Examen AMAL: Advanced MAchine Learning & Deep Learning
4/02/2020
Masters DAC et M2A – Sorbonne Université
Durée 2 h, documents de cours autorisés

Course

Half a page describing how to use recurrent neural networks for language modeling and word embeddings.

BP as constrained optimization

Notational conventions for gradients are indicated at the end of the document.

We consider a Multilayer Perceptron, (MLP), we denote $\boldsymbol{x}$ an input vector, $\boldsymbol{y}$ a target vector, $F = F_l \circ \ldots \circ F_1$ the function corresponding to the MLP, $\boldsymbol{z}(i) = F_i \circ \ldots \circ F_1(\boldsymbol{x}), i = 1 \ldots l$ the vector value at layer i, with $\boldsymbol{z}(0) = \boldsymbol{x}$ being the input, $\boldsymbol{w}(i)$ the parameter vector corresponding to F_i. Each F_i applies a linear transformation of its input followed by a sigmoid non linearity.

With these notations $\boldsymbol{z}$ and $\boldsymbol{w}$ are vectors of the appropriate size, e.g. using matrix notations, $\boldsymbol{z}(i)$ is $n_{\boldsymbol{z}}(i) \times 1$ and $\boldsymbol{w}(i)$ is $n_{\boldsymbol{w}}(i) \times 1$. Scalars are in *italics* and vectors in **bold**.

Let us consider the following optimization problem (Pb1)

$Min_{\boldsymbol{w}} c = c(\boldsymbol{z}(l), \boldsymbol{y})$

$$\text{Subject to constraint } \begin{cases} \boldsymbol{z}(l) = F_l(\boldsymbol{z}(l-1), \boldsymbol{w}(l)) \\ \boldsymbol{z}(l-1) = F_{l-1}(\boldsymbol{z}(l-2), \boldsymbol{w}(l-1)) \\ \ldots \\ \boldsymbol{z}(1) = F_1(\boldsymbol{x}, \boldsymbol{w}(1)) \end{cases}$$

Where $c()$ is a differentiable loss function, $\boldsymbol{z}(l)$ is then the computed output of the network and $\boldsymbol{y}$ the target. c is the loss for one datum (x, y) only.

1. Show that the Lagrangian $\mathcal{L}$ associated to this problem writes :

$$\mathcal{L}(x, \boldsymbol{w}) = c(\boldsymbol{z}(l), y) - \sum_{i=1}^{l} \lambda_i^T (\boldsymbol{z}(i) - F_i(z(i-1), \boldsymbol{w}(i)))$$

   Where the λ_i are the vectors of Lagrange coefficients (of size $n_{\boldsymbol{z}}(i) \times 1$).

2. Derive the expressions for the following derivatives and gradients: $\frac{\partial \mathcal{L}}{\partial \lambda_i}, i = 1 \ldots l$; $\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}(l)}$; $\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}(i)}, i = 1 \ldots l-1$; $\frac{\partial \mathcal{L}}{\partial W_i}, i = 1 \ldots l$
3. A necessary condition for a minimum is that $\frac{\partial \mathcal{L}}{\partial \lambda_i} = 0, i = 1 \ldots l$ and $\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}(i)} = 0, i = 1 \ldots l$, what is the interpretation of the first condition $\frac{\partial \mathcal{L}}{\partial \lambda_i} = 0, i = 1 \ldots l$?
4. Let us suppose that this first condition is met, show that one can choose any value for the $\lambda_i s$ in order to solve (Pb1)
5. We will then choose the $\lambda_i s$ such that $\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}(i)} = 0, i = 1 \ldots l-1$. Show that the $\lambda_i s$ can be computed sequentially, starting from λ_l

6. Show that $\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}(i)}$ can then be easily computed.
7. Give an algorithm for training a MLP using the above formalism.
8. Instantiation: we consider a simple classical 2 layer MLP, with a single output, targets $y \in \{0,1\}$ for binary classification, trained according to a cross-entropy criterion. Derive the values for the following expressions: $\lambda_2, \lambda_1, \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}(2)}, \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}(1)}$.

Neural Networks and conditional density mixture

Let us suppose available N independent observations $D = \{(\boldsymbol{x}_i, y_i); i = 1 \dots N\}$ with $\boldsymbol{x} \in R^n$, $y \in R$. Let us denote X the nxN matrix of observations $\boldsymbol{x}_i$ ($\boldsymbol{x}_i$s are the columns of X) and $Y = (y_1, \dots, y_N)^T$. Our objective is to model a multimodal conditional distribution $p(y|\boldsymbol{x})$. This means that to the same value $\boldsymbol{x}$ may correspond several values y or modes. Our objective here is to learn such conditional multimodal distributions. For that, one will use a conditional mixture model:

$$p(y|\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k(\boldsymbol{x}) p_k(y|\boldsymbol{x})$$

The π_k are mixture coefficients and the $p_k(y|\boldsymbol{x})$ are the corresponding mixture components, K is the number of components. Depending on the problem, these components can be chosen according to different distributions. Here we will consider conditional Gaussians for a regression problem.

$$p_k(y|\boldsymbol{x}) = \mathcal{N}(y|\mu_k(\boldsymbol{x}), \sigma_k^2(\boldsymbol{x}))$$

The mean $\mu_k(\boldsymbol{x}) \in R$, the variance $\sigma_k^2(\boldsymbol{x}) \in R^+$ and the proportion $\pi_k(\boldsymbol{x})$ are explicit functions of $\boldsymbol{x}$ and will be computed with a NN.

The $\pi_k(\boldsymbol{x})$ are constrained to verify $\sum_{k=1}^{K} \pi_k(\boldsymbol{x}) = 1$ and $0 \leq \pi_k(\boldsymbol{x}) \leq 1$. This is implemented via a softmax:

$$\pi_k(\boldsymbol{x}) = \frac{\exp(a_k^\pi)}{\sum_{l=1}^{K} \exp(a_l^\pi)}$$

The variances shall remain positive: $\sigma_k(\boldsymbol{x}) = \exp\,(a_k^\sigma)$, the means are denoted $\mu_k(\boldsymbol{x}) = a_k^\mu$.

$a_k^\alpha \in R$ for $\alpha = \pi, \sigma, \mu$.

1. Computing and learning the mixture parameters require computing the a_k^α for $= \pi, \sigma, \mu$. How can the conditional mixture model be implemented via a neural network such as a multilayer perceptron for example?
2. Let us denote $L(\boldsymbol{w}) = \sum_{i=1}^{N} L_i(\boldsymbol{w})$ the log likelihood of this model with $L_i(\boldsymbol{w})$ the log likelihood component for $(\boldsymbol{x}_i, y_i)$ and $\boldsymbol{w}$ the parameters of the neural network. $L_i(\boldsymbol{w}) = \log \sum_{k=1}^{K} \pi_k(\boldsymbol{x}_i) p_k(y_i|\boldsymbol{x}_i)$. For simplification, one denotes $\pi_k(\boldsymbol{x}_i) = \pi_k, p_k(y_i|\boldsymbol{x}_i) = p_{ik}$. In order to compute the derivatives w.r.t. $\boldsymbol{w}$, one needs the derivatives w.r.t. the a_k^α. Derive the expressions for $\frac{\partial L_i(w)}{\partial a_j^\pi}$, $\frac{\partial L_i(w)}{\partial a_j^\mu}$ and $\frac{\partial L_i(w)}{\partial a_j^\sigma}$.
3. Once trained, the network can be used for computing different statistics. Show that the conditional mean $E[y|\boldsymbol{x}]$, and the conditional variance $s^2(\boldsymbol{x}) = E[(|y - E[y|\boldsymbol{x}]|)^2]$ can respectively be written down as:

$$E[y|\boldsymbol{x}] = \sum_{\{k=1\}}^{N} \pi_k(\boldsymbol{x}) \mu_k(\boldsymbol{x}) \qquad \text{and} \qquad s^2(\boldsymbol{x}) = \sum_{\{k=1\}}^{K} \pi_k(\sigma_k^2 + (\mu_k - E[y|\boldsymbol{x}])^2)$$

Useful formulas

Notations: let $\alpha \in R, x \in R^n, y \in R^m$

The usual convention for vector notations and derivatives are the following

Vector: $x = (x_1, \dots, x_n)^T$

Scalar by vector: $$\frac{\partial \alpha}{\partial x} = \left(\frac{\partial \alpha}{\partial x_1}, \cdots, \frac{\partial \alpha}{\partial x_n}\right)$$

Vector by vector: $$\frac{\partial y}{\partial x} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{pmatrix}$$