Hidden Markov Support Vector Machines

Label Sequence Learning

Aija Niissalo

main ref. Yasemin Altun, Ioannis Tsochantaridis, Thomas Hofmann:
Hidden Markov Support Vector Machines. Proc. ICML'2003

April 1, 2008

Outline

Introducing A Label Sequence Learning Technique

Discriminative learning technique for **label sequences**

- **Structured output prediction problem** hard: the set of label sequences scale exponentially
- Solution: Combination of **Support Vector Machines** and **Hidden Markov Models**
- Generalization of SVM
- Handels dependencies between neighbouring labels using **Viterbi decoding**
- Learning is discriminative and is based on a **maximum/large/soft margin** criterion
- Can learn non-linear discriminant functions via **kernel** functions
- Can deal with **overlapping features**
- Example tasks (in the paper): **named entity recognition** and **part-of-speech tagging**

Learning from observation sequences

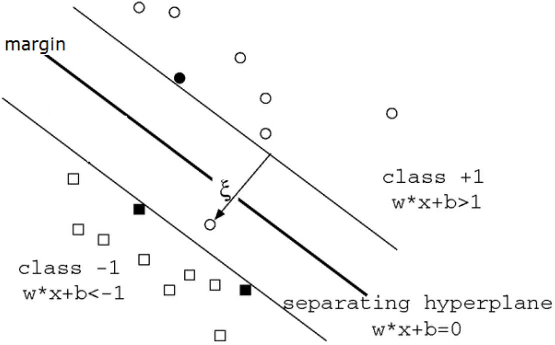
Learning from observation sequences by predicting label sequences instead of individual class labels: **Label Sequence Learning**

- *natural language processing, speech recognition, computational biology, system identification*
- Can solve problems:
  1. **Segmenting** observation sequences
  2. **Annotating** observation sequences
  3. **Recovering** underlying discrete source
- HMM (predominant formalism for modelling label sequences) limitations:
  1. Trained in a *non-discriminative* manner $\rightarrow$ challenge of finding more appropriate objective functions
  2. *Independence* assumption too restrictive $\rightarrow$ challenge of allowing direct dependencies between a label and past/future observations
  3. Based on *explicit feature representations* and *lack power of kernel* based methods
- *from HMM use*: the Markov chain dependency structure between labels: an undirected graph(chain)
- *from HMM use*: efficient dynamic programming

Useful Properties of HMM and SVM (in general)

- One use of HMM: Given the parameters of the model, find the most likely sequence of hidden states that could have generated a given output sequence.
- This problem is solved by the **Viterbi algorithm** i.e. we find pre image via DP: finds most probable state sequence and reconstruct the path in reverse direction. We have to compute the transition cost and observation cost matrix.
- SVM classifies samples based on the training set of input-output pairs $(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_m, c_m)$, where c_i indicates (1/-1) if $\mathbf{x}_i$ belongs to the class c_i or not.
- Predict label sequences $\mathbf{y}$ instead of c_i:s.

Support Vectors http://www.imtech.res.in/raghava/rbpred/svm.jpg



- The hypothesis space is given by the functions $f(\mathbf{x}) = sng(\mathbf{wx} + b)$, where $\mathbf{w}$ and b are parameters that are learned
- Samples on the margin are the *support vectors*
- Optimization problem: the norm of $\mathbf{w}$ and loss are minimized subject to right classification within the allowed error(s), ξ_i (soft margin)
- Writing the classification rule in its unconstrained *dual form* reveals that classification is only a function of the *support vectors*

Optimizing dual Problem and Non-separability

- Maximum-margin problem: find the weight vector $\mathbf{w}$ that maximizes the minimum margin of the sample.
- Using the standard trick of fixing the functional margin at 1, one can equivalently minimize the squared norm $\|\mathbf{w}\|^2$ subject to the margin constraints.
- Instead of optimizing the *primal problem* optimize the *dual problem*: introduce a Lagrangian multiplier $\alpha_{i\mathbf{y}} \rightarrow$ enforce the margin constraint for label $\mathbf{y} \neq \mathbf{y}_i$ and input $\mathbf{x}_i \rightarrow$ write out $\alpha_{i\mathbf{y}} \rightarrow$ differentiate $\rightarrow$ substitute equations of the primal into Lagrangian results, in a dual QP.
- In order to accommodate for margin violations (a non-separable case) one can generalize SVM formulation: one may add slack variable ξ_i for every training sequence.
- $k((\mathbf{x}_i, \mathbf{y}), (\mathbf{x}_j, \bar{\mathbf{y}})) = \langle \delta\Phi_i(\mathbf{y}), \delta\Phi_j(\bar{\mathbf{y}}) \rangle$ denote the kernel function and can be computed from the inner products involving values of Φ due to the linearity of inner product and validity of k as kernel.

Label Sequence Learning Problem

- Label sequence learning is the problem of inferring a label or state sequence from an observation sequence, where the state sequence may encode a labelling, annotation or segmentation of the sequence
- i.e. predict a sequence of labels $\mathbf{y} = (y^1, ..., y^m)$, $y^k \in \Sigma$, from a given observation sequence $\mathbf{x} = (x^1, ..., x^m)$.
- Map observation vectors $\mathbf{x}$ to some representation in $\Re^d$, which is the observation feature space.
- Map labels $\mathbf{y}$ to some representation in $\Re^d$, which is the output feature space.
- Make a joint feature map for pairs $(\mathbf{x}, \mathbf{y})$.
- We have training pairs $\mathcal{X} \equiv \{(\mathbf{x}_i, \mathbf{y}_i)\}$ and we want to learn a linear discriminant function $F : \mathcal{X} \times \mathcal{Y} \in \Re$ over input/output pairs from which we can derive a prediction by maximizing F over the response variable, $\mathbf{y} \in \mathcal{Y}$, for a specific given input $\mathbf{x}$.

Label Sequence Learning Problem continues ...

- Hence, the general form of hypotheses f is ($\mathbf{w}$ denotes a parameter vector)

$$f(\mathbf{x};\mathbf{w}) = \arg\max_{\mathbf{y}\in\mathcal{Y}} F(\mathbf{x},\mathbf{y};\mathbf{w}).$$

- It might be useful to think of $-F$ as a $\mathbf{w}$-parameterized family of cost functions [2], which we try to design in such a way that the minimum of $F(\mathbf{x},\cdot;\mathbf{w})$ is at the desired output $\mathbf{y}$ for inputs $\mathbf{x}$ of interest.
- We assume F to be linear in some *combined feature representation* of inputs and outputs $\Phi(\mathbf{x},\mathbf{y})$, i.e.

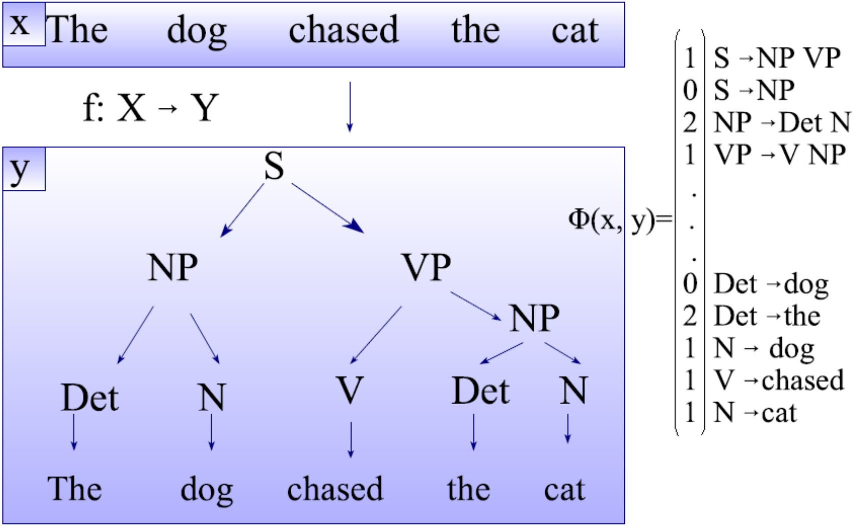
$$F(\mathbf{x},\mathbf{y};\mathbf{w}) = \langle \mathbf{w}, \Phi(\mathbf{x},\mathbf{y})\rangle$$

- **An example** (next two slides) from [2] and [3]:
The goal in natural language parsing is to predict the parse tree $\mathbf{y}$ that generates a given input sentence $\mathbf{x} = (x^1, ..., x^m)$. Each node in the tree $\mathbf{y}$ is generated by a rule of a weighted context-free grammar which is assumed to be in Chomsky normal form.

Label Sequence Learning Example

- $\mathbf{x} = (\mathit{The}, \mathit{dog}, \mathit{chased}, \mathit{the}, \mathit{cat})$ and
- $\mathbf{y}$ is a grammar based **parse tree** with rules g_j e.g. $S \rightarrow NP\ VP$: 'sentence produces noun-part and verb-part'.
- f maps a given sentence $\mathbf{x}$ to a parse tree $\mathbf{y}$.
- The number of all possible productions, d, defines the joint feature space, $\Re^d$. $\Lambda(\mathbf{y})$ and $\Psi(\mathbf{x})$ are indicator vectors of the productions in a tree: if a production is present (1) or not (0).
- Mapping $\Phi(\mathbf{x}, \mathbf{y}) = \Phi_1(\mathbf{x}, \mathbf{y}) + \Phi_2(\mathbf{x}, \mathbf{y})$, represents interdependencies between labels and the nodes of the tree.
- $\Phi(\mathbf{x}, \mathbf{y})$ is a histogram vector counting how often each grammar rule g_j occurs in the tree $\mathbf{y}$.
- $f(\mathbf{x}; \mathbf{w})$ can be efficiently computed by finding the structure $\mathbf{y} \in \mathcal{Y}$ that maximizes $F(\mathbf{x}, \mathbf{y}; \mathbf{w})$ via the CYK (aka CKY, complexity n^3) algorithm.
- Learning over structured output spaces $\mathcal{Y}$ inevitably involves loss functions other than the standard zero one classification loss. A parse tree that differs from the correct parse in a few nodes only should be treated differently from a parse tree that is radically different.

Illustration of Natural Language Parsing Model



Input-Output mappings via Joint Feature Functions

- The general approach we pursue is to learn a $\mathbf{w}$-parametrized *discriminant function*: $F : \mathcal{X} \times \mathcal{Y} \in \Re$ over input/output pairs and to maximize this function over the response variable to make a prediction.

$$f(\mathbf{x}) = \arg\max_{\mathbf{y}\in\mathcal{Y}} F(\mathbf{x}, \mathbf{y}; \mathbf{w}) \tag{1}$$

- In particular, we are interested in a setting, where F is linear in some *combined feature representation* of inputs and outputs $\Phi(\mathbf{x}, \mathbf{y})$ i.e.

$$F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle \tag{2}$$

- We want to apply kernel function to avoid explicit mapping Φ.
- In structured-output prediction:
  only a small fraction of constraints are active
  $+$ *overlap information among classes represented via the joint feature map*
  $\rightarrow$ *maintain working sets S_i* for each instance to keep track of the selected constraints which define the current relaxation. And find the most violated constraint (in $\arg\max$).

Input-Output Mappings continues ...

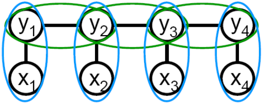
- Kernel functions avoid performing an explicit mapping Φ when this may become intractable. This is possible due to the linearity of the function F, if we have a kernel K over the joint input/output space such that

$$K((\mathbf{x},\mathbf{y}),(\tilde{\mathbf{x}},\tilde{\mathbf{y}})) = \langle \Phi(\mathbf{x},\mathbf{y}), \Phi(\tilde{\mathbf{x}},\tilde{\mathbf{y}}) \rangle \tag{3}$$

  and whenever the optimal function F has a dual representation in terms of an expansion $F(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{m} \alpha_i K((\tilde{\mathbf{x}}_i,\tilde{\mathbf{y}}_i),(\mathbf{x},\mathbf{y})), i = 1, ..., m$, over some finite set of samples $(\tilde{\mathbf{x}}_1,\tilde{\mathbf{y}}_1), ..., (\tilde{\mathbf{x}}_m,\tilde{\mathbf{y}}_m)$
- Extract features not only from the input patterns as in binary classification, but also jointly from input-output pairs.
- The compatibility of an input $\mathbf{x}$ and an output $\mathbf{y}$ may depend on a particular property of $\mathbf{x}$ in conjunction with a particular property of $\mathbf{y}$.
- This is especially relevant, if $\mathbf{y}$ is not simply an atomic label, but has an internal structure that can itself be described by certain features. These features may in turn interact in non-trivial ways.

Hidden Markov Chain Discriminants

- We want to learn mapping f from observation sequences $\mathbf{x} = (x^1, ...x^t, ...)$ to label sequences $\mathbf{y} = (y^1, ...y^t, ...)$, where each label takes values from some label set Σ, i.e. $y^t \in \Sigma$.
- Since for $\mathbf{x}$ we only consider label sequences $\mathbf{y}$ of the same length l_x, the admissible range of $f : \mathcal{X} \times \mathcal{Y}$ is effectively finite
- Output space $\mathcal{Y}$ consists of *all possible label sequences* (its cardinality grows exponentially in the size of $\mathbf{y}$).
- The definition requires a suitable parametric discriminant function F to specify a mapping Φ which extracts features from an observation/label sequence pair $(\mathbf{x}, \mathbf{y})$.
- HMMs suggest to define two types of features, interactions between attributes of the observation vectors and a specific label as well as interactions between neighbouring labels along the chain.
  Don't define a proper joint probability model. Define Φ so that f can be computed from F efficiently, i.e. using *Viterbi-like* decoding algorithm.
  Restrict label-label interactions to nearest neighbours as in HMMs.



Hidden Markov Chain Discriminants: Notations

- Ψ maps observation vectors $\mathbf{x}$ to some representation $\Psi(\mathbf{x}) \in \Re^d$ e.g. $\psi_r(x^s)$ may denote the input feature of specific word like 'rain' occurring in the s-th position in a sentence.
  $[\![y^t = \sigma]\!]$ denotes the indicator function for predicate $y^t = \sigma$ e.g. whether the t-th word is a noun or not.
- Define a set of combined label/observation features via

$$\phi^{st}_{r\sigma}(\mathbf{x}, \mathbf{y}) = [\![y^t = \sigma]\!]\psi_r(x^s),\ 1 \leq r \leq d,\ \sigma \in \Sigma \tag{4}$$

  e.g. $\phi^{st}_{r\sigma} = 1$ would indicate the conjunction of the two predicates: s-th word is 'rain' and t-th word has been labelled as a noun. (In general this may not be binary, but real-valued.) Features $\phi^{st}_{r\sigma}$ conjunctively combine input attributes ψ_r with states σ. For example, if each input is described by L attributes ψ_r and if there are $K = |\Sigma|$ possible states, then one may extract a total of $K \cdot L$ features of this type by combining every input attribute with every state.
- For the second type of features we have inter-label dependencies

$$\bar{\phi}^{st}_{\sigma\tau} = [\![y^s = \sigma \wedge y^t = \tau]\!]\psi_r(x^s), \sigma, \tau \in \Sigma \tag{5}$$

  These features simply count how often a particular combination of labels occur at neighbouring sites.

The Sum of partial Feature Maps

- From previous features a partial feature map $\Phi(\mathbf{x}, \mathbf{y}; t)$ at position t can be defined by selecting appropriate subset of the features $\phi^{st}_{r\sigma}$ and $\bar{\phi}^{st}_{\sigma\tau}$. For example, an HMM only uses input-label features of the type $\phi^{tt}_{r\sigma}$ and label-label features $\bar{\phi}^{t(t+1)}_{\sigma\tau}$, reflecting the first order Markov property of the chain. (1.) Define feature representation of input pattern, (2.) select appropriate window size and (3.) stack together all extracted features at location t.
- I.e. the feature map is extended to sequences $(\mathbf{x}, \mathbf{y})$ of length T in an additive manner (In this way, we can compare sequences of different lengths easily):

$$\Phi(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{T} \Phi(\mathbf{x}, \mathbf{y}; t) \tag{6}$$

- The similarity between two sequences depends on the number of common two-label fragments as well as the inner product between the feature representation of patterns with common label:

$$\langle \Phi(\mathbf{x}, \mathbf{y}), \Phi(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \rangle = \sum_{s,t} [\![y^{s-1} = \bar{y}^{t-1} \wedge y^s = \bar{y}^t]\!] + \sum_{s,t} [\![y^s = \bar{y}^t]\!] k(x^s, \bar{x}^t) \tag{7}$$

HM-SVM: derive a maximum margin formulation for the joint kernel learning setting

Generalize the notion of a separation margin by defining the margin of a training example with respect to a discriminant function, F, as:

$$\gamma_i = F(\mathbf{x}_i, \mathbf{y}_i) - \max_{\mathbf{y} \neq \mathbf{y}_i} F(\mathbf{x}_i, \mathbf{y}) \tag{8}$$

Then the maximum margin problem can be defined as finding a weight vector $\mathbf{w}$ that maximizes $\min_i \gamma_i$.

- Restrict the norm of $\mathbf{w}$ $(= 1)$ or fix the functional margin $(\max_i \gamma_i \geq 1)$; the latter results to quadratic objective

$$\min \frac{1}{2}\|\mathbf{w}\|^2, s.t. F(\mathbf{x}_i, \mathbf{y}_i) - \max_{\mathbf{y} \neq \mathbf{y}_i} F(\mathbf{x}_i, \mathbf{y}) \geq 1, \ \forall i. \tag{9}$$

  Each non-linear constraint in Eq. (9) can be replaced by an equivalent set of linear constraints,

$$F(\mathbf{x}_i, \mathbf{y}_i) - F(\mathbf{x}_i, \mathbf{y}) \geq 1, \ \forall i \ and \ \forall \mathbf{y} \neq \mathbf{y}_i \tag{10}$$

- Introduce an additional threshold θ_i. Function z_i stresses that $(\mathbf{x}_i, \mathbf{y}_i)$ takes the role of positive example and $(\mathbf{x}_i, \mathbf{y})$ for $\mathbf{y} \neq \mathbf{y}_i$ takes the role of negative pseudo-example.

$$z_i(\mathbf{y})(F(\mathbf{x}_i, \mathbf{y}) + \theta_i) \geq \frac{1}{2}, z_i(\mathbf{y}) = \begin{cases} 1 & \text{if } \mathbf{y} = \mathbf{y}_i \\ -1 & \text{otherwise} \end{cases} \tag{11}$$

Hidden Markov SVM continues ...

- The dual formulation of quadratic program

$$\max \mathbf{W}(\alpha) = -\frac{1}{2}\sum_{i,\mathbf{y}}\sum_{j,\bar{\mathbf{y}}}\alpha_i(\mathbf{y})\alpha_j(\bar{\mathbf{y}})z_i(\mathbf{y})z_j(\bar{\mathbf{y}})k_{i,j}(\mathbf{y},\bar{\mathbf{y}}) + \sum_{i,\mathbf{y}}\alpha_i(\mathbf{y}) \quad (12)$$

  s.t. $\alpha_i(\mathbf{y}) \geq 0$, $\forall i = 1, ..., n$, $\forall \mathbf{y} \in \mathcal{Y}$ and $\sum_{\mathbf{y}\in\mathcal{Y}} z_i(\mathbf{y})\alpha_i(\mathbf{y}) = 0, \forall i = 1, ..., n$
  where $k_{i,j}(\mathbf{y},\bar{\mathbf{y}}) = \langle\Phi(\mathbf{x}_i,\mathbf{y}),\Phi(\mathbf{x}_j,\bar{\mathbf{y}})\rangle$
- $\alpha_i(\mathbf{y}) = 0$, if $\alpha_i(\mathbf{y}_i) = 0$, i.e. only if the positive example $(\mathbf{x}_i, \mathbf{y}_i)$ is a support vector, will there be corresponding support vectors created from negative pseudo-examples.

HM-SVM Optimization

- The actual solution might be extremely sparse, since we expect that only very few negative pseudo-examples (which is possibly small subset of $\mathcal{Y}$) will become support vector.
- Design a computational scheme that exploits the anticipated sparseness of the solution
- Since the constraints only couple Lagrange parameters for same training example, the authors propose to optimize W iteratively, at each iteration optimize over the subspace spanned by all $\alpha_i(\mathbf{y})$ for a fixed i.
- If α^* is a solution of the Lagrangian dual problem in Eq. (12), then $\alpha_i^* = 0$ for all pairs $(\mathbf{x}_i, \mathbf{y})$ for which $F(\mathbf{x}_i, \mathbf{y}) < \max_{\bar{\mathbf{y}} \neq \mathbf{y}} F(\mathbf{x}_i, \bar{\mathbf{y}}; \alpha^*)$.
- Define the matrix $D((\mathbf{x}_i, \mathbf{y}), (\mathbf{x}_i, \bar{\mathbf{y}})) \equiv z_i(\mathbf{y}) z_j(\bar{\mathbf{y}}) k_{i,j}(\mathbf{y}, \bar{\mathbf{y}})$. Then $\alpha' D e_i(\mathbf{y}) = z_i(\mathbf{y}) F(\mathbf{x}_i, \mathbf{y})$, where $e_i(\mathbf{y})$ refers to the canonical basis vector corresponding to the dimension of $\alpha_i(\mathbf{y})$.
- Use working set approach to optimize over the i-th subspace that adds at most one negative pseudo-example to the working set at a time. Maximize $W_i(\alpha_i; \{\alpha_j : j \neq i\})$ over arguments α_i while keeping all other α_j's fixed.

HM-SVM Optimization Algorithm

Algorithm 1 Working set optimization for HM-SVMs

$S \leftarrow \{\mathbf{y}_i\}, \alpha_i = 0$
loop
    compute $\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \neq \mathbf{y}_i} F(\mathbf{x}_i, \mathbf{y}; \alpha)$ // a 2-best Viterbi with cost matrices
    if $F(\mathbf{x}_i, \mathbf{y}_i; \alpha) - F(\mathbf{x}_i, \hat{\mathbf{y}}; \alpha) \geq 1$ **then**
        return α_i
    else
        $S \leftarrow S \cup \{\hat{\mathbf{y}}\}$
        $\alpha_i \leftarrow$ optimize W_i over S // SVM optimization
    end if
    for $\mathbf{y} \in S$ **do**
        if $\alpha_i(\mathbf{y}) = 0$ **then**
            $S \leftarrow S - \{\mathbf{y}\}$
        end if
    end for
end loop

Soft Margin HM-SVM

- In the algorithm step 8, we want to introduce slack variable to allow margin violation. A slack variable per training data point, which is shared across all the negative pseudo-examples, is generated. Using either L_2 or L_1 penalties defines which constraints are used in step 8.
- With L_2 we have objective

$$\min \frac{1}{2}\|w\|^2 + \frac{C}{2}\sum_i \xi_i^2 \tag{13}$$

$$\text{s.t. } z_i(\mathbf{y})(\mathbf{w}, \langle \Phi(\mathbf{x}_i, \mathbf{y})\rangle + \theta_i) \geq 1 - \xi_i$$

$$\text{and } \xi_i \geq 0, \forall i = 1, ..., n, \forall \mathbf{y} \in \mathcal{Y}$$

- By solving Lagrangian function for ξ_i we get penalty terms with variables α_i. We can further absorb the penalty into kernel and get

$$K_C((\mathbf{x}_i, \mathbf{y}), (\mathbf{x}_i, \bar{\mathbf{y}})) = \langle \Phi(\mathbf{x}_i, \mathbf{y}), \Phi(\mathbf{x}_i, \bar{\mathbf{y}})\rangle + \frac{1}{C} z_i(\mathbf{y}) z_i(\mathbf{y}') \tag{14}$$

and $K_C((\mathbf{x}_i, \mathbf{y}), (\mathbf{x}_j, \mathbf{y}')) = K((\mathbf{x}_i, \mathbf{y}), (\mathbf{x}_j, \mathbf{y}'))$ for $i \neq j$.

Soft Margin HM-SVM, L_1 penalty

- We have optimization problem

$$\min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i \tag{15}$$

$$\text{s.t. } z_i(\mathbf{y})(\langle \mathbf{w}, \Phi(\mathbf{x}_i,\ \mathbf{y})\rangle + \theta_i) \geq 1 - \xi_i,\ \xi_i \geq 0$$

$$\forall i = 1, ..., n, \forall \mathbf{y} \in \mathcal{Y}$$

- The box constraints on the $\alpha_i(\mathbf{y})$ takes the following form

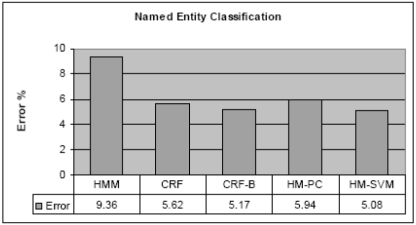
$$0 \leq \alpha_i(\mathbf{y}),\ \mathit{and}\ \sum_{\mathbf{y}\in\mathcal{Y}} \alpha_i(\mathbf{y}) \leq C \tag{16}$$

- Whenever $\xi_i > 0$, $\sum_{\mathbf{y}\in\mathcal{Y}} \alpha_i(\mathbf{y}) = C$. This means that

$$\alpha_i(\mathbf{y}_i) \ = \ \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_i(\mathbf{y}) = C/2 \tag{17}$$

Applications

Named Entity Recognition (NER): find phrases, e.g. containing person, location and organization names. Each entry is annotated with the *type* of its expression and its *position* in the expression, i.e. the beginning or the continuation of the expression. There are 9 labels. In the particular example, there are 34 support sequences, whereas the size of $\mathcal{Y}$ is 16^9.




```
PP ESTUDIA YA PROYECTO LEY TV REGIONAL REMITIDO
O  N       N  N        M   m  m        N

POR LA JUNTA Merida ( EFE ) .
N   N  O     L      N O   N N

ONNNMmmNNNOLNONN
--------M-------
------N---------
----------P-----
----N-----------
N---P-----------
-------m--------
-----------o----
```

(left) Test error of NER task over a window of size 3 using 5-fold cross validation. (right) Example sentence, the correct named entity labelling, and a subset of the corresponding support sequences. Only labels different from the correct labels have been depicted for support sequences. The support sequences with maximal $\alpha_i(\mathbf{y})$ have been selected.

References

1. Yasemin Altun, Ioannis Tsochantaridis, Thomas Hofmann:
Hidden Markov Support Vector Machines. Proc. ICML'2003

2. Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, Yasemin Altun:
Support Vector Machine Learning for Interdependent and Structured Output Spaces, International Conference on Machine Learning (ICML), 2004

3. Predicting Structured Data, MIT Press, 2007