Linear models for structure prediction

Fernando C. N. Pereira

Department of Computer and Information Science
University of Pennsylvania, Philadelphia, Pennsylvania, USA
pereira@cis.upenn.edu

10.21437/Interspeech.2005-2

Abstract

Over the last few years, several groups have been developing models and algorithms for learning to predict the structure of complex data, sequences in particular, that extend well-known linear classification models and algorithms, such as logistic regression, the perceptron algorithm, and support vector machines. These methods combine the advantages of discriminative learning with those of probabilistic generative models like HMMs and probabilistic context-free grammars. I will introduce linear models for structure prediction and their simplest learning algorithms, and exemplify their benefits with applications to text and speech processing, including information extraction, parsing, and language modeling.

1. The Problem

The noisy-channel metaphor dominated statistical pattern recognition methods in speech and language for thirty years. Building on advances in information theory and cryptology, it was natural to think of speech recognition and language translation as decoding problems in we are given an encoded and possibly corrupted message $\boldsymbol{x}$ and seek to find the corresponding plaintext $\boldsymbol{y}$. The well-known noisy-channel formulation for these problems starts from a simple use of the Bayes rule

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{p(\boldsymbol{x},\boldsymbol{y})}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x}|\boldsymbol{y})p(\boldsymbol{y})}{p(\boldsymbol{x})} \tag{1}$$

where $p(\boldsymbol{x}|\boldsymbol{y})$ describes the (stochastic) encoding process and $p(\boldsymbol{y})$ gives the distribution of plaintexts to be encoded. Building a noisy-channel model involves estimating these two distributions. The corresponding decoder is then

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x}) = \arg\max_{\boldsymbol{y}} p(\boldsymbol{x}|\boldsymbol{y})p(\boldsymbol{y}) \tag{2}$$

In language problems, both $\boldsymbol{x}$ and $\boldsymbol{y}$ are structured objects of arbitrary size (speech waveforms, written sentences). Therefore, $p(\boldsymbol{x}|\boldsymbol{y})$ and $p(\boldsymbol{y})$ have to be specified as functions of the parts of $\boldsymbol{x}$ and $\boldsymbol{y}$ and how they are put together. The most convenient form for such specifications is that of *generative models*. In a generative model, the probability of a complex object is given as the product of the probabilities of conditionally independent generative steps that build the object incrementally from parts. The most familiar examples are Markov models for language modeling, hidden Markov models (HMMs) for speech, and probabilistic context-free grammars (PCFGs) for text.

The conditional independence assumption in generative models restricts the accuracy of noisy channel models. For example, an HMM model for a speech unit assumes conditional independence of successive observations of the speech signal given the state of the model. This assumption is grossly violated by actual speech, even when observation decorrelation techniques are used. As another example, PCFG models of natural language cannot model correlations between words that are separated by a nonterminal in the parse tree.

These restrictions of generative models are compensated at least in part by ease of training a model to maximize (with appropriate regularization) the probability of a set of $(\boldsymbol{x},\boldsymbol{y})$ pairs, for example, spoken utterances and their transcriptions. In the speech case, the language model $p(\boldsymbol{y})$ may be a Markov model, and training consists of estimating transition probabilities from empirical transition counts. This is not trivial because of the number of transitions in the model may be large relative to the available training data, leading to sparse data problems, but it is fairly well understood. For $p(\boldsymbol{x}|\boldsymbol{y})$, the situation may be a bit more complicated. In speech again, the segmentation of the speech signal $\boldsymbol{x}$ is not known, so the relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$ involves hidden choices of segmentation, requiring an appropriate version of the expectation-maximization method to estimate model parameters. Nevertheless, the methods for both estimation tasks are well understood, and with the help of suitable approximations, they can applied effectively to the largest problems.

However, one might legitimately ask whether the mismatch between the actual statistics of a problem and the independence assumptions of generative models can be alleviated. One might also ask whether maximizing the probability of a the training data is the right training criterion. This second question has been addressed in speech recognition, for instance, with alternative training criteria like maximum mutual information (MMI) that focus on optimal discrimination among possible decodings $\boldsymbol{y}$. For the first question, generative models with richer dependency structures, such as dynamic Bayes nets, are able to better model the distribution of $(\boldsymbol{x},\boldsymbol{y})$ pairs. However, the computation in (2) is in general intractable for those models, as is exact training, requiring approximate inference and training methods whose accuracy and computational properties are difficult to quantify.

For a simple illustration of these issues, consider the problem of part-of-speech (POS) tagging, in which $\boldsymbol{x}$ is a word sequence and $\boldsymbol{y}$ is the corresponding sequence of parts of speech. POS tagging is a useful pre-processing stage for language processing tasks such as parsing, information extraction, and speech synthesis. Early automatically-trained POS taggers used generative HMM-like models such as

$$p(\boldsymbol{x},\boldsymbol{y}) = p(y_1)p(x_1|y_1)\prod_{i=2}^{n} p(y_i|y_{i-1})p(x_i|y_i) \tag{3}$$

Models like this assume conditional independence between successive words given their parts of speech, which is clearly un-

realistic. Furthermore, taggers of this form were soon superseded in accuracy by discriminatively-trained POS taggers, in which a classifier is trained to predict the POS tag y_i of word x_i from features of neighboring words and previously predicted POS tags. For instance, we might train discriminatively a probabilistic classifier (for example, a maximum-entropy classifier) $p(y_i|y_{i-1}, \boldsymbol{x}, i)$ that estimates the probability of the ith tag conditioned on the preceding tag and features of some appropriate window on $\boldsymbol{x}$ around position i. Such a model gives us an estimate

$$p(\boldsymbol{y}|\boldsymbol{x}) = p(y_1|\text{start}, \boldsymbol{x}, 1) \prod_{i=2} p(y_i|y_{i-1}, \boldsymbol{x}, i) \tag{4}$$

from which $\arg\max_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x})$ can be easily computed by Viterbi decoding.

One important advantage of models like (4) [1, 2] is that, unlike generative models, they make no assumptions about the input distribution, and can depend on arbitrary features of the input, leading to improved discrimination. Unfortunately, something is also lost relative to generative models. In contrast to an HMM, a discriminative sequence model of the above form suffers from *label bias* in that it cannot be "surprised" by later inputs incompatible with an earlier decision [3]. In other words, the model is unable to balance earlier decisions against later decisions in seeking the most likely decoding of an input sequence.

2. Conditional Random Fields

The previous discussion identified two problems with the noisy-channel metaphor and generative models, but also recognized that the obvious way of solving those problems with discriminative classifiers is suffers from another problem, label bias. Fortunately, there is a way of combining the good aspects of generative and discriminative models that solves the three problems simultaneously. The main insight is to view the decoding problem as a single classification problem in which a whole decoding $\boldsymbol{y}$ is selected for the observation sequence $\boldsymbol{x}$. For concreteness, consider the standard log-linear formulation of a maximum entropy classifier

$$p(\boldsymbol{y}|\boldsymbol{x}; \mathbf{w}) = \frac{\exp s(\boldsymbol{x}, \boldsymbol{y}; \mathbf{w})}{Z(\boldsymbol{x}; \mathbf{w})} \tag{5}$$

$$s(\boldsymbol{x}, \boldsymbol{y}; \mathbf{w}) = \mathbf{w} \cdot \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}) \tag{6}$$

$$Z(\boldsymbol{x}; \mathbf{w}) = \sum_{\boldsymbol{y} \in Y(\boldsymbol{x})} \exp s(\boldsymbol{x}, \boldsymbol{y}; \mathbf{w}) \tag{7}$$

with corresponding decoding problem

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} s(\boldsymbol{x}, \boldsymbol{y}; \mathbf{w}) \tag{8}$$

where $Y(\boldsymbol{x})$ is the set of possible outputs for $\boldsymbol{x}$. The quantity $s(\boldsymbol{x}, \boldsymbol{y}; \mathbf{w})$ is called the *score* of the decoding $\boldsymbol{y}$ of input $\boldsymbol{x}$, under the given weight vector. The function $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y})$ computes a vector of informative features about the pair $(\boldsymbol{x}, \boldsymbol{y})$. For example, in POS tagging, a feature may count how many times a noun tag follows the word *the*, another one how many times the suffix *-ing* occurs with a verb tag. Crucially, unlike in generative models, we do not need to worry about the conditional independence of observation features.

Given a training set $\mathcal{T} = \{(\boldsymbol{x}_k, \boldsymbol{y}_k)\}_{k=1}^N$, the above model can be learned by maximizing the (concave) log-likelihood function

$$\begin{aligned} \mathcal{L}_{\mathbf{w}} &= \textstyle\sum_k \log p(\boldsymbol{y}_k|\boldsymbol{x}_k; \mathbf{w}) \\ &= \textstyle\sum_k \left[s(\boldsymbol{x}_k, \boldsymbol{y}_k; \mathbf{w}) - \log Z(\boldsymbol{x}_k; \mathbf{w}) \right] \end{aligned} \tag{9}$$

whose gradient is

$$\nabla \mathcal{L}_{\mathbf{w}} = \sum_k \left[\boldsymbol{F}(\boldsymbol{x}_k, \boldsymbol{y}_k) - E_{p(\boldsymbol{y}|\boldsymbol{x}_k; \mathbf{w})} \boldsymbol{F}(\boldsymbol{x}_k, \boldsymbol{y}) \right] \tag{10}$$

At first sight, this seems a backward step. The number of possible $\boldsymbol{y}$ sequences is exponential on the size of $\boldsymbol{x}$, but computing the normalization in (7), the highest-scoring $\boldsymbol{y}$ in the decoder (8), or the expectation $E_{p(\boldsymbol{y}|\boldsymbol{x}_k; \mathbf{w})} \boldsymbol{F}(\boldsymbol{x}_k, \boldsymbol{y})$ in (10) requires summing over the exponentially many possible $\boldsymbol{y}$s. However, the problem becomes radically easier if the feature function $\boldsymbol{F}$ is a sum of terms each involving only k consecutive labels $y_{i-k+1} \cdots y_i$. For a first-order model, we would have

$$\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \mathbf{f}_i(y_{i-1}, y_i, \boldsymbol{x}) \tag{11}$$

for appropriate feature functions $\mathbf{f}_i$. Then, all of the sums over $\boldsymbol{y}$ above can be calculated by dynamic programming: Viterbi for decoding, and a variant of the forward-backward algorithm for the expectations and normalization. Any model of this form can be seen as a Markov random field (MRF) with potentials on selected subsets of $\boldsymbol{y}$. These potentials also depend on $\boldsymbol{x}$, but since the the model is conditioned on $\boldsymbol{x}$, we do not need to include elements of $\boldsymbol{x}$ in the cliques of the MRF. We have called these models *conditional random fields* (CRFs) [3].

CRFs have been used for a variety of language processing tasks, including tagging, shallow parsing [4], information extraction [5, 6, 7], text similarity modeling and normalization [8], language modeling [9], pitch accent prediction [10], and spoken sentence boundary detection [11]. All of these tasks benefit from using rich sets of non-independent features in the feature function $\boldsymbol{F}$.

In practice, the main cost in training CRFs is the repeated computation of feature expectations for estimating gradients (10) when maximizing the log-likelihood (9). This can be very expensive if the set of possible tags is large, since the complexity of forward-backward is $O(t^{k+1}n)$ where t is the number of possible tags (values of y_i) and k is the order of the model.

3. Linear Structure Models

Looking at the decoding rule (8), we see that it makes no reference to the log-linear probability model and its normalization. In fact, the decoding rule looks like a multiclass extension of a standard linear discriminant [12]. One may then ask whether the extensive theory of linear classification could be an alternative to the probabilistic log-linear formulation of the previous section. For binary classification, we can see logistic regression, the perceptron algorithm, and support-vector machines (SVMs) as different ways of learning a linear separator between positive and negative examples. CRFs are the structured output counterpart of logistic regression. What are the structured output counterparts of the perceptron and of SVMs? For the perceptron, the answer is very simple [13]. For each training example $(\boldsymbol{x}_k, \boldsymbol{y}_k)$, we decode using the current weight vector to find the proposed best decoding

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} s(\boldsymbol{x}_k, \boldsymbol{y}; \mathbf{w}) \tag{12}$$

If $\hat{\boldsymbol{y}} \neq \boldsymbol{y}_k$, that is, there are errors in the decoding, the weight vector is updated

$$\mathbf{w}' = \mathbf{w} + \boldsymbol{F}(\boldsymbol{x}_k, \boldsymbol{y}_k) - \boldsymbol{F}(\boldsymbol{x}_k, \hat{\boldsymbol{y}}) \tag{13}$$

$\mathbf{w}_0 = \mathbf{0};\ j = 0$
for $t = 1, \ldots, T$:
  for $k = 1, \ldots, N$:
    $\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} s(\boldsymbol{x}_k, \boldsymbol{y}; \mathbf{w}_j)$
    if $\hat{\boldsymbol{y}} \neq \boldsymbol{y}_k$:
      $\mathbf{w}_{j+1} = \mathbf{w}_j + \boldsymbol{F}(\boldsymbol{x}_k, \boldsymbol{y}_k) - \boldsymbol{F}(\boldsymbol{x}_i, \hat{\boldsymbol{y}})$
    else :
      $\mathbf{w}_{j+1} = \mathbf{w}_j$
    $j \leftarrow j + 1$
$\mathbf{w} = \sum_j \mathbf{w}_j / NT$

Figure 1: Averaged perceptron algorithm

This training procedure can be improved by averaging successive weight vectors [13]. Figure 1 shows an averaged training algorithm for linear structure models.

The main advantage of perceptron training is that is requires only a decoding method. When the set of possible tags is large, it is easy to use beam search instead of full decoding. This would be harder for CRF training, because pruning the forward-backard lattice used to compute expectation can cause very poor gradient estimation. Perceptron training typically reaches good accuracy with many fewer passes over the data than CRF training. On the negative side, models trained by perceptron are usually slightly less accurate than those trained as CRFs [4], and training may fail to converge if the training data is far from separable.

The techniques of support-vector machines can also be extended to structured outputs. The basic idea is to choose weights that separate the score of the correct output $\boldsymbol{y}$ for each training input $\boldsymbol{x}$ from the scores of all incorrect outputs $\boldsymbol{y}'$ by a large *margin* proportional to the *loss* $L(\boldsymbol{y}, \boldsymbol{y}'; \boldsymbol{x})$ of $\boldsymbol{y}'$ relative to $\boldsymbol{y}$. The loss function L quantifies the errors of the hypothesis $\boldsymbol{y}'$ relative to the truth $\boldsymbol{y}$ in an application-appropriate way. For example, in tagging tasks the loss is typically the number of wrong tags, called the Hamming loss by obvious analogy with the Hamming distance. However, if we cared more about certain errors than others, we could use a loss function that weighs different errors appropriately.

Using a standard derivation from support-vector machines [14, 15], maximizing the training set margins is equivalent to minimizing the norm of the weight vector subject to margin constraints [16]

$$\begin{aligned} &\min_{\mathbf{w}} \|\mathbf{w}\| + C \textstyle\sum_k \xi_k \quad \text{such that} \\ &\quad s(\boldsymbol{x}_k, \boldsymbol{y}_k; \mathbf{w}) - s(\boldsymbol{x}_k, \boldsymbol{y}'; \mathbf{w}) \geq L(\boldsymbol{y}_k, \boldsymbol{y}'; \boldsymbol{x}) - \xi_k \\ &\quad \forall (\boldsymbol{x}_k, \boldsymbol{y}_k) \in \mathcal{T}, \forall \boldsymbol{y}' \in Y(\boldsymbol{x}_k) \end{aligned} \tag{14}$$

where the ξ_k are *slack variables* allowing some degree of constraint violation, and C trades off parameter margin maximization against constraint violation.

As for CRFs, (14) is intractable unless we can factor efficiently the enumeration over $\boldsymbol{y}' \in Y(\boldsymbol{x})$. Remarkably, this is possible in the max-margin Markov network (M^3N) formulation [16], which cleverly exploits the belief propagation equations underlying the forward-backward algorithm. The details are too intricate to go into here, but the main intuition is that the exponential number of constraints in (14) can be replaced by a polynomial number of constraints involving belief propagation messages.

The M^3N approach has interesting theoretical generalization guarantees that generalize those for SVMs, and has been shown to achieve better accuracy than CRF training in experiments [16]. Like SVM, M^3N extends easily to dual kernel representations, which are more flexible than feature representations. CRFs can also used with kernels [17], but training computations are more challenging because the dual optimization problem is not sparse as it is for SVMs.

On the negative side, M^3N has not yet been shown to scale to the large learning problems that are common in text and speech. As of this writing, the largest problem so far attempted with this method has been the discriminative training of an English parser from a subset of the Penn treebank [18]. The accuracy improvements were promising, but the computational cost precluded experimentation with the full treebank.

Fortunately, it is possible to achieve most if not all of the benefits of margin maximization with a simple approximation of (14). Instead of constraints for all the alternative outputs $\boldsymbol{y}' \in Y(\boldsymbol{x})$, we can consider one training example at a time, and attempt to satisfy m constraints for the m best decodings under the current weight vector, for a small, empirically chosen m. This idea leads to an online training algorithm based on the MIRA method for online margin maximization [19, 20], with the following update (leaving aside slack variables for simplicity):

$$\begin{aligned} \mathbf{w}' \quad = \quad &\min_{\hat{\mathbf{w}}} \|\hat{\mathbf{w}} - \mathbf{w}\| \quad \text{such that} \\ &s(\boldsymbol{x}, \boldsymbol{y}; \hat{\mathbf{w}}) - s(\boldsymbol{x}, \boldsymbol{y}'; \hat{\mathbf{w}}) \geq L(\boldsymbol{y}_t, \boldsymbol{y}'; \boldsymbol{x}) \\ &\forall \boldsymbol{y}' \in \text{best}_m(\boldsymbol{x}; \mathbf{w}) \end{aligned} \tag{15}$$

where $\text{best}_m(\boldsymbol{x}; \mathbf{w})$ is the set of m best decodings of $\boldsymbol{x}$ under the weights $\mathbf{w}$. Efficient algorithms for m-best decoding are well known. Intuitively, (15) projects $\mathbf{w}$ onto the subspace defined by the constraints.

Experiments with the MIRA m-best method (under submission), on information extraction, shallow parsing, and handwriting recognition problems show that this very simple method achieves faster training and competitive accuracy with CRF and M^3N training. We have also used this method for discriminative training of dependency parsers from dependency treebanks with excellent results [21].

4. Future Directions

Direct discriminative training of linear models for sequence decoding problems overcome limitations of generative models in the noisy-channel decoding framework for a wide variety of text processing and a growing range of speech processing applications. There is a price to pay in computationally more demanding training, although online training methods like perceptron and m-best MIRA improve this significantly, as do other methods under current investigation [22]. The methods discussed here also have interesting extensions to more general undirected graphical models among the output labels [23, 24].

5. Acknowledgments

This survey summarizes joint research with Koby Crammer, John Lafferty, Andrew McCallum, Ryan McDonald, and Fei Sha, partially funded by NSR ITR grants 0205456, 0205448, and 0428193. I have also benefited from discussions on these topics with Michael Collins, David McAllester, Charles Sutton, Ben Taskar, and Hanna Wallach.

6. References

[1] A. Ratnaparkhi, "A maximum entropy model for part-of-speech tagging," in *Proc. EMNLP*. New Brunswick, New Jersey: ACL, 1996.

[2] A. McCallum, D. Freitag, and F. Pereira, "Maximum entropy Markov models for information extraction and segmentation," in *Proc. ICML 2000*, Stanford, California, 2000, pp. 591–598.

[3] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. ICML-01*, 2001, pp. 282–289.

[4] F. Sha and F. Pereira, "Shallow parsing with conditional random fields," in *Proceedings of HLT-NAACL 2003*. Association for Computational Linguistics, 2003, pp. 213–220.

[5] F. Peng and A. McCallum, "Accurate information extraction from research papers using conditional random fields," in *Proceedings of Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT/NAACL-04)*, 2004.

[6] R. McDonald and F. Pereira, "Identifying gene and protein mentions in text using conditional random fields," *BMC Bioinformatics*, vol. 6, no. Suppl 1, p. S6, 2005.

[7] S. Sarawagi and W. W. Cohen, "Semi-markov conditional random fields for information extraction," in *Advances in Neural Information Processing Systems 17 (NIPS 2004)*, 2005.

[8] A. McCallum, K. Bellare, and F. Pereira, "A conditional random field for discriminatively-trained finite-state string edit distance," in *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI 2005)*, 2005.

[9] B. Roark, M. Saraclar, M. Collins, and M. Johnson, "Discriminative language modeling with conditional random fields and the perceptron algorithm." in *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004),*, 2004.

[10] M. L. Gregory and Y. Altun, "Using conditional random fields to predict pitch accents in conversational speech," in *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, 2004.

[11] Y. Liu, A. Stolcke, E. Shriberg, and M. Harper, "Using conditional random fields for sentence boundary detection in speech," in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Ann Arbor, Michigan: Association for Computational Linguistics, June 2005, pp. 451–458. [Online]. Available: http://www.aclweb.org/anthology/P/P05/P05-1056

[12] K. Crammer and Y. Singer, "Ultraconservative online algorithms for multiclass problems," *Journal of Machine Learning Research*, 2003.

[13] M. Collins, "Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms," in *Proc. of EMNLP*, 2002.

[14] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[15] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Jornal of Machine Learning Research*, vol. 2, pp. 265–292, 2001.

[16] B. Taskar, C. Guestrin, and D. Koller, "Max-margin Markov networks," in *Proc. of NIPS*, 2003.

[17] J. Lafferty, Y. Liu, and X. Zhu, "Kernel conditional random fields: Representation, clique selection, and semi-supervised learning," in *International Conference on Machine Learning*, 2004.

[18] B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning, "Max-margin parsing," in *Proc. of EMNLP*, 2004.

[19] K. Crammer and Y. Singer, "A new family of online algorithms for category ranking," *Jornal of Machine Learning Research*, vol. 3, pp. 1025–1058, 2003.

[20] ——, "Ultraconservative online algorithms for multiclass problems," *Jornal of Machine Learning Research*, vol. 3, pp. 951–991, 2003.

[21] R. McDonald, K. Crammer, and F. Pereira, "Online large-margin training of dependency parsers," in *43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, 2005.

[22] C. Sutton and A. McCallum, "Piecewise training of undirected models," in *21st Conference on Uncertainty in Artificial Intelligence*, 2005.

[23] B. Taskar, P. Abbeel, and D. Koller., "Discriminative probabilistic models for relational data," in *Eighteenth Conference on Uncertainty in Artificial Intelligence*, 2002.

[24] C. Sutton, K. Royanimanesh, and A. McCallum, "Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data," in *ICML 04*, 2004.