Machine Learning for Structured Prediction

Grzegorz Chrupała

National Centre for Language Technology
School of Computing
Dublin City University

NCLT Seminar

Structured vs Non-structured prediction

In supervised learning we try to learn a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ where $x \in \mathcal{X}$ are inputs and $y \in \mathcal{Y}$ are outputs.

- Binary classification: $\mathcal{Y} = \{-1, +1\}$
- Multiclass classification: $\mathcal{Y} = \{1, \ldots, K\}$ (finite set of labels)
- Regression: $\mathcal{Y} = \mathbb{R}$
- In contrast, in structured prediction elements $\mathcal{Y}$ are *complex*

The prediction is based on the *feature function* $\Phi : \mathcal{X} \rightarrow \mathcal{F}$ where usually $\mathcal{F} = \mathbb{R}^D$ (D-dimensional vector space)

Structured prediction tasks

- Sequence labeling: e.g. POS tagging
- Parsing: given an input sequence, build a tree whose yield (leaves) are the elements in the sequence and whose structure obeys some grammar.
- Collective classification: E.g. relation learning problems, such as labeling web pages given link information.
- Bipartite matching: given a bipartite graph, find the best possible matching. e.g. word alignment in NLP and protein structure prediction in computational biology.

Linear classifiers

- In binary classification, linear classifiers separate classes in a multidimensional input space with a hyperplane
- The classification function giving the output score is

$$y = f(\overrightarrow{x} \cdot \overrightarrow{w} + b) = f(\textstyle\sum_{d \in D} w_d x_d + b)$$

  where
  - $\overrightarrow{x}$ is the input feature vector $= \Phi(x)$,
  - $\overrightarrow{w}$ is a real vector of feature weights,
  - b is the bias term, and
  - f maps the dot product of the two vectors to output (e.g. $f(x) = -1$ if $x < 0$, $+1$ otherwise)
  - The use of *kernels* permits to use linear classifiers for non-linearly separable problems such as the XOR function

Linear classifier – hyperplanes

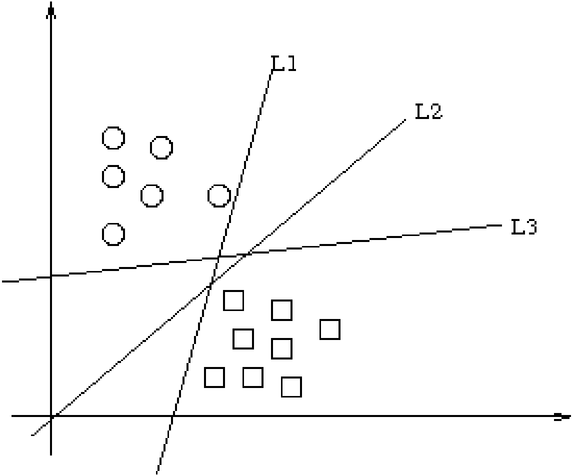

Figure: Three separating hyperplanes in 2-dimensional space (Wikipedia)

Perceptron

- The Perceptron iterates I times through the $1 \ldots n$ training examples, updating the parameters (weights and bias) if the current example n is classified incorrectly:
  - $\overrightarrow{w} \leftarrow \overrightarrow{w} + y_n \overrightarrow{x}_n$
  - $b \leftarrow b + y_n$

  where y_n is the output for the n^{th} training example
- Averaged Perceptron, which generalizes better to unseen examples, does *weight averaging*, i.e. the final weights returned are the average of all weight vectors used during the run of the algorithm

Perceptron Algorithm

Algorithm AVERAGEDPERCEPTRON$(x_{1:N}, y_{1:N}, I)$

$\boldsymbol{w}_0 \leftarrow \langle 0, \ldots, 0 \rangle$, $b_0 \leftarrow 0$
$\boldsymbol{w}_a \leftarrow \langle 0, \ldots, 0 \rangle$, $b_a \leftarrow 0$
$c \leftarrow 1$
for $i = 1 \ldots I$ **do**
    for $n = 1 \ldots N$ **do**
        if $y_n \left[\boldsymbol{w}_0{}^\top \Phi(x_n) + b_0 \right] \leq 0$ **then**
            $\boldsymbol{w}_0 \leftarrow \boldsymbol{w}_0 + y_n \Phi(x_n)$, $b_0 \leftarrow b_0 + y_n$
            $\boldsymbol{w}_a \leftarrow \boldsymbol{w}_a + c y_n \Phi(x_n)$, $b_a \leftarrow b_a + c y_n$
        end if
        $c \leftarrow c + 1$
    end for
end for
return $(w_0 - w_a/c, b_0 - b_a/c)$

Daume III (2006)

Structured Perceptron

- For structured prediction the feature function is $\Phi : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{F}$ (second argument is the hypothesized output)
- Structured prediction algorithms need Φ to admit tractable search
- The argmax problem:

$$\hat{y} = \underset{y \in \mathcal{Y}}{argmax}\ \overrightarrow{w} \cdot \Phi(x, y)$$

- Structured Perceptron: for each example n wherever the predicted $\hat{y}_n$ for x_n differs from y_n the weights are updated:
  - $\overrightarrow{w} \leftarrow \overrightarrow{w} + \Phi(x_n, y_n) - \Phi(x_n, \hat{y}_n)$
- Similar to plain Perceptron — key difference: $\hat{y}$ is calculated using the arg max.
- Incremental Perceptron is a variant used in cased where arg max isn't available — a beam search algorithm is used instead

Logistic regression / Maximum Entropy

- Conditional probability of class y is proportional to $exp\ f(x)$

$$
\begin{aligned}
p(y|x; \overrightarrow{w}, b) &= \frac{1}{Z_{x;\overrightarrow{w},b}} exp\ [y(\overrightarrow{w} \cdot \Phi(x) + b)] \\
&= \frac{1}{1 + exp\ [-2y(\overrightarrow{w} \cdot \Phi(x) + b)]}
\end{aligned}
$$

- For training we try to find $\overrightarrow{w}$ and b which maximize the likelihood of training data
- Gaussian prior used to penalize large weights
- Generalized Iterative Scaling algorithm used to solve the maximization problem

Maximum Entropy Markov Models

- MEMMs are an extension of MaxEnt to sequence labeling
- Replace $p(observation|state)$ with $p(state|observation)$
- For first-order MEMM, the conditional distribution on the label y_n given full input x, the previous label y_{n-1}, feature function Φ and weights $\overrightarrow{w}$ is:

$$p(y_n|x, y_{n-1}; \overrightarrow{w}) = \frac{1}{Z_{x;y_{n-1}\overrightarrow{w}}} exp\left[\overrightarrow{w} \cdot \Phi(x, y_n, y_{n-1})\right]$$

- When MEMM is trained *true* label y_{n-1} is used
- At prediction time, Viterbi is applied to solve arg max
- *Predicted* y_{n-1} label is used in classifying n^{th} element of the sequence

Conditional Random Fields

- Alternative generalization of MaxEnt to structured prediction
- Main difference:
  - while MEMM uses per state exponential models for the conditional probabilities of next states given the current state
  - CRF has a single exponential model for the joint probability of the whole sequence of labels
- The formulation is:

$$p(y|x;\overrightarrow{w}) = \frac{1}{Z_{x;\overrightarrow{w}}} exp\left[\overrightarrow{w} \cdot \Phi(x,y)\right]$$

$$Z_{x;\overrightarrow{w}} = \sum_{y' \in \mathcal{Y}} exp\left[\overrightarrow{w} \cdot \Phi(x,y')\right]$$

where elements $y' \in \mathcal{Y}$ are incorrect outputs

CRF - experimental comparison to HMM and MEMMs

model	*error*	*oov error*
HMM	5.69%	45.99%
MEMM	6.37%	54.61%
CRF	5.55%	48.05%
MEMM^{+}	4.81%	26.99%
CRF^{+}	4.27%	23.76%

$^{+}$Using spelling features

Figure 4. Per-word error rates for POS tagging on the Penn tree-bank, using first-order models trained on 50% of the 1.1 million word corpus. The oov rate is 5.45%.

J. Lafferty, A. McCallum, F. Pereira - Proc. 18th International Conf. on Machine Learning, 2001

Large Margin Models

- Choose parameter settings which maximize the distance between the hyperplane separating classes and the nearest data points on either side.
- Support Vector Machines. For data separable with margin 1:

$$\mathit{minimize}_{\overrightarrow{w},b} \quad \frac{1}{2}\|\overrightarrow{w}\|^2$$

subject to

$$\forall n, \quad y_n[\overrightarrow{w} \cdot \Phi(x_n) + b] \geq 1$$

SVM

- Soft-margin formulation: don't enforce perfect classification of training examples
- The *slack* variables ξ measures how far an example is from the correct hard margin constraint

$$minimize_{\overrightarrow{w},b} \quad \frac{1}{2}\|\overrightarrow{w}\|^2 + C\sum_{n=1}^{N}\xi_n$$

subject to

$$\forall n, \quad y_n[\overrightarrow{w} \cdot \Phi(x_n) + b] \geq 1 - \xi_n$$
$$\xi_n \geq 0$$

- The parameter C trades off fitting the training data and a small weight vector

Margins and Support Vectors

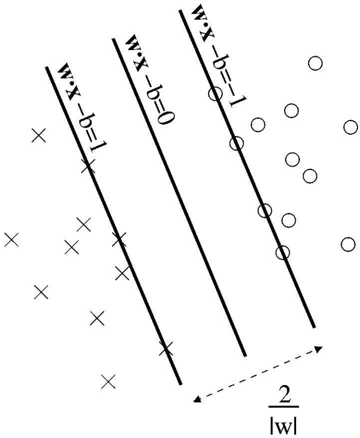

Figure: SVM maximum-margin hyperplanes for binary classification. Examples along the hyperplanes are support vectors (from Wikipedia)

Maximum Margin models for Structured Prediction

- Maximum Margin Markov Networks (M^3N)
- The difference in score (under loss function l) between true output y and any incorrect output $\hat{y}$ is at least $l(x, y, \hat{y})$ — M^3N scales the margin to be proportional to loss.
- Support Vector Machines for Interdependent and Structured Outputs: SVM^{struct}
- Instead of scaling the margin, scale the slack variables by the loss
- The two approaches differ in the optimization techniques used (SVM^{struct} constrains the loss function less but is more difficult to optimize).

Summary of structured prediction algorithms

	Loss			Features			Efficient	Easy to Implement
	0/1	Hamming	Any	argmax and sum	argmax only	Neither		
Structured Perceptron	√				√		√	√
Conditional Random Field	√			√				–
Max-margin Markov Network	√	√			√			
SVM for Structured Outputs	√	√			√			
Reranking	√	√	√			√	–	–

Figure from Daume III (2006)

Other approaches to structured outputs

- Searn (Search + Learn)
- Analogical learning
- Miscellaneous hacks:
  - Reranking
  - Stacking
  - Class n-grams

References

- M. Collins, 2002, Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms, EMNLP-02
- McCallum, Andrew, Dayne Freitag, and Fernando Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. ICML
- J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random elds: Probabilistic models for segmenting and labeling sequence data. ICML
- B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In Advances in Neural Information Processing Systems (NIPS).
- I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. 2005. Large margin methods for structured and interdependent output variables. JMLR
- Hal Daumé III, 2006, *Practical Structured Learning Techniques for Natural Language Processing*, PhD thesis