Structured Prediction

w/ Large Margin Methods

Thomas Hofmann

thomas.hofmann@ethz.ch

thofmann@google.com

Data Analytics Laboratory, ETH Zürich

& Google Engineering Center, Zürich

Machine Learning Summer School

Tübingen, September 3-4, 2013

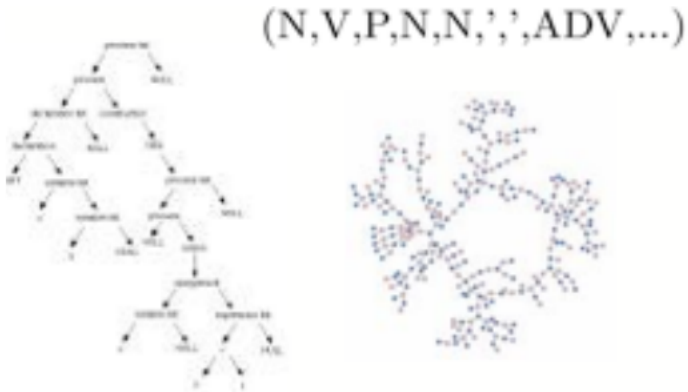
Section 1

Motivation & Overview

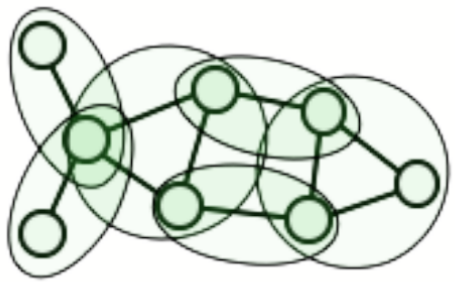
Structured Prediction

Generalize supervised machine learning methods to deal with structured outputs and/or with multiple, interdependent outputs.

Structured objects such as sequences, strings, trees, labeled graphs, lattices, etc.



Multiple response variables that are interdependent = collective classification



Jiggsaw Metaphor

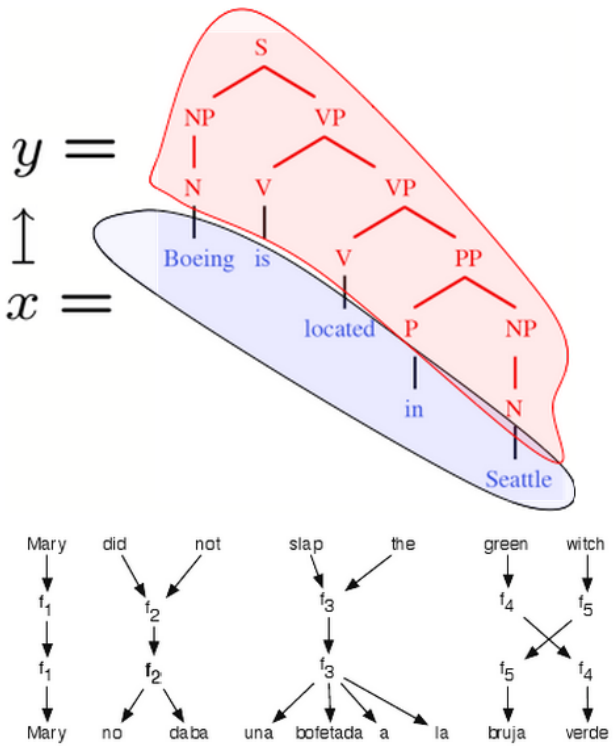
Holistic prediction $\neq$ independent prediction of pieces

It is not just about solving one instance of a puzzle, but learning how to solve a whole class of puzzles.

inspired by Ben Taskar's tutorial

Natural Language Processing

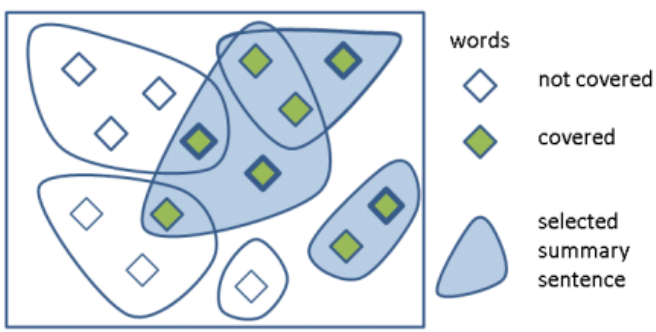
- PoS tagging, named entity detection, language modeling
- Syntactic sentence parsing, dependency parsing
- Semantic parsing



- B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning, Max-Margin Parsing, EMNLP, 2004
- R. McDonald, K. Crammer, F. Pereira, Online large-margin training of dependency parsers. ACL 2005
- H. C. Daume III, Practical Structured Learning Techniques for Natural Language Processing, Ph.D. Thesis, Univ. Southern California, 2006
- R. McDonald, K. Hannan, Kerry, T. Neylon, M. Wells, J. Reynar, Structured models for fine-to-coarse sentiment analysis, ACL 2007
- B. Roark, M. Saraclar, M. Collins: Discriminative n-gram language modeling. Computer Speech & Language 21(2): 373-392, 2007
- Y. Zhang, S. Clark, Syntactic processing using the generalized perceptron and beam search, Computational Linguistics 2011.
- C. Cherry, G. Foster, Batch tuning strategies for statistical machine translation, NAACL 2012.
- L. S. Zettlemoyer, M. Collins, Learning to Map Setences to Logical Form: Structured classification with probabilistic categorial grammars, arXiv, 2012

Information Retrieval

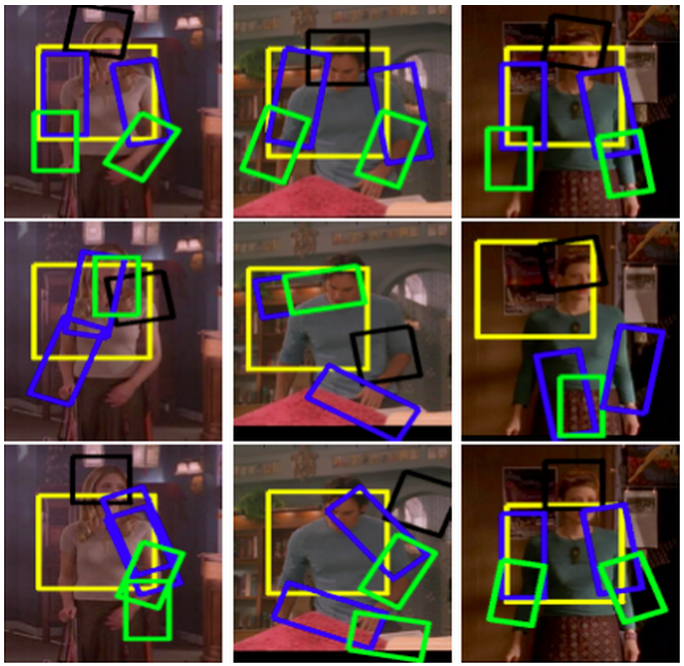
- Learning to rank, e.g. search engines
- Multidocument summarization
- Whole page clickthrough prediction
- Entity linking and reference resolution



- Y Yue, T. Finely, F. Radlinski, T. Joachims: A support vector method for optimizing average precision. SIGIR 2007
- L. Li et al: Enhancing diversity, coverage and balance for summarization through structure learning, WWW 2009.
- T. Berg-Kirkpatrick, Taylor, D. Gillick, D. Klein: Jointly Learning to Extract and Compress, ACL. 2011
- R. Sipos, P. Shivaswamy, T. Joachims: Large-margin learning of submodular summarization models, ACL 2012

Computer Vision

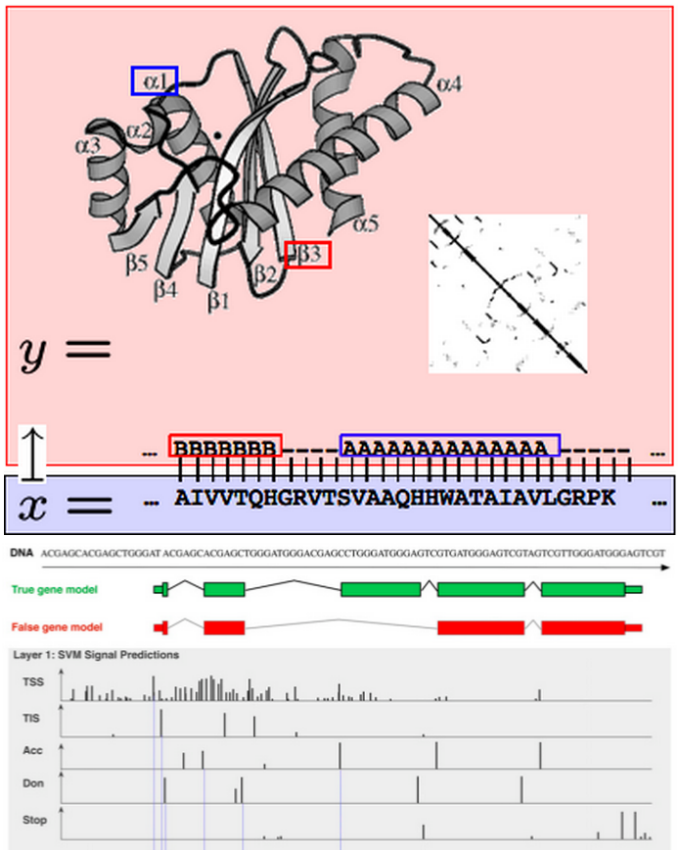
- Image segmentation
- Scene understanding
- Object localization & recognition



- T. Caetano & R. Hartley, ICCV 2009 Tutorial on Structured Prediction in Computer Vision
- M. P. Kumar, P. Torr, A. Zisserman, Efficient Discriminative Learning of Parts-based Models, ICCV 2009.
- C. H. Lampert, M. B. Blaschko, T. Hofmann: Efficient Subwindow Search: A Branch and Bound Framework for Object Localization, PAMI 2009
- A. G. Schwing, T. Hazan, M. Pollefeys, R. Urtasun: Efficient structured prediction for 3D indoor scene understanding, CVPR 2012
- A. Patron-Perez, M. Marszalek, I. Reid, A. Zisserman: Structured learning of human interactions in TV shows, PAMI 2012

Computational Biology

- Protein structure & function prediction
- Gene finding, structure prediction (splicing)



- Y. Liu, E. P. Xing, and J. Carbonell, Predicting protein folds with structural repeats using a chain graph model, ICML 2005
- G. Rätsch and S. Sonnenburg, Large Scale Hidden Semi-Markov SVMs, NIPS 2006.
- A. Sokolov and A. Ben-Hur, A Structured-Outputs Method for Prediction of Protein Function, In Proceedings of the 3rd International Workshop on Machine Learning in Systems Biology, 2008.
- K. Astikainen et al., Towards Structured Output Prediction of Enzyme Function, BMC Proceedings 2008, 2 (Suppl. 4):S2
- G. Schweikert et al, mGene: Accurate SVM-based gene finding with an application to nematode genomes, Genome Res. 2009 19: 2133-2143
- N. Görnitz, C. Widmer, G. Zeller, A. Kahles, S. Sonnenburg, G. Rätsch: Hierarchical Multitask Structured Output Learning for Large-scale Sequence Segmentation, NIPS 2011

Overview

1. $\Rightarrow$ Overview
2. Model
   - Structured prediction SVM
   - Margins & loss functions for structured prediction
3. Oracle-based Algorithms
   - Cutting plane methods
   - Subgradient-based approaches
   - Frank-Wolfe algorithm
   - Dual extragradient method
4. Decomposition-based Algorithms
   - Representer theorem and dual decomposition
   - Conditional random fields
   - Exponentiated gradient
5. Conclusion & Discussion

Section 2

Model

Structured Prediction

- Input space $\mathcal{X}$, output space $\mathcal{Y}$
- $|\mathcal{Y}| = m$ can be large due to combinatorics
  - e.g. label combinations, recursive structures
- Given training data $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \ldots, n$
  - drawn i.i.d. from unknown distribution $\mathcal{D}$
- Goal: find a mapping F

$$F : \mathcal{X} \to \mathcal{Y}$$

  - with a small prediction error

$$\text{err}(F) = \mathbf{E}_D \left[\triangle(Y, F(X)) \right]$$

  - relative to some loss function $\triangle : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_0^+$, with $\triangle(y, y) = 0$ and $\triangle(y, y') > 0$ for $y \neq y'$.

Examples: Loss Functions

- Multilabel prediction
  - $\mathcal{Y} = \{-1, 1\}^k$
  - $\triangle(y, y') = \frac{1}{2}(k - \langle y, y' \rangle)$ (Hamming loss)
- Taxonomy classification
  - $\mathcal{Y} = \{1, \ldots, k\}$, k classes arranged in a taxonomy
  - $\triangle(y, y') =$ tree distance between y and y'
  - cf. [CH04, BMK12]
- Syntactic parsing
  - $\mathcal{Y} = \{$labeled parse trees$\}$
  - $\triangle(y, y') = \#$ labeled spans on which y and y' do not agree
  - cf. [TKC+04]
- Learning to rank
  - $\mathcal{Y} = \{$permutations of set of items$\}$
  - $\triangle(y, y') =$ mean average precision of ranking y' vs. optimal y
  - cf. [YFRJ07]

Multiclass Prediction

- Apply standard multiclass approach to $\mathcal{Y}$ with $|\mathcal{Y}| = m$.
- Define $\mathcal{Y}$-family of discriminant functions $f_y : \mathcal{X} \to \mathbb{R}$, $y \in \mathcal{Y}$
- Prediction based on winner-takes-all rule

$$F(x) = \arg\max_{y \in \mathcal{Y}} f_y(x)$$

- Typical: linear discriminants with weight vector $w_y \in \mathbb{R}^d$ and

$$f_y(x) := \langle \phi(x), w_y \rangle$$

  - shared input representation via feature map $\phi : \mathcal{X} \to \mathbb{R}^d$

- Trained via one-vs-all or as a single 'machine'
- References: [RK04, WW99, CS02, LLW04]

Multiclass Prediction $\prec$ Structured Prediction

- What happens as $m > n$?
  - Not enough training data to even have a single example for every output.
- Taking outputs as atomic entities without any internal structure does not enable generalization across outputs
  - There is no learning, only memorization of outputs.
- Need to go beyond the standard multiclass setting and enable learning across $\mathcal{X} \times \mathcal{Y}$. Two lines of thought:
- Feature-based Prediction: extract features from inputs & outputs, define discriminant functions with those features
- Factor-based Prediction: decompose output space into variables and identify factors [*coming back to this later*]

Feature-based Prediction

- Joint feature maps

$$\psi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d, \quad k_\psi((x,y),(x'y')) := \langle \psi(x,y), \psi(x',y') \rangle$$

  to extract features from input-output pairs.

- Canonical construction by crossing features extracted separately from inputs and outputs

$$\psi = \phi_\mathcal{X} \times \phi_\mathcal{Y} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^{d_x \cdot d_y}, \quad \psi(x,y) := \phi_\mathcal{X}(x) \times \phi_\mathcal{Y}(y).$$

  - Can be more selective about features crossed (subsets).
  - Other constructions (beyond crossing) are possible.

- When using inner products one gets the compelling factorization

$$k_\psi((x,y),(x',y')) = k_{\phi\mathcal{X}}(x,x') \cdot k_{\phi\mathcal{Y}}(y,y').$$

Example: Label Sequence (HMM) [ATH+03]

- Hidden Markov Models: $\mathcal{X} = (\mathbb{R}^d)^l$, $\mathcal{Y} = \{1, \ldots, k\}^l$, where
  - l : length of sequence
  - k : cardinality of hidden variable
  - d : dimensionality of observations
- First feature template: local observations

$$\psi^1_c(x, y) = \sum_{t=1}^{l} \mathbf{1}[y^t = c] \cdot \phi(x^t)$$

  - adding up all observations that are assigned to same class $c \in \{1, \ldots k\}$
- Second feature template: pairwise nearest neighbor interactions

$$\psi^2_{c,\bar{c}}(x, y) = \sum_{t=1}^{l-1} \mathbf{1}[y^t = c] \cdot \mathbf{1}[y^{t+1} = \bar{c}]$$

  - counting number of times labels $(c, \bar{c})$ are neighbors

Example: Optimizing Ranking [YFRJ07]

- Kandall's tau:

$$\tau = \frac{\#\text{ concord. pairs - }\#\text{ discord. pairs}}{\#\text{ all pairs}} = 1 - \frac{2 \cdot \#\text{ discordant pairs}}{\#\text{ all pairs}}$$

- Output ranking encoded via pairwise ordering

$$\mathcal{Y} = \{-1, 1\}^{k \times k}, \quad y_{ij}^{\prec} = \begin{cases} 1 & \text{if } i \prec j \\ -1 & \text{otherwise} \end{cases}$$

- Combined feature function

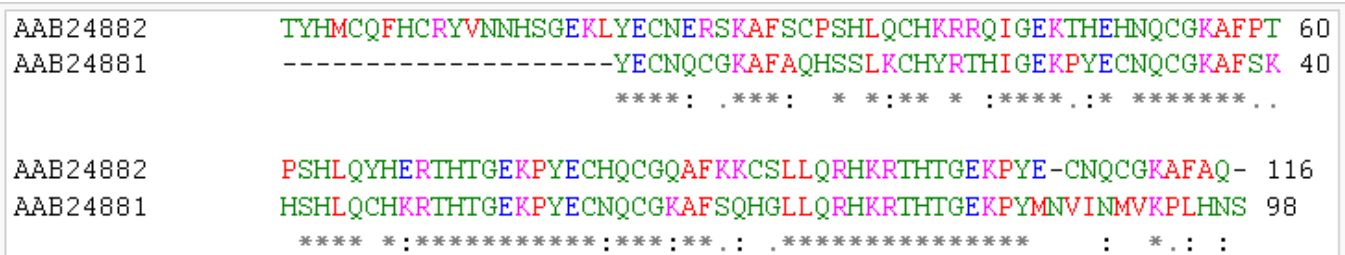
$$\psi(x, y) = \sum_{i<j} y_{ij}[\phi(x_i) - \phi(x_j)]$$

Bipartite case $C^+(x) \cup C^-(x) = \{1, \ldots, k\}$, relev./non-relev. items

$$\psi(x, y) = \sum_{i \in C^+(x)} \sum_{j \in C^-(x)} y_{ij}[\phi(x_i) - \phi(x_j)]$$

Example: Learning Alignments [JHYY09]

- Input: two annotated (protein) sequences $x = (s_a, s_b)$.
- Output: alignment y between two sequences x
- Joint features:



A sequence alignment, produced by ClustalW, of two human zinc finger proteins, identified on the left by GenBank accession number.
Key: Single letters: amino acids. Red: small, hydrophobic, aromatic, not Y. Blue: acidic. Magenta: basic. Green: hydroxyl, amine, amide, basic. Gray: others. "*": identical. ":": conserved substitutions (same colour group). ".": semi-conserved substitution (similar shapes).[2]

- Types of features: combinations of amino acid, secondary structure, solvent accessibility; sliding window; PSI-BLAST profile scores;

Multiclass + Output Features = Structured Predction

- Generalize multiclass prediction and define linear discriminants

$$\text{multiclass} \rightarrow f_y(x; w) := f(x, y; w) = \langle \psi(x, y), w \rangle \leftarrow \text{structured}$$

- Parameter sharing across outputs (with same features)
- Recover feature-less multiclass by defining (1 out of m encoding)

$$\langle \phi_{\mathcal{Y}}(y), \phi_{\mathcal{Y}}(y') \rangle = \delta_{yy'}$$

  i.e. feature vectors involving different classes y, y' are orthogonal.
- Allows to incorporate prior knowledge into multiclass problems
  - Hierarchical classification - encode class taxonomy []
  - Entity reference resolution - encode prior entity names and types
- Requires single 'machine' formulation as weight vectors are not separated $\rightarrow$ How can we generalize SVMs?

Binary Support Vector Machine

Convex Quadratic Program (primal)

$$(w^*, \xi^*) = \arg\min_{w,\xi \geq 0} \mathcal{H}(w, \xi) := \frac{\lambda}{2}\langle w, w\rangle + \frac{1}{n}\|\xi\|_1$$
$$\text{subject to} \quad y_i\langle w, \phi(x_i)\rangle \geq 1 - \xi_i \quad (\forall i)$$

- Examples $(x_i, y_i) \in \mathcal{X} \times \{-1, 1\}$, $i = 1, \ldots, n$
- Feature map $\phi : \mathcal{X} \to \mathbb{R}^d$
- Weight vector $w \in \mathbb{R}^d$
- Slack variables $\xi_i \geq 0$
- Regularization parameter $\lambda \in \mathbb{R}^+$

Margin-rescaled Constraints

For each instance (x_i, y_i) define $m := |\mathcal{Y}|$ constraints via

$$f(x_i, y_i; w) - f(x_i, y; w) \geq \triangle(y_i, y) - \xi_i \quad (\forall y \in \mathcal{Y})$$

- Require correct output y_i to be scored higher than all incorrect outputs $y \neq y_i$ by a margin
- Adjust target margin for incorrect outputs to be $\triangle(y_i, y)$
- Provides an upper bound on the empirical loss via

$$\begin{aligned} \xi_i^* &= \max_y \{\triangle(y_i, y) - [f(x_i, y_i; w) - f(x_i, y; w)] \\ &\geq \triangle(y_i, \hat{y}) - \underbrace{[f(x_i, y_i; w) - f(x_i, \hat{y}; w)]}_{\leq 0 \text{ for } \hat{y}} \geq \triangle(y_i, \hat{y}) \end{aligned}$$

where $\hat{y}_i := \arg\max_y f(x_i, y; w)$ is the predicted output

Slack-rescaled Constraints

For each instance (x_i, y_i) define $m := |\mathcal{Y}| - 1$ constraints via

$$f(x_i, y_i; w) - f(x_i, y; w) \geq 1 - \frac{\xi_i}{\triangle(y_i, y)} \quad (\forall y \in \mathcal{Y} - \{y_i\})$$

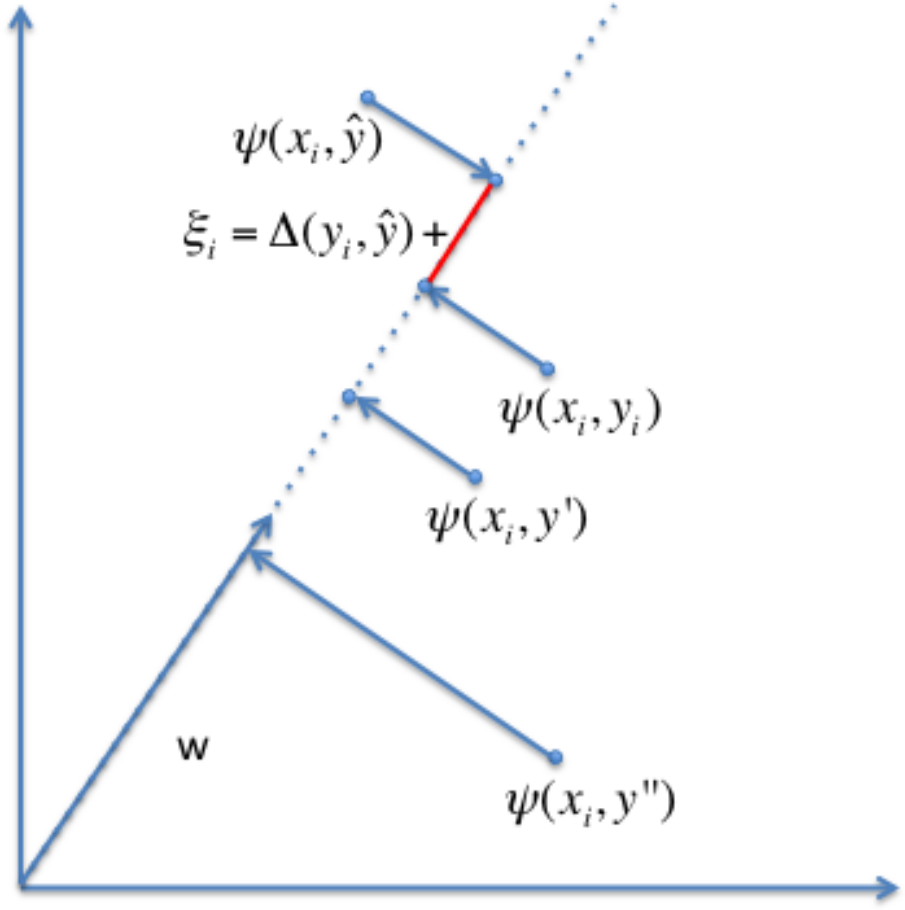
- Require correct output y_i to be scored higher than all incorrect outputs $y \neq y_i$ by a margin
- Penalize margin violations proportional to $\triangle(y_i, y)$
- Provides an upper bound on the empirical loss via

$$\begin{aligned}\xi_i^* &= \max_y \{\triangle(y_i, y) - \triangle(y_i, y)[f(x_i, y_i; w) - f(x_i, y; w)]\} \\ &\geq \triangle(y_i, \hat{y}) - \underbrace{\triangle(y_i, \hat{y})}_{\geq 0}\underbrace{[f(x_i, y_i; w) - f(x_i, \hat{y}; w)]}_{\leq 0} \geq \triangle(y_i, \hat{y})\end{aligned}$$

where $\hat{y}_i := \arg\max_y f(x_i, y; w)$ is the predicted output

Softmargin, Illustration

Geometric sketch



Structured Prediction SVM

Convex Quadratic Program (primal)

$$(w^*, \xi^*) = \arg\min_{w,\xi \geq 0} \mathcal{H}(w, \xi) := \frac{\lambda}{2}\langle w, w\rangle + \frac{1}{n}\|\xi\|_1$$

$$\text{subject to} \quad \langle w, \delta\psi_i(y)\rangle \geq \triangle(y_i, y) - \xi_i \quad (\forall i, \forall y \in \mathcal{Y} - \{y_i\})$$

$$\text{binary: [subject to} \quad \langle w, y_i\phi(x_i)\rangle \geq 1 \qquad - \xi_i \quad (\forall i) \qquad]$$

where $\delta\psi_i(y) := \psi(x_i, y_i) - \psi(x_i, y)$.

- Examples $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \ldots, n$
- Feature map $\psi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$
- Weight vector $w \in \mathbb{R}^d$
- Slack variables $\xi_i \geq 0$, regularization parameter $\lambda \in \mathbb{R}^+$
- Generalizes multiclass SVM [CS02]

RepresenterTheorem

- Denote by $\mathcal{H}$ and RKHS on $\mathcal{X} \times \mathcal{Y}$ with kernel k. A sample set $\mathcal{S} = \{(x_i, y_i) : i = 1, \ldots, n\}$ is given. Furthermore let $\mathcal{C}(f; \mathcal{S}')$ be a functional that depends on f only through its values on the augmented sample $\mathcal{S}' := \{(x_i, y) : (x_i, y_i) \in \mathcal{S}\}$. Let Λ be a strictly monotonically increasing function. Then the solution of the optimization problem $\hat{f}(\mathcal{S}) := \arg\min_{f \in \mathcal{H}} \mathcal{C}(f, \mathcal{S}) + \Lambda(\|f\|_{\mathcal{H}})$ can be written as

$$\hat{f}(\cdot) = \sum_{i,y} \beta_{iy} k(\cdot, (x_i, y))$$

- Linear case

$$\hat{w} = \sum_{i,y} \beta_{iy} \psi(x_i, y)$$

- See: T. Hofmann, B. Schölkopf, A.J. Smola, Kernel methods in machine learning, The Annals of Statistics 2008.

Deriving the Wolfe Dual (1)

Lagrangian

$$\mathcal{L}(\dots) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\|\xi\|_1 - \sum_{i,y\neq y_i} \alpha_{iy}\left[\langle \delta\psi_i(y), w\rangle - \triangle(y_i, y) + \xi_i\right] - \langle \xi, \hat{\xi}\rangle$$

Gradient components

$$\nabla_\xi \mathcal{L} = \frac{1}{n} - \hat{\xi} - \sum_y \alpha_{\bullet y} \stackrel{!}{=} 0 \;\Rightarrow\; 0 \leq \sum_y \alpha_{iy} \leq \frac{1}{n} \quad (\forall i)$$

$$\nabla_w \mathcal{L} = \lambda w - \sum_{i,y\neq y_i} \alpha_{iy} v_{iy} \delta\psi_i(y) \stackrel{!}{=} 0 \;\Rightarrow w^*(\alpha) = \frac{1}{\lambda}\sum_{i,y\neq y_i} \alpha_{iy}\delta\psi_i(y)$$

Re-writing in matrix notation as $w(\alpha)^* = Q\alpha$ with

$$Q := (Q_{r,iy}) \in \mathbb{R}^{d\times n(m-1)}, \quad \text{with} \quad Q_{\bullet,iy} := \frac{1}{\lambda}\delta\psi_i(y)$$

Deriving the Wolfe Dual (2)

Plugging-in solution and exploiting known inequalities

$$\min_{\alpha \geq 0} h(\alpha) := \frac{1}{2}\|Q\alpha\|^2 - \langle \alpha, \triangle \rangle \quad \text{s.t.} \quad n \sum_y \alpha_{iy} \leq 1 \; (\forall i)$$

$$\text{binary: } [\, \frac{1}{2}\|\tilde{Q}\alpha\|^2 - \langle \alpha, \mathbf{1} \rangle \quad \text{s.t.} \quad n\alpha_i \leq 1 \; (\forall i) \,]$$

- Quantity: $n \cdot m$ dual variables instead of n
- Quality: structure of dual is very similar
- Constraints only couple variables in blocks $\{\alpha_{iy} : y \in \mathcal{Y} - \{y_i\}\}$
- Natural factorization of $\alpha \in \mathbb{R}_{\geq 0}^{n(m-1)} = \underbrace{\mathbb{R}_{\geq 0}^{(m-1)} \times \cdots \times \mathbb{R}_{\geq 0}^{(m-1)}}_{n \text{ times}}$
- α/n is a probability mass function $\alpha_{iy_i} := 1 - n \sum_{y \neq y_i} \alpha_{iy}$
- What is a support vector? pair (i, y) with active constraint

Linear Case: Representer Theorem

Looking at the solution w^* we see that

$$
\begin{aligned}
w^* &= \sum_i \sum_{y \neq y_i} \alpha_{iy} \delta\psi_i(y) = \sum_i \sum_{y \neq y_i} \alpha_{iy} [\psi(x_i, y_i) - \psi(x_i, y)] \\
&= \sum_i \underbrace{\left(\sum_{y \neq y_i} \alpha_{iy} \right)}_{:= \beta_{iy_i}} \psi(x_i, y_i) + \sum_i \sum_{y \neq y_i} \underbrace{(-\alpha_{iy})}_{:= \beta_{iy}} \psi(x_i, y) \\
&= \sum_{i,y} \beta_{iy} \psi(x_i, y)
\end{aligned}
$$

as it should be according to the representer theorem.

Section 3

Oracle-Based Algorithms

The Challenge

SVMstruct QP

$$(w^*, \xi^*) = \arg\min_{w,\xi \geq 0} \mathcal{H}(w, \xi) := \frac{\lambda}{2}\langle w, w \rangle + \frac{1}{n}\|\xi\|_1$$

$$\text{with} \quad (w, \xi) \in \bigcap_{iy} \Omega_{iy}, \quad i = 1, \ldots, n, \; y \in \mathcal{Y} - \{y_i\}$$

$$\text{where} \quad \Omega_{iy} := \{(w, \xi) : \langle w, \delta\psi_i(y) \rangle \geq \triangle(y_i, y) - \xi_i\}$$

- Structure of QP is not changed, but number of constraints can be vastly increased relative to binary classification
  - e.g. if $\mathcal{Y}$ is vector of binary labels so that $\mathcal{Y} = \{-1, 1\}^l$ and $m = 2^l$
- Scalable algorithms for this challenge? 10 years of research!

Structured Prediction Perceptron

- Michael Collins 2002, Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms [Col02]
- Perceptron learning avoids the challenge by only focussing on the worst output at a time
  - instead of enforcing constraints over all possible incorrect outputs
- Standard perceptron algorithm with the following modifications
  - Compute prediction

$$\hat{y}_i := F(x_i) = \arg\max_y \langle w, \psi(x_i, y) \rangle$$

  - Perform update according to

$$w \leftarrow \begin{cases} w + \psi(x_i, y_i) - \psi(x_i, \hat{y}) = w + \delta\psi_i(\hat{y}) & \text{if } \hat{y} \neq y_i \\ w & \text{otherwise} \end{cases}$$

  - Novikoff's theorem and mistake bound can be generalized

Separation Oracles

- One idea of the perceptron algorithm turns out to be key: identify the output with the most violating margin constraint
- We call such a sub-routine a separation oracle

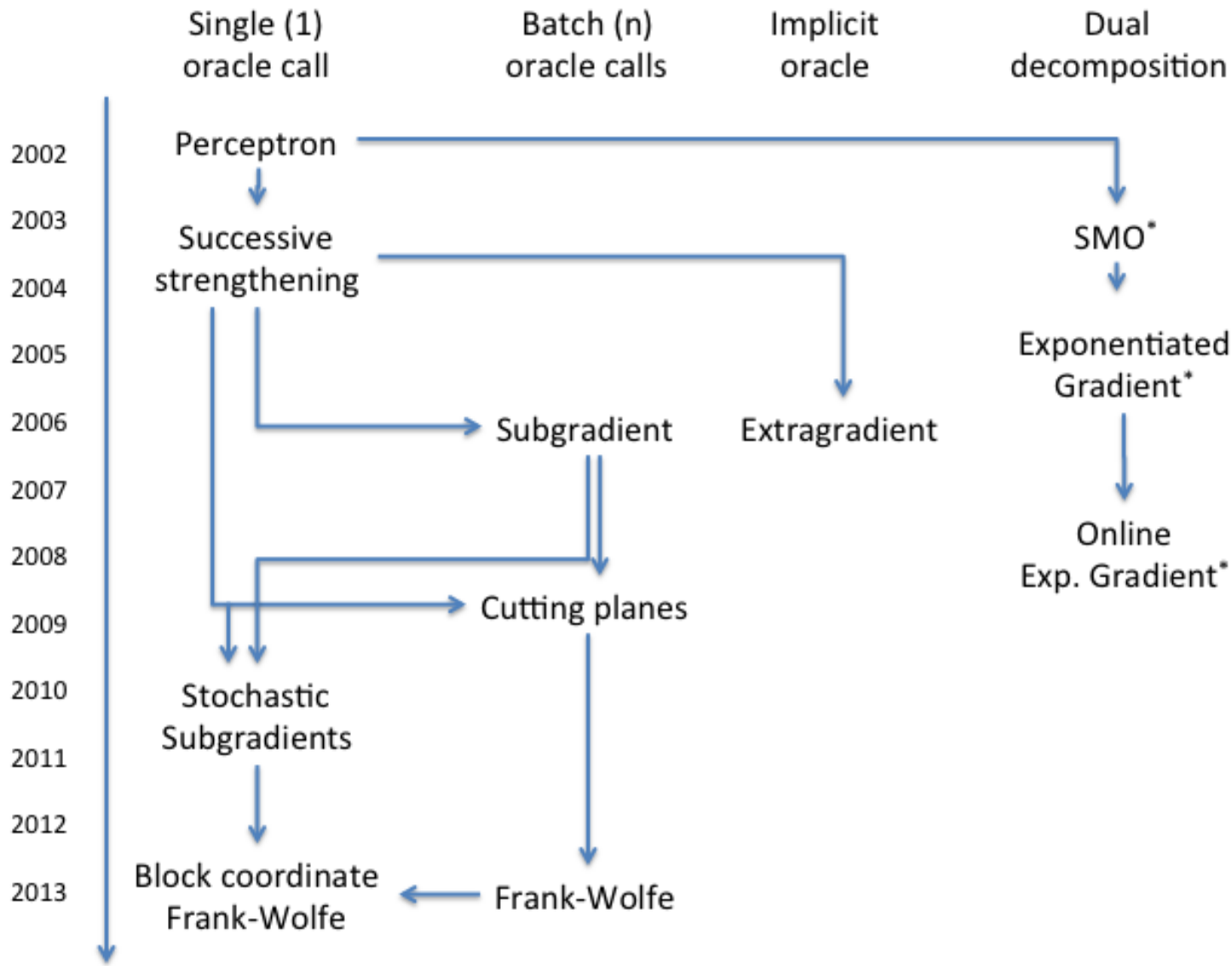
$$\text{Perceptron} \quad \hat{y}_i \in \arg\max_y f(x_i, y; w)$$

$$\text{Margin re-scaling} \quad \hat{y}_i \in \arg\max_y \{\triangle(y_i, y) - f(x_i, y_i; w) + f(x_i, y; w)\}$$

$$\text{Slack re-scaling} \quad \hat{y}_i \in \arg\max_y \{\triangle(y_i, y)[1 - f(x_i, y_i; w) + f(x_i, y; w)]\}$$

- Dependent on the method, the separation oracle is used to identify
  - violated constraints (successive strengthening)
  - update directions for the primal (subgradient)
  - variables in the dual (SMO)
  - update directions for the dual (Frank-Wolfe)

Large Margin Algorithms - Taxonomy & History



Successive QP Strengthening

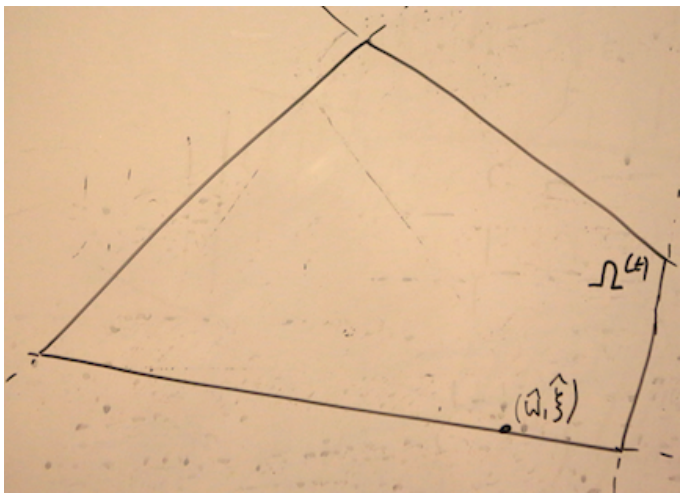
- Create sequence of QPs that are relaxations of *SVMstruct*.
- Feasible domain $\Omega = \bigcap_{iy} \Omega_{iy} \cap (\mathbb{R}^d \times \mathbb{R}^n_{\geq 0})$
- Relaxed QP: same objective, yet $\hat{\Omega} \supset \Omega$
  - optimal solution $(\hat{w}, \hat{\xi})$ for relaxed QP will have $\mathcal{H}(\hat{w}, \hat{\xi}) \leq \mathcal{H}(w^*, \xi^*)$, but possibly $(\hat{w}, \hat{\xi}) \in \hat{\Omega} - \Omega$.
  - goal: fulfill remaining constraints with tolerance ϵ, $(\hat{w}, \hat{\xi} + \epsilon) \in \Omega$
  - why? this would give $\mathcal{H}(\hat{w}, \hat{\xi} + \epsilon) = \mathcal{H}(\hat{w}, \hat{\xi}) + \epsilon \geq \mathcal{H}(w^*, \xi^*)$.
- Construct strict sequence of increasingly stronger relaxations via

$$\Omega(0) = \mathbb{R}^d \times \mathbb{R}^n_{\geq 0}, \quad \Omega(t+1) := \Omega(t) \cap \Omega_{i\hat{y}}$$

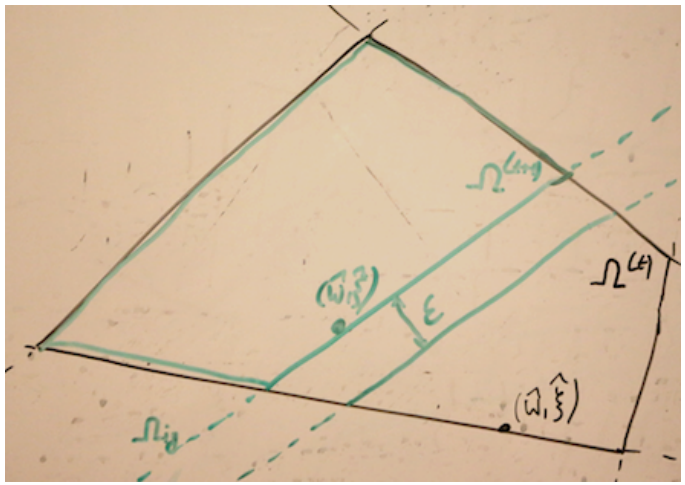
where $(i, \hat{y})$ is a constraint selected at step t fulfilling

$$\underset{(w,\xi)\in\Omega(t)}{\arg\min} \; \mathcal{H}(w, \xi) \notin \Omega^{\epsilon}_{i\hat{y}}, \quad \Omega^{\epsilon}_{iy} := \{(w, \xi) : (w, \xi + \epsilon) \in \Omega_{iy}\}$$

Strengthening via Separation Oracle



$\rightarrow$



Strengthening via Separation Oracle

- Loop through all training examples (in fixed order)
- Call separation oracle for (x_i, y_i)
- Concretely for margin re-scaling

$$\hat{y}_i \in \arg\max_y \{\triangle(y_i, y) - f(x_i, y_i; w) + f(x_i, y; w)\}$$

will identify (one of) the most violating constraint(s) for given i, provided there are such constraints

- We can easily check, whether violation is $> \epsilon$.
- Termination at step T, if no such constraints exist for $i = 1, \ldots, n$.

- Significance: can ensure $T \leq O(n/\epsilon^2)$ or (with mild conditions) even $T \leq O(n/\epsilon)$. No dependency on $|\mathcal{Y}|$!

Strengthening via Separation Oracle; Example ($n = 1$)

- Step 0: $(\hat{w}, \hat{\xi}) = \arg\min_{\Omega(0)} \mathcal{H}(w, \xi) = (0, 0)$

Strengthening via Separation Oracle; Example ($n = 1$)

- Step 0: $(\hat{w}, \hat{\xi}) = \arg\min_{\Omega(0)} \mathcal{H}(w, \xi) = (0, 0)$
- Step 1: $(\hat{w}, \hat{\xi}) = \arg\min_{\Omega(1)} \mathcal{H}(w, \xi)$, where

$$\hat{y} \in \arg\max_{y} \triangle(y_1, y),$$

$$\Omega(1) = \Omega(0) \cap \Omega_{1\hat{y}}$$

Strengthening via Separation Oracle; Example ($n = 1$)

- Step 0: $(\hat{w}, \hat{\xi}) = \arg\min_{\Omega(0)} \mathcal{H}(w, \xi) = (0, 0)$
- Step 1: $(\hat{w}, \hat{\xi}) = \arg\min_{\Omega(1)} \mathcal{H}(w, \xi)$, where

$$\hat{y} \in \arg\max_{y} \triangle(y_1, y),$$

$$\Omega(1) = \Omega(0) \cap \Omega_{1\hat{y}}$$

- Step 2: $(\hat{w}, \hat{\xi}) = \arg\min_{\Omega(2)} \mathcal{H}(w, \xi)$, where

$$\hat{y} \in \arg\max_{y} \triangle(y_1, y) - \langle \delta\psi_1(y), \hat{w} \rangle$$

$$\Omega(2) = \Omega(1) \cap \Omega_{1\hat{y}}$$

provided that $\Omega^{\epsilon}_{iy} \cap \Omega(1) \neq \emptyset$.

Strengthening via Separation Oracle; Example ($n = 1$)

- Step 0: $(\hat{w}, \hat{\xi}) = \arg\min_{\Omega(0)} \mathcal{H}(w, \xi) = (0, 0)$
- Step 1: $(\hat{w}, \hat{\xi}) = \arg\min_{\Omega(1)} \mathcal{H}(w, \xi)$, where

$$\hat{y} \in \arg\max_{y} \triangle(y_1, y),$$

$$\Omega(1) = \Omega(0) \cap \Omega_{1\hat{y}}$$

- Step 2: $(\hat{w}, \hat{\xi}) = \arg\min_{\Omega(2)} \mathcal{H}(w, \xi)$, where

$$\hat{y} \in \arg\max_{y} \triangle(y_1, y) - \langle \delta\psi_1(y), \hat{w} \rangle$$

$$\Omega(2) = \Omega(1) \cap \Omega_{1\hat{y}}$$

provided that $\Omega^{\epsilon}_{iy} \cap \Omega(1) \neq \emptyset$.

- Step 3: $(\hat{w}, \hat{\xi}) = \arg\min_{\Omega(3)} \mathcal{H}(w, \xi)$, where

...

Improved Cutting Planes: Motivation

- Successive strengthening (as above) is expensive
  - only one constraint (for one example) gets added in each step
  - requires re-optimization (= solving a QP) after each such step
  - can warm-start, but still...
- How about, we compute all oracles in parallel

$$\hat{y} = (\hat{y}_1, \ldots, \hat{y}_n) \in \mathcal{Y}^n$$

- Derive a strengthening from that $\Omega(t+1) = \Omega(t) \cap \Omega_{\hat{y}}$
- Naively could set $\Omega_{\hat{y}} := \bigcap_i \Omega_{i\hat{y}_i}$
  - ... but how would that give us improved termination guarantees?
  - ... how can we avoid blow-up in number of constraints?
- Instead summarize into a single linear constraint with a single shared slack variable $\zeta \geq 0$. Fulfill margin on average

$$\sum_{i=1}^{n} \langle \psi_i(\hat{y}_i), w \rangle \geq \sum_{i=1}^{n} \triangle(y_i, \hat{y}_i) - \zeta$$

Improved Cutting Planes: Algorithm

- [JFY09] show that the QP containing all such average constraints for all combinations $\mathbf{y} \in \mathcal{Y}^n$ is solution equivalent to SVMstruct, if one identifies $\zeta = \|\xi_i\|_1$.

$\min_{w,\xi} \frac{\lambda}{2}\langle w, w\rangle + \frac{1}{n}\|\xi\|_1$ s.t.	$\min_{w,\zeta} \frac{\lambda}{2}\langle w, w\rangle + \zeta$ s.t.
$\langle \delta\psi_i(y), w\rangle \geq \triangle(y_i, y) - \xi_i$	$\sum_i \langle \delta\psi_i(y), w\rangle \geq \sum_i \triangle(y_i, y) - \zeta$
$(\forall i, y \in \mathcal{Y}) \sim n \cdot m$	$(\forall y \in \mathcal{Y}^n) \sim m^n$

- [JFY09] also provide $O(1/\epsilon)$-bounds on the number of epochs
  - overall runtime $O(n/\epsilon)$ (in the linear case), not counting oracle
- Dual QP optimization
  - one variable for each selected (average constraint), highly sparse
  - complexity of $O(n^2)$; with reduced rank approx. $O(nr + r^3)$

Improved Cutting Planes: Experiments

Experiments from [JFY09]

			CPU-Time		# Sep. Oracle		# Support Vec.	
	n	N	1-slack	n-slack	1-slack	n-slack	1-slack	n-slack
MultiC	522,911	378	1.05	1180.56	4,183,288	10,981,131	98	334,524
HMM	35,531	18,573,781	0.90	177.00	1,314,647	4,476,906	139	83,126
CFG	9,780	154,655	2.90	8.52	224,940	479,220	70	12,890

- # calls to separation oracle 2-3x reduced
- CPU time, 5x-1000x dependent on time spent on QP vs. oracle
  $\Rightarrow$ much more efficient usage of optimization time
- 1000-10000x fewer support vectors, but not when multiplied by n
- Approximation result $O(1/\epsilon)$
- Book-keeping overhead for storing #SVs $\cdot n$ descriptors of size $O(\log m)$

Subgradient Method for SVMstruct

- Can we avoid solving many relaxed QPs?
  How about a gradient descent flavor method?
- We can avoid linearizing (i.e. rolling out) the constraints.
  Work directly with (unconstrained) piecewise linear objective

$$w^* = \arg\min_w \frac{\lambda}{2}\langle w, w\rangle + \frac{1}{n}\sum_{i=1}^{n} \underbrace{\max_y \{\triangle(y_i, y) - \langle \delta\psi_i(y), w\rangle\}}_{\text{oracle } \hat{y}_i := \arg\max()}$$

- Compute subgradient, e.g. via

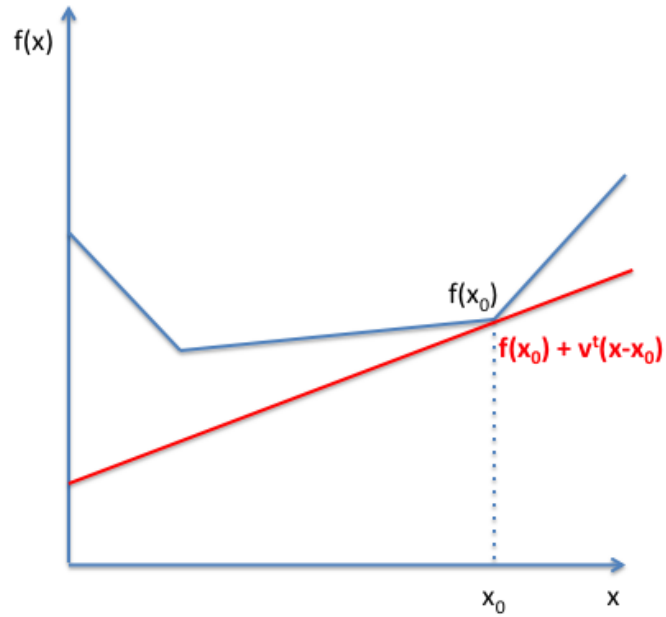
$$g = \lambda w + \frac{1}{n}\sum_{i=1}^{n} \delta\psi_i(\hat{y}_i)$$

- Perform batch or stochastic updates on w (learning rate?)
- Proposed by [RBZ07]; see also PEGASOS [SSSSC11]

Background: Subgradient Methods

- Let $f : \mathbb{R}^D \to \mathbb{R}$ be a convex, not necessarily differentiable function. A vector $v \in \mathbb{R}^D$ is called a **subgradient** of f at x_0, if

$$f(x) \geq f(x_0) + \langle v, x - x_0 \rangle$$
for all x



- Differentiable point x_0: unique subgradient = gradient $\nabla f(x_0)$.

Frank-Wolfe Algorithm

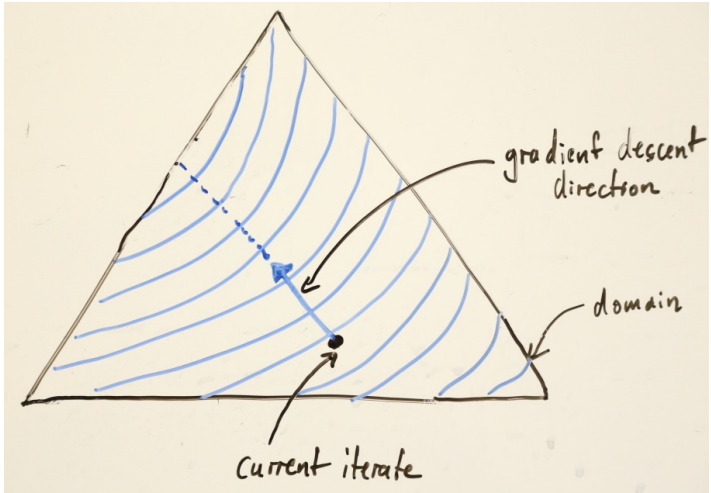
- Frank & Wolfe, 1956: *An algorithm for quadratic programming*
- Minimize linearization at current iterate over corners of domain

$$\text{'new iterate'} := (1-\eta) \cdot \text{'old iterate'} + \eta \cdot \text{'optimal corner'}$$

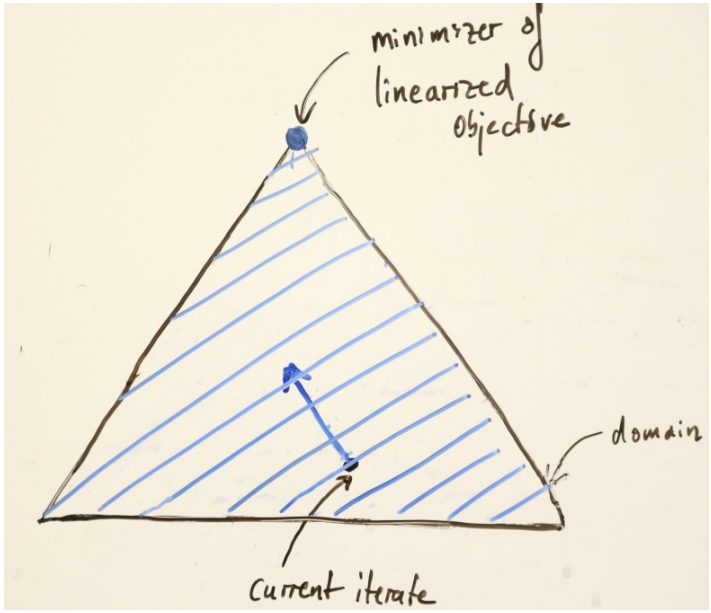
- Features
  - **linearity**: linear, not quadratic function minimization in every step
  - **sparseness**: convex combination of selected corners
  - **projection-free**: iterates stay in convex domain
  - **learning rate**: $O(1/t)$ schedule or via line search
  - **duality gap**: implicitly computes duality gap
- Applied to SVMstruct by [LJJSP13]

Frank-Wolfe Algorithm: Quadratic vs. Linearized

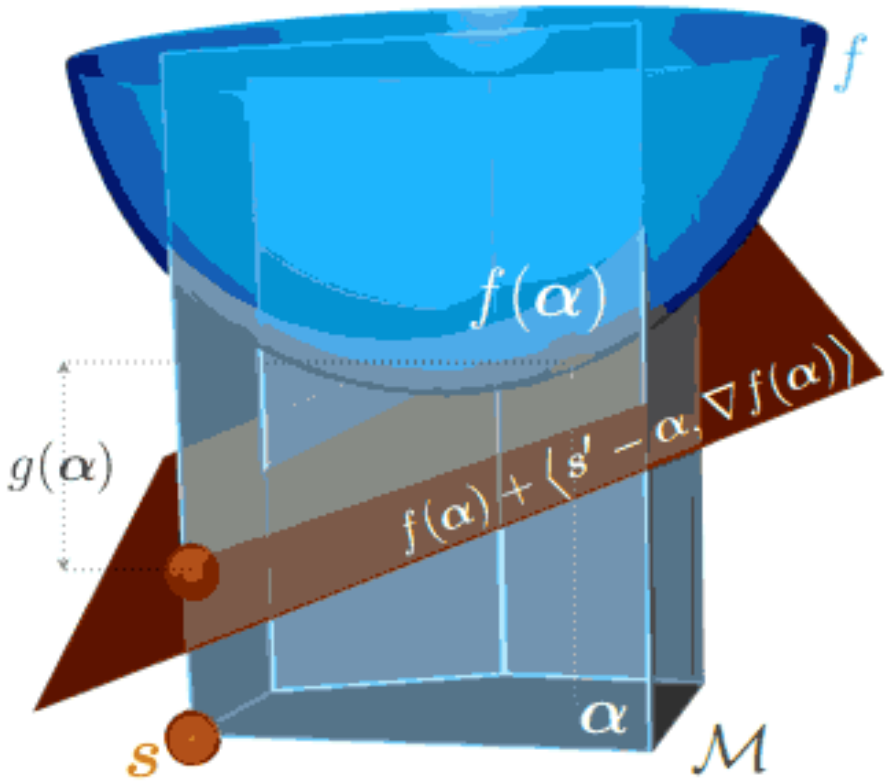
- Quadratic objective (contour line plot)



- Linearized objective



Frank-Wolfe Algorithm: Schematic 3D View



[taken from Lacoste-Julien et al., 2013]

Frank-Wolfe Algorithm: Dual SVM-struct Objective

- Dual objective

$$h(\alpha) = \frac{1}{2}\|Q\alpha\|^2 - \langle \triangle, \alpha \rangle$$

- Gradient

$$\nabla_\alpha h(\alpha^*) = (Q'Q)\alpha^* - \triangle$$

- Linearization

$$\bar{h}(\alpha; \alpha^*) = \underbrace{h(\alpha^*)}_{=\text{const.}} + \langle \nabla_\alpha h(\alpha^*), \alpha - \alpha^* \rangle \underbrace{\leq h(\alpha)}_{\text{convexity}}$$

- Minimization problem

$$e^* := \underset{\{e_r : r=1,\ldots,m\}}{\arg\min} \; \bar{h}(e_r; \alpha^*), \quad \text{with } e_r\text{: } r\text{-th unit vector}$$

Frank-Wolfe Algorithm: Deciphered

- What does the minimization problem over corners mean?

$$\begin{aligned}\bar{h}(e_{y'};\alpha^*) + const. &= \langle \underbrace{\nabla_\alpha h(\alpha^*)}_{=(Q'Q)\alpha^* - \triangle}, e_{y'}\rangle \\ &= \langle Qe_{\mathbf{y}'}, \underbrace{Q\alpha^*}_{=w}\rangle + \triangle(y,y') \\ &= \langle \textstyle\sum_i \delta\psi_i(y_i'), w\rangle + \sum_i \triangle(y_i, y_i')\end{aligned}$$

so that

$$\hat{y}_i = \arg\max_{y'} \{\langle \delta\psi_i(y'), w\rangle + \triangle(y_i, y')\}$$

which is just the separation oracle!

Algorithms: Frank-Wolfe, Subgradient, Cutting Plane

- How does Frank-Wolfe relate to the other methods?
- FW $\leftrightarrow$ Subgradients:
  - Same update direction of primal solution w
  - But: Smarter step-size policy derived from dual (see below)
  - But: Duality gap for meaningful termination condition (see below)
- FW $\leftrightarrow$ improved cutting planes:
  - Selected dual variables correspond to added constraints
  - But: incremental update step vs. optimization of relaxed QP
  - But: #SV can be larger due to incremental method, no need to re-formulate SVM struct
- Further advantages
  - Simple and clean analysis
  - Per-instance updates (block-coordinate optimization)

Frank-Wolfe Algorithm: Primal-Dual Version

- Apply Frank-Wolfe to dual QP, but translate into primal updates
- Compute primal update direction (subgradient)

$$\bar{w} := \frac{1}{\lambda} \sum_{i=1}^{n} \delta\psi_i(\hat{y}_i), \quad \bar{\triangle} := \frac{1}{n} \sum_{i=1}^{n} \triangle(y_i, \hat{y}_i)$$

- Perform convex combination update

$$w^{t+1} = (1 - \gamma^t) w^t + \gamma^t \bar{w}, \quad \triangle^{t+1} = (1 - \gamma^t) \triangle^t + \gamma \bar{\triangle}$$

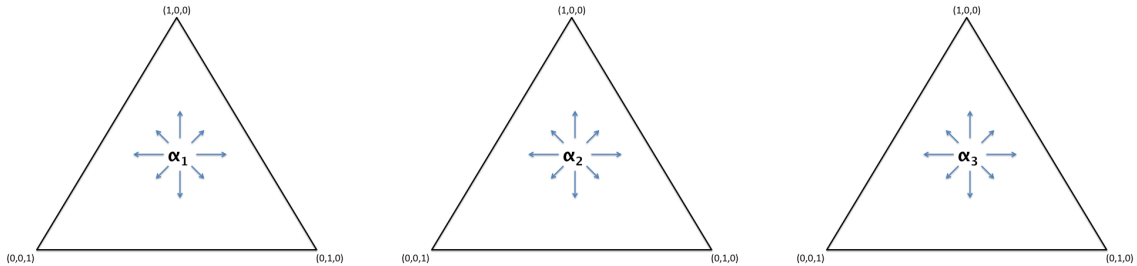
  here the optimal γ^t can be computed analytically (closed-form line search) from w^t, $\bar{\triangle}$ and $\bar{w}$

- Convergence rate: ϵ-approximation is found in at most $O(\frac{R^2}{\lambda\epsilon})$ steps

Block-Coordinate Frank-Wolfe

- Domain of the dual QP factorizes $\alpha \in \mathcal{S}_{m-1}^n$ (product of simplicies)

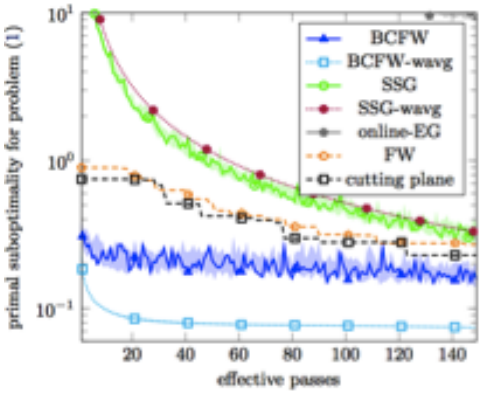
$$\alpha = (\alpha_i)_{i=1}^n, \quad \text{s.t.} \quad \alpha_i \geq \mathbf{0} \quad \text{and} \quad \langle \alpha_i, \mathbf{1} \rangle = 1$$



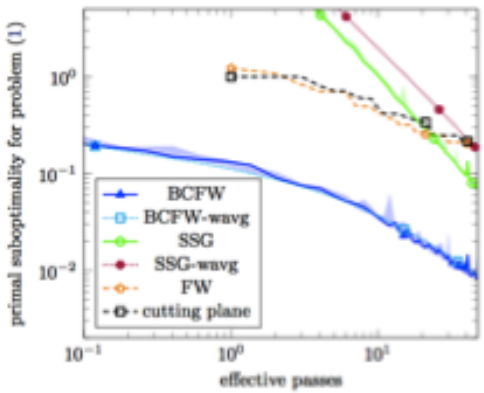
$$\alpha = (\quad \alpha_1 \quad \alpha_2 \quad \alpha_3 \quad \dots)$$

- Perform Frank-Wolfe update over each block (randomly selected).
  - single-instance mode: alternates single oracle call and update step
  - back to successive strengthening, but replace: re-optimization with fast updates
  - convergence rate analysis; duality gap as stopping criterion
  - excellent scalability

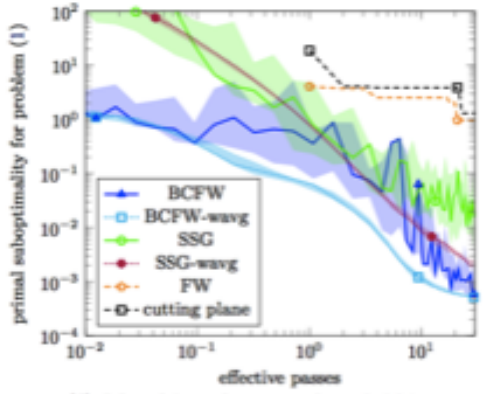
Frank-Wolfe Methods: Scalability [LJJSP13]



(c) OCR dataset, $\lambda = 1/n$.



(d) CoNLL dataset, $\lambda = 1/n$.



(f) Matching dataset, $\lambda = 0.001$.

- Frank-Wolfe very similar to improved cutting plane method
- Block-coordinate version much faster, better than stochastic subgradient descent
- Main caveat: primal-dual version needs to store one weight vector per training instance!!

Implicit Oracle as LP Relaxation

- Sometimes, oracle can be integrated into the QP

$$\begin{aligned}&\max_{y \in \mathcal{Y}} \{\langle \delta\psi_i(y), w\rangle + \triangle(y_i, y)\} \\ &= \max_{z_i \in \mathcal{Z}} \langle z_i, c_i + F_i w\rangle + d_i\end{aligned}$$

- Examples: binary MRFs with sub modular potentials, matchings, tree-structured MRFs
- Saddle point formulation:

$$\min_w \max_z \left\{ \frac{\lambda}{2}\|w\|^2 + \sum_{i=1}^{n} \langle z_i, c_i + F_i w\rangle - \langle \psi(x_i, y_i), w\rangle \right\}$$

- Make use of extragradient method [TLJJ06] - gradients & projections

Bi-partite Matching

- Graph $\mathcal{G}(V, E)$ with $V = V^s \cup V^t$, $E = V^s \times V^t$
- Matching scores $s_{jk} \in \mathbb{R}$ for each edge $(j, k) \in E$.
- Alignment variables $y_{jk} \in \{0, 1\}$ and their relaxation $z_{jk} \in [0; 1]$
- LP relaxation of integer program

$$\max_{0 \le z \le 1} \sum_{(j,k) \in E} s_{jk} z_{jk}, \quad \text{s.t.} \sum_j z_{jk} \le 1 \; (\forall k) \quad \text{and} \quad \sum_k z_{jk} \le 1 \; (\forall j)$$

- LP is guaranteed to have integral solutions
- Integrating into SVM struct QP

$$\max_{\{0 \le z_i \le 1\}} \sum_{e \in E} z_{ie} \underbrace{\langle \psi(x_i, y_e), w \rangle}_{s_{i,jk},\ e=(j,k)} + \underbrace{(1 - 2y_{ie})}_{\text{Hamming loss}}$$

Section 4

Decomposition-Based Algorithms

Factor Graphs

- In many cases of practical interest, the compatibility function naturally allows for an additive decomposition over factors or parts

$$f(x, y) = \sum_{c \in \mathcal{C}} f_c(x_c, y_c)$$

  which can formally be described as a factor graph.
- In the linear case, this can be induced via a feature decomposition

$$\psi(x, y) = \sum_{c \in \mathcal{C}} \psi_c(x_c, y_c), \quad \text{such that}$$

$$f(x, y; w) = \langle w, \psi(x, y) \rangle = \sum_c \underbrace{\langle w, \psi_c(x_c, y_c) \rangle}_{=: f_c(x_c, y_c)}$$

- We typically require that the loss decomposes in a compatible manner

$$\triangle(y, y'; x) = \sum_{c \in \mathcal{C}} \triangle_c(y_c, y'_c; x_c)$$

Representer Theorem for the Factorized Case

- Conditions as before but factor structure assumed. Denote configurations for factor c as $z \in \mathcal{Z}(c)$.
- Representation

$$f(x,y) = \sum_{i=1}^{n} \underbrace{\sum_{c \in \mathcal{C}} \sum_{z \in \mathcal{Z}(c)}}_{\sum_c |\mathcal{Z}(c)| \ll |\mathcal{Y}|} \mu_{icz} \underbrace{\langle \psi_c(x_{ic}, z), \psi_c(x_{ic}, y_c) \rangle}_{=: k_c((x_{ic},z),(x_{ic},y_c))}$$

- Note that this offers the possibility to
  1. define kernels on a *per factor* level
  2. use a low-dimensional parametrization that does not need to rely on sparseness

Decomposing the Dual QP

- Dual has the following structure (rescaling by n as appropriate to make α probability mass function)

$$\min_{\alpha \geq 0} h(\alpha) := \frac{1}{2}\|Q\alpha\|^2 - \langle \alpha, \triangle \rangle \quad \text{s.t.} \quad \sum_y \alpha_{iy} = 1 \; (\forall i)$$

- Introduce **marginal probabilities**

$$\mu_{icz} := \sum_{i,y} \mathbf{1}[y_c = z]\, \alpha_{iy}, \quad \sum_z \mu_{icz} = \sum_{i,y} \alpha_{iy} = 1$$

- Decompose loss (similar for $Q^t Q$)

$$\sum_y \alpha_{iy} \triangle(y_i, y) = \sum_y \alpha_{iy} \sum_c \triangle_c(y_{ic}, y_c)$$

$$= \sum_{c,z} \underbrace{\left(\sum_y \mathbf{1}[y_c = z]\, \alpha_{iy} \right)}_{=\mu_{icz}} \triangle_c(y_{ic}, z)$$

Decomposing the Dual QP (continued)

- Define with multi-index (icz):

$$Q_{\bullet,icz} := \psi_c(x_{ic}, y_{ic}) - \psi_c(x_{ic}, z), \quad \mu_{\mathcal{C}} := (\mu_{icz}), \quad \triangle_{\mathcal{C}} := (\triangle_{icz})$$

- Factorized QP

$$\mu^*_{\mathcal{C}} = \arg\min_{\mu_{\mathcal{C}} \geq 0} \left\{ \frac{1}{2} \|Q\mu_{\mathcal{C}}\|^2 - \langle \mu_{\mathcal{C}}, \triangle_{\mathcal{C}} \rangle \right\}$$

s.t. $\mu_{\mathcal{C}}$ is on the **marginal polytope**

  - $\mu_{\mathcal{C}}$ needs to be normalized and locally consistent (non-trivial).
  - objective broken up into parts - global view enforced via constraints!

- Example: **Singly connected factor graph**. Local consistency:

$$\sum_{r:(r,s)\in\mathcal{C}} \mu_{irs} = \mu_{is} \quad (\forall i, s)$$

- For general factor graphs: only enforce local consistency = **relaxation** in the spirit of approximate belief propagation [TGK03]

Conditional Exponential Family Models

- Structured prediction from a statistical modeling angle
- f from some RHKS with kernel k over $\mathcal{X} \times \mathcal{Y}$
- **Conditional exponential families**

$$p(y|x; f) = \exp\left[f(x, y) - g(x, f)\right], \quad \text{where}$$
$$g(x, f) := \int_{\mathcal{Y}} \exp\left[f(x, y)\right] d\nu(y)$$

- Univariate case ($y \in \mathbb{R}$), generalized linear models

$$p(y|x; w) = \exp\left[y\langle w, \phi(x)\rangle - g(x, w)\right]$$

- Non-parameteric models, e.g. ANOVA kernels

$$k((x, y), (x', y') = yy' k(x, x')$$

Conditional Random Fields

- Conditional log-likelihood criterion [LMP01, LZL04]

$$f^* := \arg\min_{f \in \mathcal{H}} \frac{\lambda}{2} \underbrace{\|f\|_{\mathcal{H}}^2}_{\text{stabilizer}} \underbrace{- \frac{1}{n} \sum_{i=1}^{n} \log p(y_i | x_i; f)}_{\text{log-loss}}$$

- Optimization methods:
  - improved iterative scaling [LMP01]
  - pre-conditioned conjugate gradient descent, limited memory quasi-Newton [SP03]
  - finite dimensional case: requires computing expectations of sufficient statistics $\mathbf{E}[\psi(Y, x)]$ for $x = x_i$, $i = 1, \ldots, n$.

$$\nabla_w [\ldots] \stackrel{!}{=} 0 \iff \lambda w^* = \underbrace{\frac{1}{n} \sum_{i=1}^{n} \psi(x_i, y_i)}_{\text{sample statistics}} - \underbrace{\frac{1}{n} \sum_{i=1}^{n} \sum_{y \in \mathcal{Y}} \psi(x_i, y) p(y | x_i; w^*)}_{\text{expected statistics}}$$

Dual CRF

- Representer theorems apply to log-loss. Log-linear dual:

$$\alpha^* = \arg\min_{\alpha \geq 0} h(\alpha) := \frac{1}{2}\|Q\alpha\|^2 + \sum_{i=1}^{n} \sum_{y \in \mathcal{Y}} \alpha_{iy} \log \alpha_{iy}$$

$$\text{s.t.} \quad \sum_{y \in \mathcal{Y}} \alpha_{iy} = 1 \quad (\forall i)$$

- Compare with SVM struct

$$\frac{1}{2}\|Q\alpha\|^2 + \sum_{i=1}^{n} \sum_{y \in \mathcal{Y}} \alpha_{iy} \log \alpha_{iy}$$

$$\frac{1}{2}\|Q\alpha\|^2 - \sum_{i=1}^{n} \sum_{y \in \mathcal{Y}} \alpha_{iy} \triangle(y_i, y)$$

  - same data matrix Q constructed from $\delta\psi_i(y)$ (or Q^tQ via kernels)
  - same n-factor simplex constraints on α
  - entropy maximization, instead of linear penalty (based on loss)

Exponentiated Gradient Descent

- Exponentiated gradient descent [CGK+08] can be applied to solve both duals (hinge loss and logarithmic loss)
- General update equation

$$\begin{aligned}\alpha_{iy}^{(t+1)} &\propto \alpha_{iy}^{(t)} \cdot \exp\left[\nabla h(\alpha)\right] \\ &= \alpha_{iy}^{(t)} \cdot \exp\left[\lambda\langle w^*, \delta\psi_i(y)\rangle - \triangle(y_i, y)\right]\end{aligned}$$

  - Can be motivated by performing gradient descent on the canonical/natural parameters (and re-formulating in mean-value parameterization)

$$\theta^{(t+1)} = \theta^{(t)} + \delta\theta^{(t)} \Rightarrow \alpha^{(t+1)} = \exp[\langle\psi, \theta^{(t+1)}\rangle] = \exp[\langle\psi, \delta\theta^{(t)}\rangle]\alpha^{(t)}$$

  - on-line version: generalizes SMO for solving dual problem (when no closed form solution exists)

Factorized Exponentiated Gradient Descent

- Work with factorized dual QP: e.g. [TGK03], SMO over marginal variables $\mu_{\mathcal{C}}$.
- Better: adopt exponentiated gradient descent [CM05, CGK+08]
- Derivation: summing on both sides of the update equation...

$$\begin{aligned}
\mu_{icz}^{(t+1)} &= \sum_{y} \mathbf{1}[y_c = z]\, \alpha_{iy}^{(t)} \exp\left[\lambda \langle w^*, \delta\psi_i(y)\rangle - \triangle(y_i, y)\right] \\
&\propto \sum_{y} \mathbf{1}[y_c = z]\, \alpha_{iy}^{(t)} \exp\left[\lambda \langle w^*, \psi_c(x_i, y_{iz}) - \psi_c(x_i, z)\rangle - \triangle(y_{iz}, z)\right] \\
&= \mu_{icz}^{(t)} \cdot \exp\left[\lambda \langle w^*, \psi_c(x_i, y_{iz}) - \psi_c(x_i, z)\rangle - \triangle(y_{iz}, z)\right]
\end{aligned}$$

  - w^* can (representer theorem) computed from μ and ψ_c (or via k_c), $\triangle_c$ terms.
- Similar for log-loss, faster convergence rates $O(\log 1/\epsilon)$.

Section 5

Conclusion & Discussion

Structured Prediction

- Support Vector Machines: can be generalized to structured prediction in a scalable manner
- Oracle-based architecture: decouples general learning method from domain-specific aspects
- Features & loss function: can be incorporated in a flexible manner
- Kernels: efficient dual methods exist that can rely on kernels (crossed feature maps, factor-level kernels)
- Algorithms: rich set of scalable methods; cutting planes, subgradients, Frank-Wolfe, exponentiated gradient
- Decomposition-based methods: can exploit insights and algorithms from approximate probabilistic inference
- Conditional random fields: close relation (decomposition, dual, sparseness?)
- Applications: ever increasing number of applications and use cases

Yasemin Altun, Ioannis Tsochantaridis, Thomas Hofmann, et al.

Hidden Markov Support Vector Machines.

In *ICML*, volume 3, pages 3–10, 2003.

Alexander Binder, Klaus-Robert Müller, and Motoaki Kawanabe.

On taxonomies for multi-class image categorization.

International Journal of Computer Vision, 99(3):281–301, 2012.

O. Chapelle, C.B. Do, Q.V. Le C.H. Teo, and A.J. Smola.

Tighter bounds for structured estimation.

In *Advances in neural information processing systems*, pages 281–288, 2008.

Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L Bartlett.

Exponentiated gradient algorithms for conditional random fields and max-margin markov networks.

The Journal of Machine Learning Research, 9:1775–1822, 2008.

Lijuan Cai and Thomas Hofmann.

Hierarchical document categorization with support vector machines.

In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 78–87. ACM, 2004.

Peter L Bartlett Michael Collins and Ben Taskar David McAllester.

Exponentiated gradient algorithms for large-margin structured classification.

In *Advances in Neural Information Processing Systems 17: Proceedings of the 2004 Conference*, volume 17, page 113. The MIT Press, 2005.

Michael Collins.

Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms.

In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics, 2002.

Koby Crammer and Yoram Singer.

On the algorithmic implementation of multiclass kernel-based vector machines.

The Journal of Machine Learning Research, 2:265–292, 2002.


R. Collobert, F. Sinz, Jason J. Weston, and L. Bottou.

Trading convexity for scalability.

In *Proceedings of the 23rd International Conference on Machine Learning*, pages 201–208. ACM, 2006.


Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu.

Cutting-plane training of structural SVMs.

Machine Learning, 77(1):27–59, 2009.


Thorsten Joachims, Thomas Hofmann, Yisong Yue, and Chun-Nam Yu.

Predicting structured objects with Support Vector Machines.

Communications of the ACM, 52(11):97–104, 2009.


Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher.

Block-coordinate Frank-Wolfe optimization for structural SVMs.

In *International Conference on Machine Learning (ICML)*, 2013.


Yoonkyung Lee, Yi Lin, and Grace Wahba.

Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data.

Journal of the American Statistical Association, 99(465):67–81, 2004.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira.

Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

John Lafferty, Xiaojin Zhu, and Yan Liu.

Kernel conditional random fields: representation and clique selection.

In *Proceedings of the twenty-first international conference on Machine learning*, page 64. ACM, 2004.

Nathan D Ratliff, J Andrew Bagnell, and Martin Zinkevich.

(approximate) subgradient methods for structured prediction.

In *International Conference on Artificial Intelligence and Statistics*, pages 380–387, 2007.


Ryan Rifkin and Aldebaro Klautau.

In defense of one-vs-all classification.

The Journal of Machine Learning Research, 5:101–141, 2004.


Fei Sha and Fernando Pereira.

Shallow parsing with conditional random fields.

In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics, 2003.


Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter.

Pegasos: Primal estimated sub-gradient solver for SVM.

Mathematical Programming, 127(1):3–30, 2011.


Ben Taskar, Carlos Guestrin, and Daphne Koller.

Max-margin markov networks.

In *Advances in Neural Information Processing Systems*. MIT Press, 2003.

Ben Taskar, Dan Klein, Michael Collins, Daphne Koller, and Christopher Manning.

Max-margin parsing.

In *In Proceedings of EMNLP*, 2004.

Ben Taskar, Simon Lacoste-Julien, and Michael I Jordan.

Structured prediction, dual extragradient and Bregman projections.

The Journal of Machine Learning Research, 7:1627–1653, 2006.

Jason Weston and Chris Watkins.

Support vector machines for multi-class pattern recognition.

In *ESANN*, volume 99, pages 61–72, 1999.

Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims.

A support vector method for optimizing average precision.

In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 271–278. ACM, 2007.

Alan L Yuille and Anand Rangarajan.

The concave-convex procedure.

Neural Computation, 15(4):915–936, 2003.