Propagating errors and beliefs for large-scale nonlinear structure prediction

Roland Memisevic ROLAND@CS.TORONTO.EDU
Department of Computer Science, University of Toronto, Toronto ON M5S3G4 Canada

Abstract

Nonlinear extensions to structured supervised learning can be achieved by replacing the common *linear* compatibility score with a nonlinear function, such as a neural network. A potential advantage of such a nonlinear variant is that it can, in contrast to kernel based approaches, be trained on very large datasets, because computational complexity scales only linearly with the number of examples.

1. Introduction

Supervised learning is one of the most successful paradigms in machine learning, because it allows us to construct complex systems entirely 'by example'. In particular, recent extensions (e.g. (Lafferty et al., 2001)) of standard supervised learning show that supervised methods can be successfully used to construct systems that produce complex responses to given inputs, and solve problems for which previously generative models have been mainly used.

Common to practically all these recent extensions is that they are *linear*: Given an input vector $\boldsymbol{x}$ they measure the compatibility with a potential output vector $\boldsymbol{y}$ using a linear *score* $c(\boldsymbol{x}, \boldsymbol{y}) := \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}, \boldsymbol{y})$, where $\phi()$ is a (usually hand-crafted) feature vector, containing features on which the relation between $\boldsymbol{x}$ and $\boldsymbol{y}$ is assumed to depend. The models are trained by adapting $\boldsymbol{w}$ so as to maximize an optimality criterion, using a training set $\{(\boldsymbol{x^i}, \boldsymbol{y^i})\}_{i=1:l}$. A common criterion is the maximum likelihood criterion:

$$L(\boldsymbol{w}) = \sum_i \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}^i, \boldsymbol{y}^i) - \log \sum_{\boldsymbol{y}} \exp(\boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}^i, \boldsymbol{y})),$$

which is simply the average log-probability under a probabilistic output-model defined as:

$$p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{w}) = \frac{\exp(\boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}, \boldsymbol{y}))}{\sum_{\boldsymbol{y}} \exp(\boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}, \boldsymbol{y}))}. \quad (1)$$

After training, the output for a new input x^{test}, can be computed as the maximizer of $p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{w})$:

$$\boldsymbol{y^{\mathbf{test}}} = \arg\max_{\boldsymbol{y}} \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}^{\mathrm{test}}, \boldsymbol{y}). \quad (2)$$

Both the sum over $\boldsymbol{y}$ in Eq. 1 and the arg-max in Eq. 2 are intractable, if $\mathbf{y}$ is reasonably high-dimensional. However, we can obtain tractable problems, by letting the feature vector $\phi(\boldsymbol{x}, \boldsymbol{y})$ decompose into the sum of 'sub-feature' vectors, defined on subsets of the components of $\boldsymbol{y}$. In sequence prediction models, we could set, for example:

$$\phi(\boldsymbol{x}, \boldsymbol{y}) = \sum_t \phi_t(\boldsymbol{x}, y_{t-1}, y_t), \quad (3)$$

where $\phi_t(\boldsymbol{x}, y_{t-1}, y_t)$ is a sub-feature that depends only on the components y_{t-1}, y_t of $\boldsymbol{y}$. Due to linearity, the compatibility $\boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}, \boldsymbol{y})$ then decomposes as:

$$\sum_t \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}, y_{t-1}, y_t). \quad (4)$$

The 'arg-max' in Eq. 2, in turn, can be 'pushed' into the sum, and dynamic programming can be used to compute it; similarly for the sum in Eq. 1.

The linear model can be generalized to perform nonlinear predictions, by realizing that the optimal model can typically be written as a kernel expansion (see e.g. (Lafferty et al., 2004)). A drawback of kernel methods in general, but in particular in this context, is computational complexity: These methods scale quadratically with the number of training data-points (and quadratically with *both*, the number of training points and the number of sequence elements in the case of sequence prediction), and are therefore not applicable to large data-sets. In the following we describe an alternative based on neural networks, that scales only linearly with the number of data-points.

Appearing in *Proceedings of the First North East Student Colloquium on Artificial Intelligence*, Ithaca, NY, 2006.

2. Additivity vs. linearity

The reason that the feature decomposition (Eq. 3) leads to a tractable model is not actually the linearity of the score, but rather *additivity*: The fact that the score decomposes as the *sum* of 'sub-scores' (Eq. 4) defined on components of $\boldsymbol{y}$ is what makes dynamic programming possible. This observation suggests a nonlinear variant of the standard model, in which we define the score as the sum of (possibly nonlinear) sub-scores as follows: We use an arbitrary nonlinear function $f_{\boldsymbol{w}}(\phi_t(\boldsymbol{x}, y_{t-1}, y_t))$ and define the overall score as

$$c(\boldsymbol{x}, \boldsymbol{y}) = \sum_t f_{\boldsymbol{w}}(\phi_t(\boldsymbol{x}, y_{t-1}, y_t)) \tag{5}$$

The log-likelihood criterion then still decouples as

$$\begin{aligned} L(\boldsymbol{w}) &= \sum_i \sum_t f_{\boldsymbol{w}}(\phi(\boldsymbol{x}^i, y^i_{t-1}, y^i_t)) \\ &\quad - \log \sum_y \exp(\sum_t f_{\boldsymbol{w}}(\phi(\boldsymbol{x}^i, y_{t-1}, y_t))). \end{aligned}$$

The function $f_w()$ is a nonlinear substitute for $\boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}, y_{t-1}, y_t)$ used above. It can be an arbitrary (scalar-valued) nonlinear function defined on the (still pre-defined) feature vectors $\phi()$. In particular, we can define $f_w()$ using a neural network and train with the standard back-propagation algorithm. We then obtain an optimization scheme, in which we propagate errors and beliefs simultaneously, as shown in detail in the following.

3. Propagating errors and beliefs

For simplicity, we restrict our attention to networks with one hidden layer and linear output units in the following, *ie.* we set:

$$f_{\boldsymbol{w}}(\phi(\boldsymbol{x}, y_{t-1}, y_t)) = \boldsymbol{u}^{\mathrm{T}}\sigma(\boldsymbol{V}^{\mathrm{T}}\phi(\boldsymbol{x}, y_{t-1}, y_t)), \tag{6}$$

where $\boldsymbol{w} := (\boldsymbol{u}, \boldsymbol{V})$, $\boldsymbol{u}$ is a vector of hidden-to-output weights, $\boldsymbol{V}$ is a matrix of feature-to-hidden-layer weights, and $\sigma()$ is a component-wise sigmoid activation function. The crucial observation is, that the gradient of $L(\boldsymbol{w})$ then still decouples as:

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{w}} &= \sum_i \sum_t \frac{\partial f_{\boldsymbol{w}}}{\partial \boldsymbol{w}}(\phi(\boldsymbol{x}^i, y^i_{t-1}, y^i_t)) \\ &\quad - \sum_y p(y|\boldsymbol{x}^i) \sum_t \frac{\partial f_{\boldsymbol{w}}}{\partial \boldsymbol{w}}(\phi(\boldsymbol{x}^i, y_{t-1} y_t)). \end{aligned} \tag{7}$$

Therefore, to obtain the derivatives wrt. the weights $\boldsymbol{u}$ and $\boldsymbol{V}$, we can simply plug-in the form (Eq. 6) and get:

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{u}} &= \sum_i \sum_{\boldsymbol{y}} (\delta_{\boldsymbol{y},\boldsymbol{y}^i} - p(\boldsymbol{y}|\boldsymbol{x}^i)) \times \\ &\quad \sum_t \sigma(\boldsymbol{V}^{\mathrm{T}}\phi(\boldsymbol{x}^i, y_{t-1}, y_t)), \\ \frac{\partial L}{\partial \boldsymbol{V}^{\mathrm{T}}} &= \boldsymbol{u}\mathbf{1}^{\mathrm{T}} \star \sum_i \sum_{\boldsymbol{y}} (\delta_{\boldsymbol{y},\boldsymbol{y}^i} - p(\boldsymbol{y}|\boldsymbol{x}^i)) \\ &\quad \sum_t \sigma^i_{y_{t-1},y_t}(1 - \sigma^i_{y_{t-1},y_t})\phi(y_{t-1}, y_t)\mathbf{1}^{\mathrm{T}}, \end{aligned}$$

where $\mathbf{1}^{\mathrm{T}}$ is the row-vector of all ones, $\star$ denotes element-wise multiplication, and we abbreviate $\sigma^i_{y_{t-1},y_t} := \sigma(\boldsymbol{V}^{\mathrm{T}}\phi(\boldsymbol{x}^i, y_{t-1}, y_t))$. Since the gradient (wrt. any layer) is simply an expectation over the output-distribution, we just need to perform inference in the model to be able to compute it, which opens up a large number of possibilities for approximate inference and nonlinear prediction in intractable models, providing an interesting extension also to structured neural network models (LeCun et al., 1997). The generalization to several hidden layers is straightforward.

4. Discussion

After having lost some of their popularity to kernel machines in the late nineties, neural networks are currently experiencing another renaissance, mainly because of the increasing awareness that large datasets are important in practice. In the context of structure prediction, neural networks could be especially useful, since they implicitly address the problem of *feature extraction*, which has been a problem of ongoing concern in this area (McCallum, 2003).

References

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proc. 18th International Conf. on Machine Learning.*

Lafferty, J., Zhu, X., & Liu, Y. (2004). Kernel conditional random fields: representation and clique selection. *Proc. 21. Intern. Conf. on Machine Learning.*

LeCun, Y., Bottou, L., & Bengio, Y. (1997). Reading checks with multilayer graph transformer networks. *Proceedings of the Intern. Conf. on Acoustics, Speech, and Signal Processing 1997.*

McCallum, A. (2003). Efficiently inducing features of conditional random fields. *Proc. of the 19th Annual Conf. on Uncertainty in AI (UAI-03).*