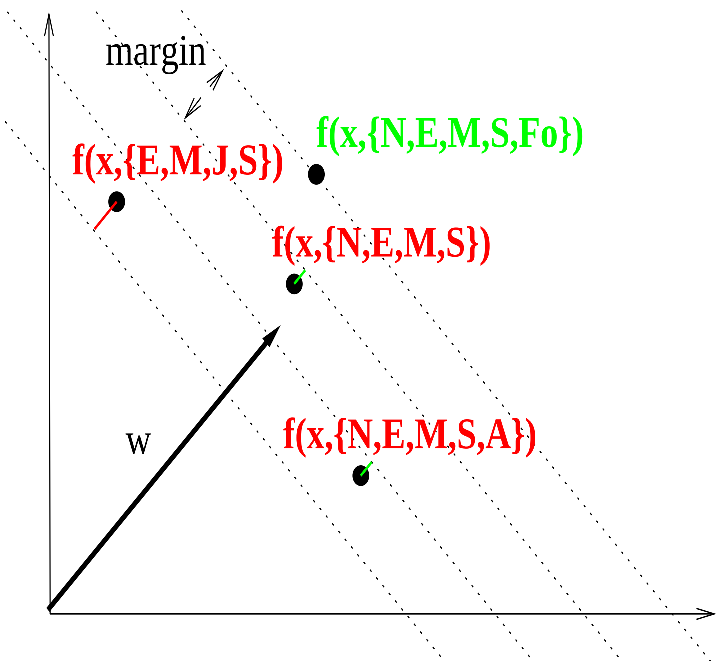
$$\text{argmax}_{\mathbf{w}} \min_{x_i, \mathbf{y} \neq \mathbf{y}_i} \mathbf{w}^T \Delta\boldsymbol{\phi}(x_i, \mathbf{y}) = \text{argmax}_{\mathbf{w}} \min_{x_i, \mathbf{y} \neq \mathbf{y}_i} \mathbf{w}^T \boldsymbol{\phi}(x_i, \mathbf{y}_i) - \mathbf{w}^T \boldsymbol{\phi}(x_i, \mathbf{y})$$

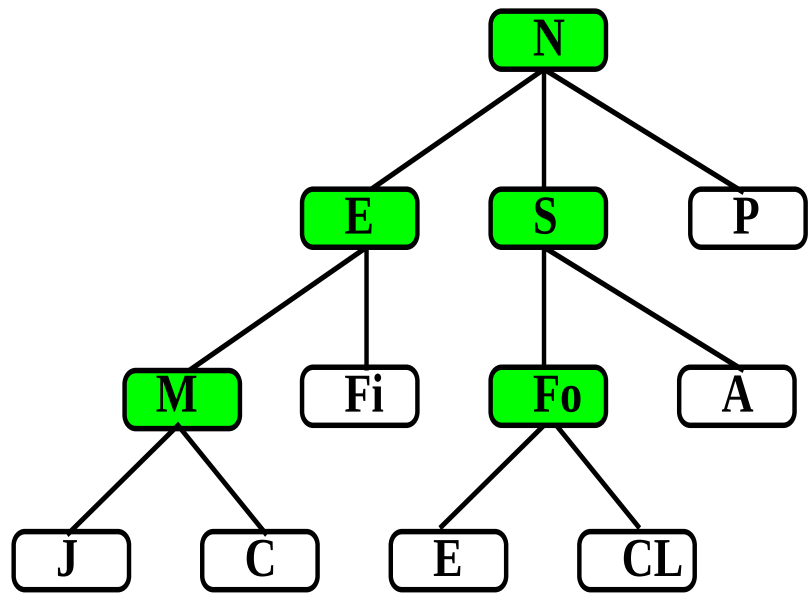
Scaling the margin

In general, we would like to push high-loss pseudo-examples far from the correct pseudo-example while allowing nearly-correct pseudo-examples get closer.

Also it is useful to allow slack for the examples.

If we insist the margin to be proportional to the loss, we produce a grading of the feature space:





Primal optimization problem

The margin maximization problem can be written as

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

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

- $\Delta\boldsymbol{\phi}(\mathbf{x}_i,\mathbf{y}) = \boldsymbol{\phi}(\mathbf{x}_i,\mathbf{y}_i) - \boldsymbol{\phi}(\mathbf{x}_i,\mathbf{y})$ is the different of the feature vectors,
- ξ_i is the slack alloted to example x_i, same slack is used for each pseudo-example $(x_i,\mathbf{y})$.
- exponential number of constraints in the number of microlabels!

Dual problem

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

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

- a dual variable $\alpha(x_i,\mathbf{y})$ for each pseudo-example $(x_i,\mathbf{y})$ corresponding to the primal constraints.
- the kernel contains an entry for each pair of pseudo-examples: $K(x_i,\mathbf{y};x_i',\mathbf{y}') = \Delta\boldsymbol{\phi}(\mathbf{x}_i,\mathbf{y})^T\Delta\boldsymbol{\phi}(\mathbf{x}_i',\mathbf{y}')$.
- one box constraint per training example

Solving the optimization problem

Problem: Exponential number (in the length of $\mathbf{y}$) of variables in the dual (contraints in primal)

For example, datasets we'll experiment with: Reuters - 2500×2^{34}, WIPO-alpha - 1372×2^{188})

Ways to avoid solving the full problem:

- Working set approaches (Hofmann et al.): solve the problem for a subset of exanples, incrementally add misclassified pseudo-examples . Polynomial number of pseudo-examples sufiices for an approximate solution.
- Marginalization of the dual variables (Taskar et al.): transform the problem into a polynomial-sized one by utilizing the Markov structure

Our approach is a variant of the latter.

Marginalizing the problem

We want to express the optimisation problem in terms of marginal dual variables:

$$\mu_e(x_i, \mathbf{y}_e) = \sum_{\{\mathbf{u}|\mathbf{u}_e = \mathbf{y}_e\}} \alpha(x_i, \mathbf{u}).$$

Need to express the kernel, loss, and constraints in terms of the edges:

$$K(x_i, \mathbf{y}; x_i', \mathbf{y}') = \sum_{e \in E} \Delta\boldsymbol{\phi}_e(x_i, \mathbf{y}_e)^T \Delta\boldsymbol{\phi}_e(x_i', \mathbf{y}_e') = \sum_{e \in E} K_e(x_i, \mathbf{y}_e; x_i', \mathbf{y}_e'),$$

The losses ℓ_Δ and $\ell_{\tilde{H}}$ (but not $\ell_{0/1}$ or ℓ_H) can be expressed as a sum of edge-wise losses

$$\ell(\mathbf{y}, \mathbf{y}') = \sum_e \ell_e(\mathbf{y}_e, \mathbf{y}_e').$$

The box constraints get the form

$$\sum_{\mathbf{u}_e} \mu_e(x_i, \mathbf{u}_e) \leq C, \forall i, e \in E.$$

Ensuring marginal consistency

We need to ensure that the marginal dual variables $\mu_e(x_i, \mathbf{y}_e)$ correspond to a valid $\alpha(x_i, \mathbf{y})$. That is, the marginals should lie on the marginal polytope of the hierarchy T.

If two edges share a node j, it is necessary and sufficient to have equal node-marginals μ_j:

$$\sum_{y'} \mu_e(x_i, y, y') = \mu_j(x_i, y) = \sum_{y'} \mu_{e'}(x_i, y, y').$$

We can achieve local consistency by pairing up each edge with its parent.

Marginalized problem

$$\max_{\boldsymbol{\mu}>0} \sum_{e\in E}\sum_{x_i,\mathbf{y}_e} \mu_e(x_i,\mathbf{y}_e)^T \boldsymbol{\ell}_e(x_i,\mathbf{y}_e) - \frac{1}{2}\sum_{e\in E}\sum_{x_i,\mathbf{y}_e}\sum_{x'_i,\mathbf{y}'_e} \mu_e(x_i,\mathbf{y}_e)^T K_e(x_i,\mathbf{y}_e;x'_i,\mathbf{y}'_e)\mu_e(x'_i,\mathbf{y}'_e)$$

$$\text{s.t} \sum_{y,y'} \mu_e(i,y,y') \leq C, \forall i, e\in E,$$

$$\sum_{y'} \mu_e(i,y',y) = \sum_{y'} \mu_{e'}(i,y,y'),\ \forall i, \forall y, (e,e') : e = pa(e'),$$

This problem is considerably smaller than the original: e.g. on Reuters data we have ca. 330000 marginal dual variables, in WIPO-alpha sligthly over one million.

But this is still too large to solve with off-the-shelf QP methods.

Decomposing the problem

The optimization problem has some structure:

- The constraints leave different x:s independent, but tie edges together
- The objective decomposes by the edges, but ties examples together

A gradient-based approach lets us decompose the problem by the examples.

Let us use the shorthands $\boldsymbol{\mu}_i = (\mu_e(x_i, \mathbf{u}_e))_{e,\mathbf{u}_e}$, $\boldsymbol{\ell}_i = (\ell_e(x_i, \mathbf{y}_e))_{e,\mathbf{y}_e}$, $K_{ij} = (K_e(x_i, \mathbf{y}_e; x_j, \mathbf{y}'_e))_{e,\mathbf{y}_e,\mathbf{y}'_e}$.

When starting solving the subproblem for x_i, we need to obtain initial gradient $\mathbf{g}_i = \boldsymbol{\ell}_i - \sum_j K_{ij}\boldsymbol{\mu}_j$. This involves all active marginal dual variables and a corresponding slice of the kernel matrix

When updating $\boldsymbol{\mu}_i$ only, gradient update is much cheaper as it only involves the kernel block K_{ii}: $\Delta\mathbf{g}_i = -K_i i \Delta\boldsymbol{\mu}_i$

The optimization algorithm

The main idea of the algorithm is the following:

1. Obtain a working set of examples
2. Make one optimization pass over examples x_i in working set:
   (a) Obtain an initial gradient for x_i.
   (b) Update $\boldsymbol{\mu}_i$ by making a few conditional gradient steps
3. Compute KKT conditions, slacks, and the duality gap
4. If duality gap small enough, stop, otherwise repeat from step 1.

Conditional Gradient Ascent

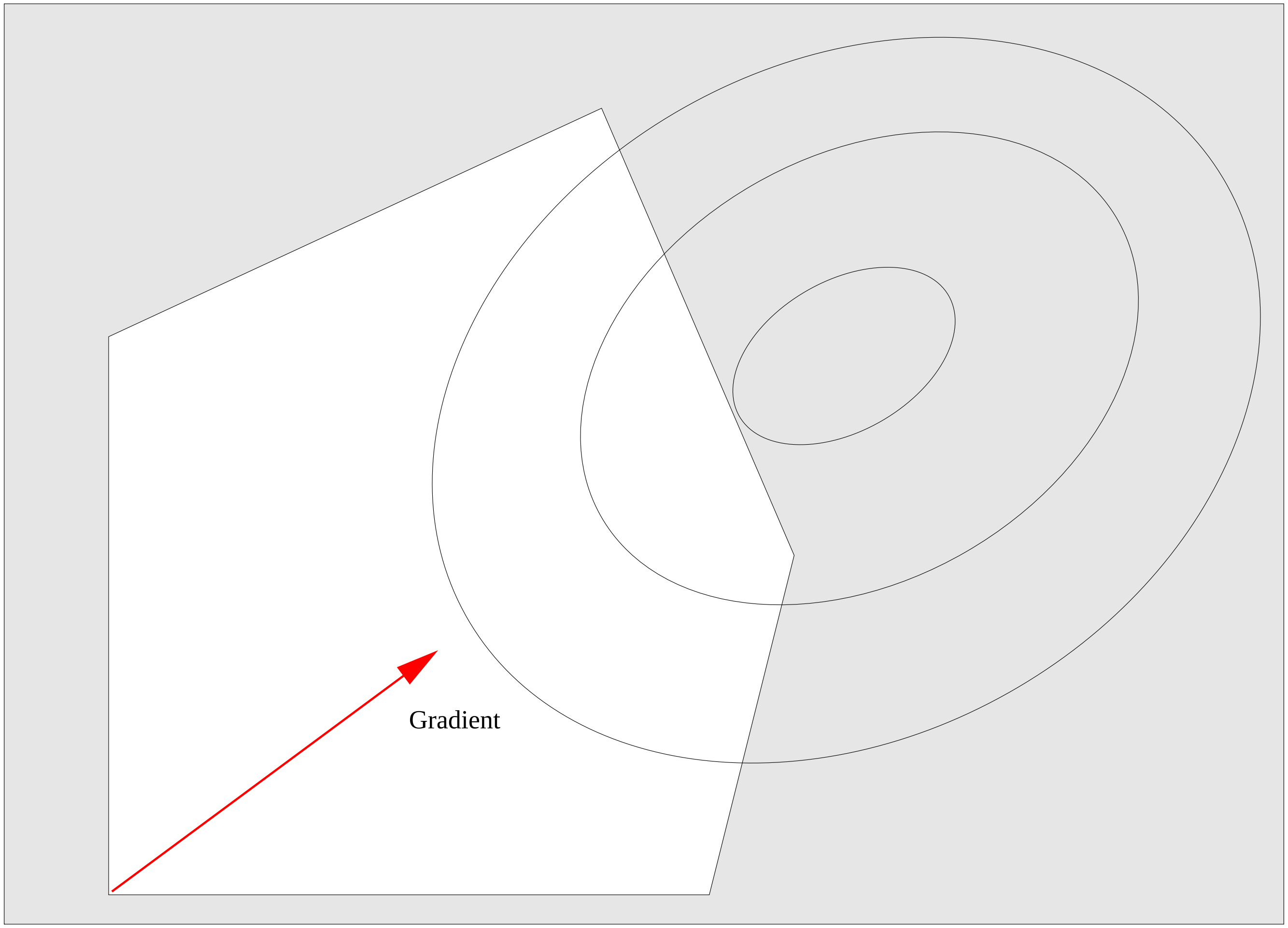
To update $\boldsymbol{\mu}_i$ we use a variant of conditional gradient ascent.

We iterate the following:

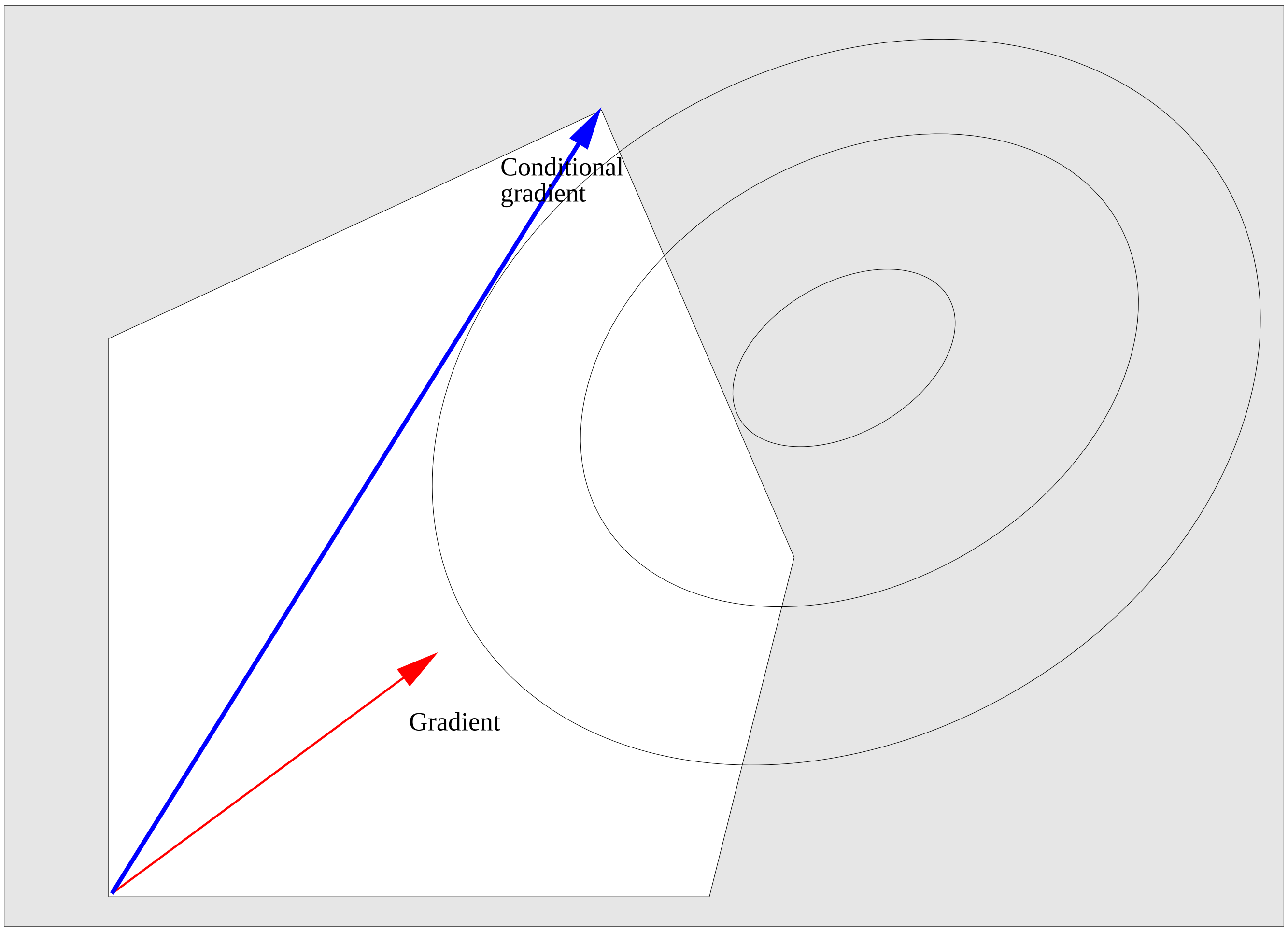
1. Find the best feasible point $\boldsymbol{\mu}'_i$ with respect to the gradient $\mathbf{g}_i$. This requires solving a linear program $\boldsymbol{\mu}'_i = \text{argmax}_{\mathbf{z} \in \mathcal{F}} \mathbf{g}_i^T \mathbf{z}$. If $\boldsymbol{\mu}' = \boldsymbol{\mu}$ we have the optimum and can stop.
2. Find a saddle point $\boldsymbol{\mu}_*$ of the quadratic objective along the ray $\boldsymbol{\mu}_i + a(\boldsymbol{\mu}'_i - \boldsymbol{\mu}), a > 0$.
3. Update $\boldsymbol{\mu}_i = \boldsymbol{\mu}_*$ if saddle point feasible, otherwise update $\boldsymbol{\mu}_i = \boldsymbol{\mu}'_i$.

In step 1, we use MATLABs linear interior point solver. We typically make only a few iterations before moving onto the next example.

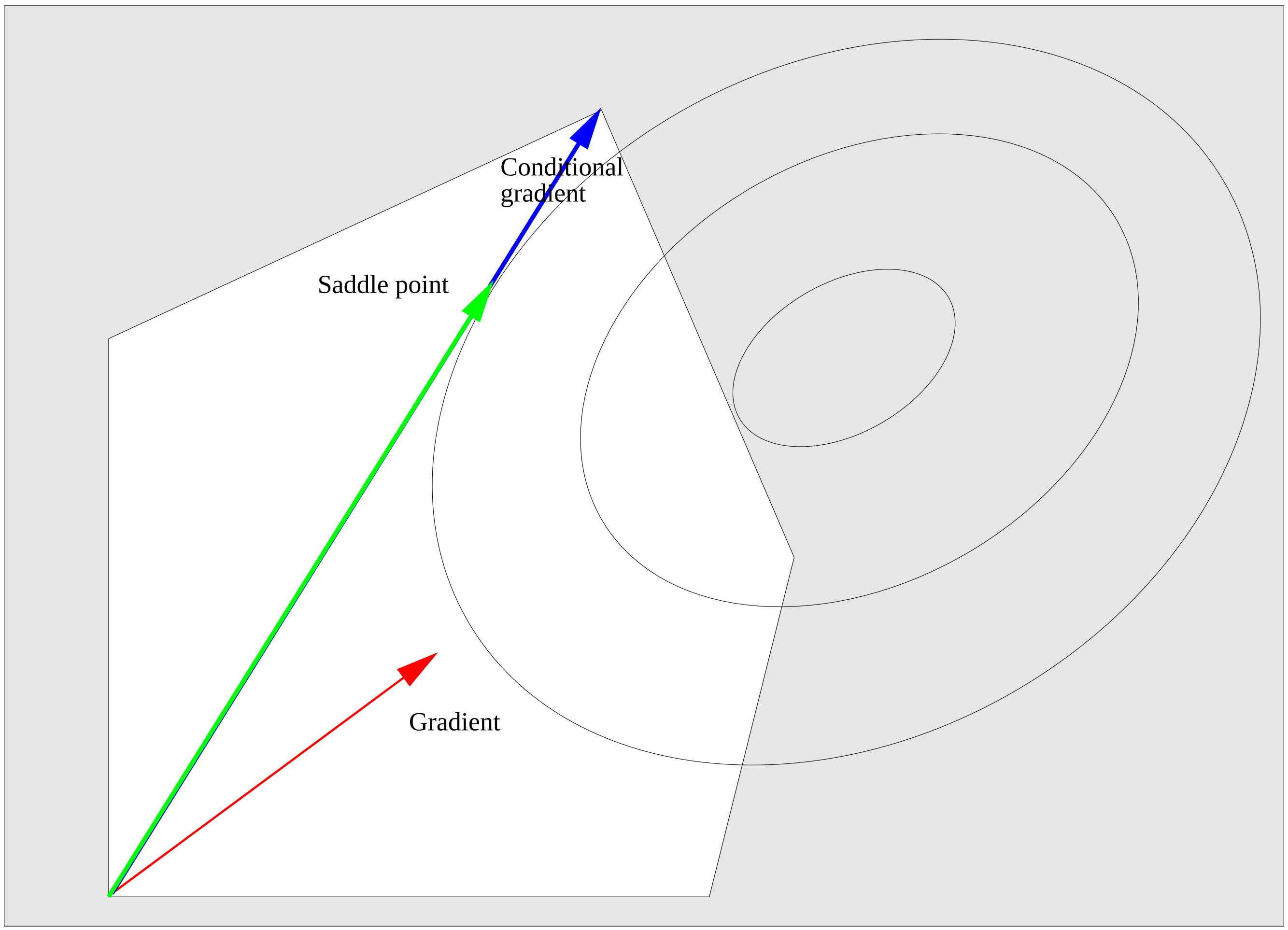
Conditional Gradient Ascent



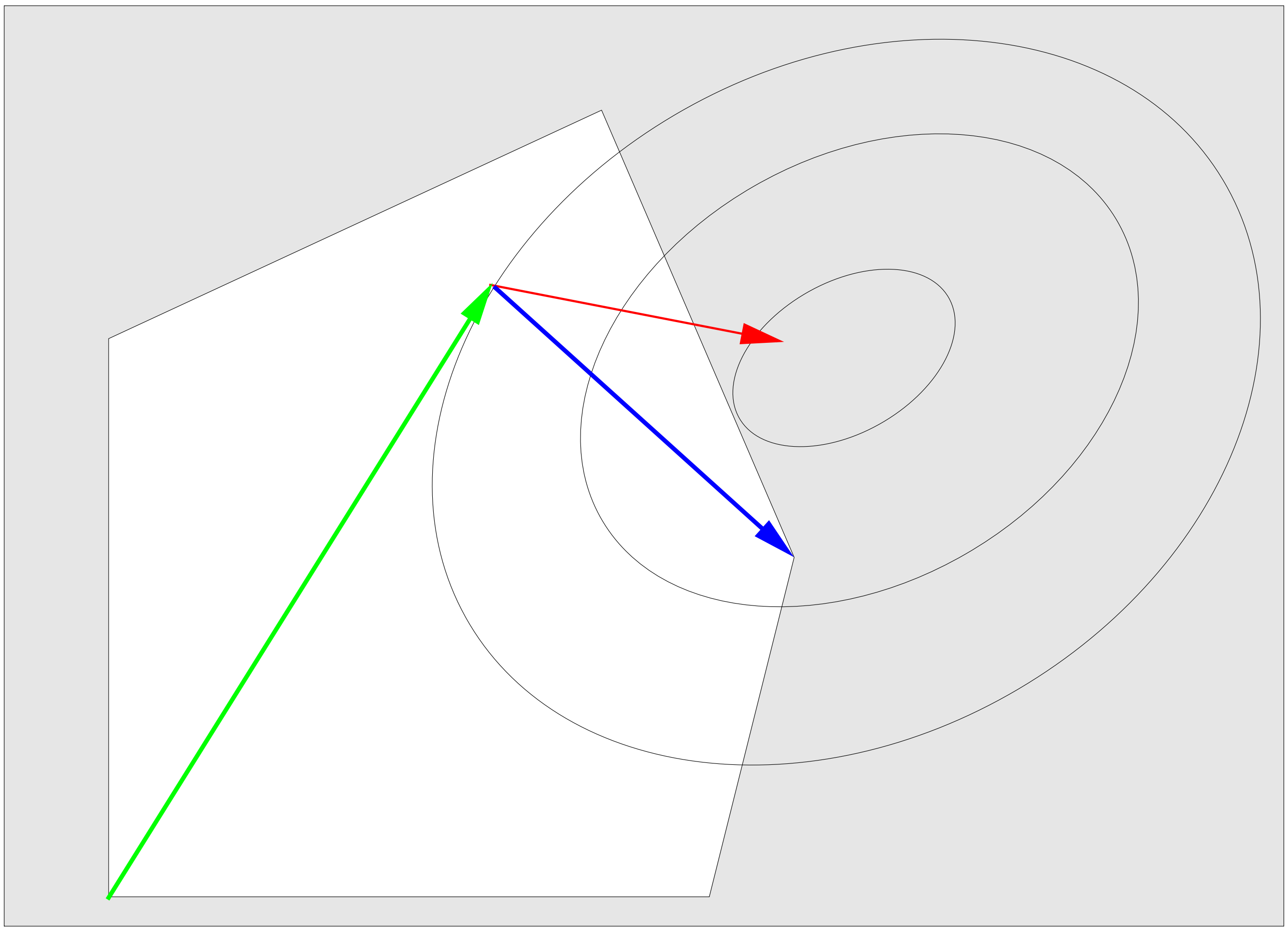
Conditional Gradient Ascent



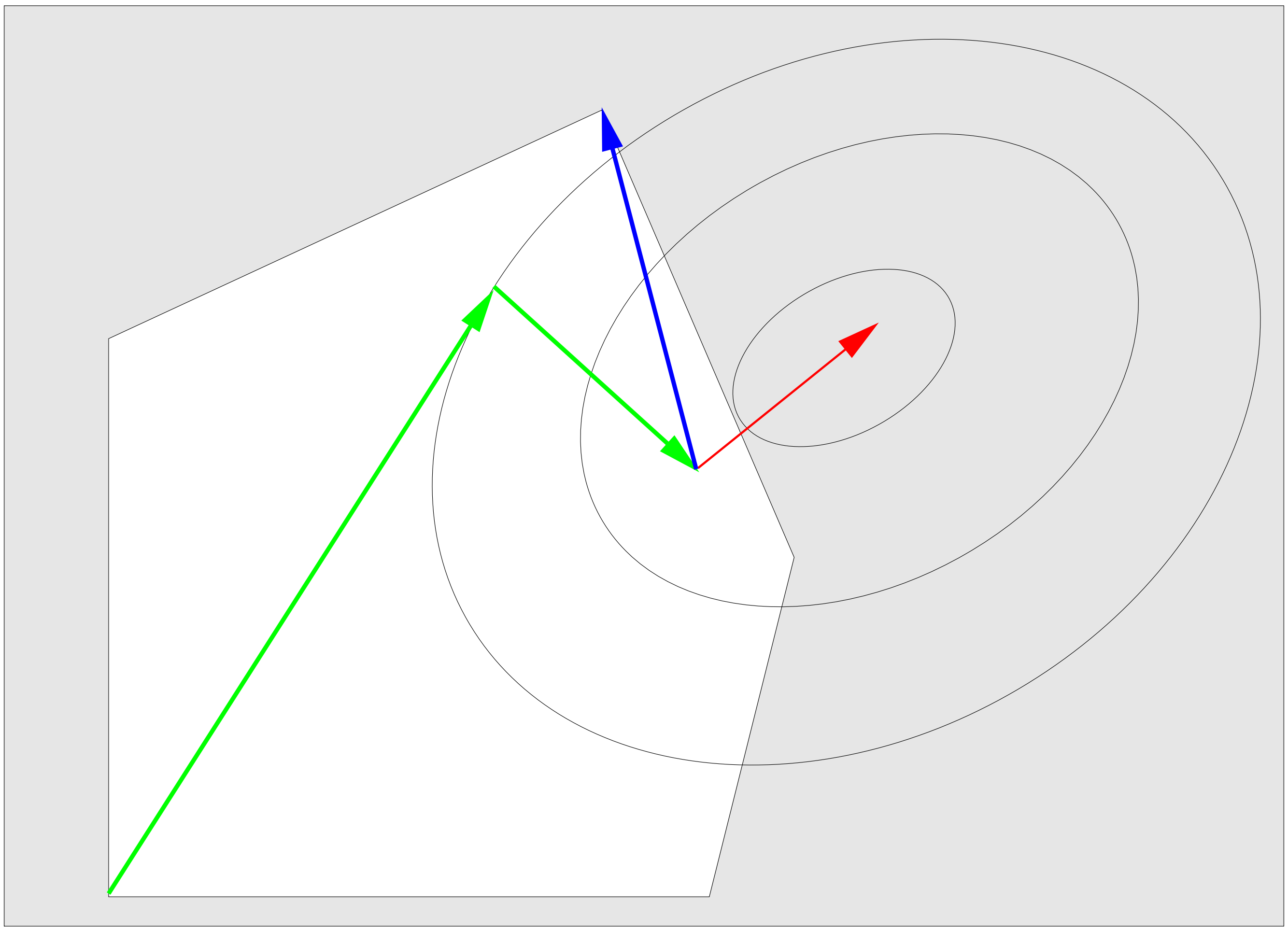
Conditional Gradient Ascent



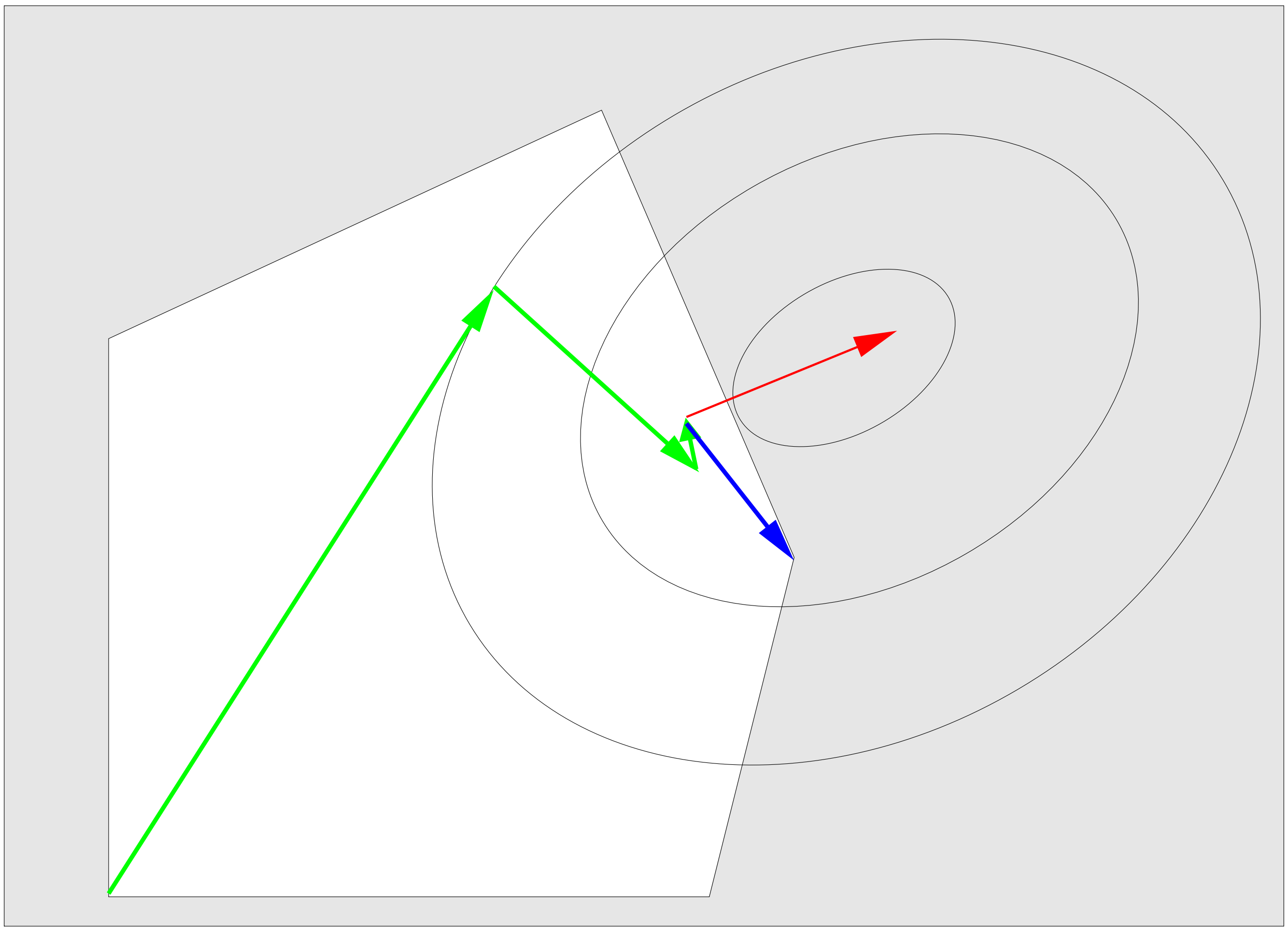
Conditional Gradient Ascent



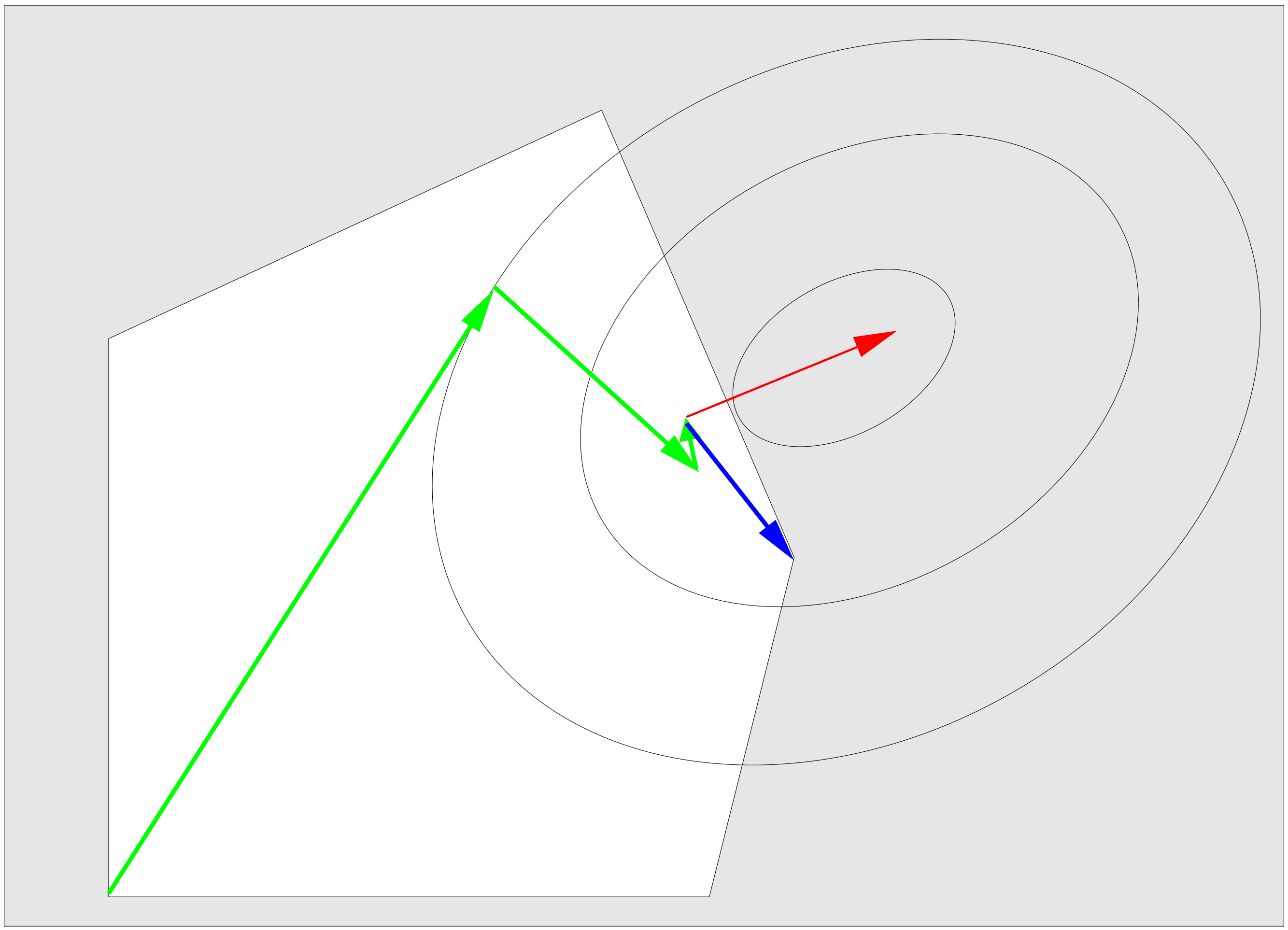
Conditional Gradient Ascent



Conditional Gradient Ascent



Conditional Gradient Ascent



Working set maintenance

- Observation: with problems of this kind, most of the training examples will end up being active at optimum; no use trying to keep working set small:
- We take every training example that is active or violates margins.
- But we try to work on more examples that contribute to duality gap a lot: give them more conditional gradient iterations.

Experiments

Datasets:

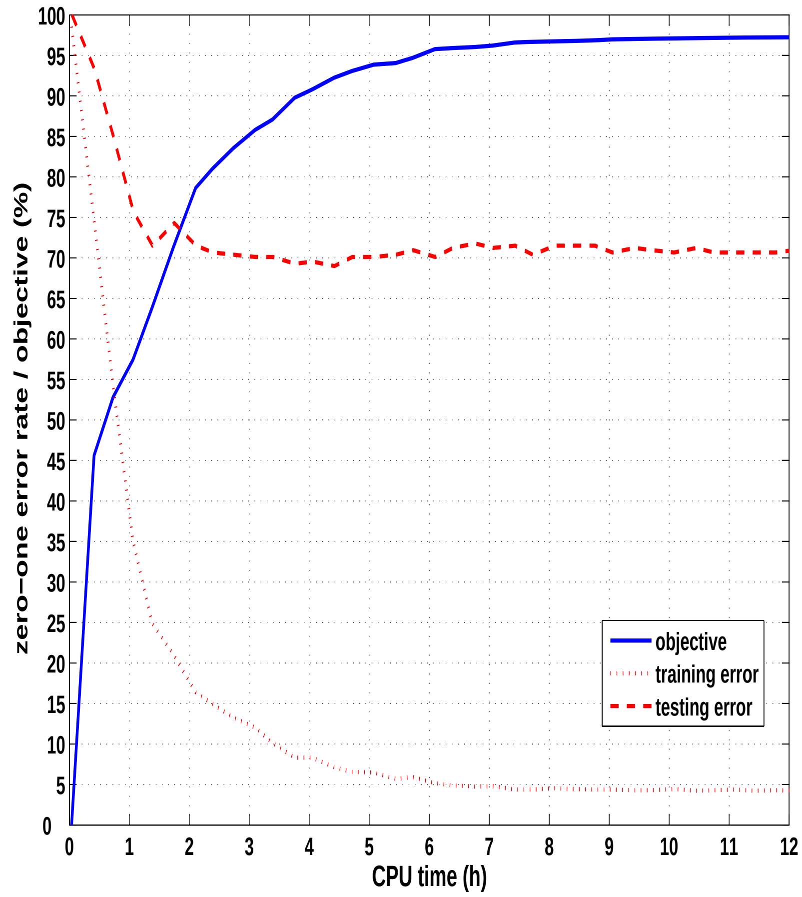
- Reuters Corpus Volume 1 ('CCAT' family), 34 microlabels, maximum tree depth 3, bag-of-words with TFIDF wieghting, 2500 documents were used for training and 5000 for testing.
- WIPO-alpha patent dataset (D section), 188 microlabels, maximum tree depth 4, 1372 documents for training, 358 for testing.

Algorithms:

- Our algorithm: H-M^3 ('Hierarchical Maximum Margin Markov Networks')
- 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

Example learning curve

- WIPO-alpha,training with $\ell_{\tilde{H}}$-loss
- Optimizing 10^6+ marginal dual variables, majority are non-zero at the optimum
- Error rates at bottom out faster than objective, early stopping is tempting
- No significant overfitting observed



Results

Table 1: Prediction losses obtained using different learning algorithms on Reuter's (left) and WIPO-alpha data (right). The loss $\ell_{0/1}$ is given as a percentage, the other losses as averages per-example.

Algorithm	$\ell_{0/1}$	ℓ_{Δ}	ℓ_H
SVM	32.9	0.611	0.099
H-SVM	29.8	0.570	0.097
H-RLS	28.1	0.554	0.095
H-M^3-ℓ_{Δ}	27.1	0.575	0.114
H-M^3-$\ell_{\tilde{H}}$	27.9	0.588	0.109

Algorithm	$\ell_{0/1}$	ℓ_{Δ}	ℓ_H
SVM	87.2	1.84	0.0532
H-SVM	76.2	1.74	0.0511
H-RLS	72.1	1.69	0.0495
H-M^3-ℓ_{Δ}	70.9	1.67	0.0504
H-M^3-$\ell_{\tilde{H}}$	65.0	1.73	0.0478

Conclusions

- We presented an kernel-based approach for hierarchical text classification when documents can belong to more than one category at a time
- Utilizing the dependency structure of microlabels in a Markovian way leads to improved prediction accuracy, especially on WIPO-alpha, where the hierarchy is deeper than Reuter's.
- Optimization is made feasible by utilizing decomposition of the original problem and making incremental conditional gradient search in the subproblems.