10-701: Introduction to Machine Learning — Gradient Explanation

Mitchell, Section A
Michael Rosenberg
mmrosenb@andrew.cmu.edu
April 6, 2016

General Structure of The Neural Network

Consider a problem where we have training examples of dimension d and k possible labels on each training example. Say that we have a vocabulary V and for each word $w \in V$, there is a vector $x_w \in \mathbb{R}^d$ associated with that word. This vector is stored in our $d \times |V|$ word embedding matrix W_e. Based on the structure of the parse tree, each phrase p has two constituent phrases p_1 and p_2. We generate the langauge vector for p ($x_p \in \mathbb{R}^d$) Using the function $x_p = \tanh(\underset{d\times d}{W_l}(x_{p_1} + x_{p_2}))$, where W_l is our $d \times d$ language-level matrix. Once we have generated the sentence level vector x_s, we develop a predicted label for it ($\hat{y}$) using the function

$$\hat{y}_s = \text{softmax}(\underset{k\times d}{W_{sm}} x_s),$$

where W_{sm} is our $k \times d$ softmax layer matrix and the softmax function is defined such that $\text{softmax}(q) = \frac{\exp(q)}{\sum_i^k q_i}$.

For generated the predicted label probability vector, $\hat{y}$, we use The function `forwardProp` to predict a label based on a given parse tree. To get the one-hot vector label version of this classification, we use the function `predict`, which is essentially a wrapper for `forwardProp`.

For our baseline loss function, we assumed a Cross-entropy loss between the true labels y and the predicted labels $\hat{y}$:

$$L(y, \hat{y}) = -\sum_{j=1}^{k} y_i \log(\hat{y})_i.$$

Due to the recursive dependence of the network, we train our three weight matrices W_e, W_l, W_{sm} using a gradient descent algorithm known as backpropagation through structure. To explain the algorithm, consider the case where we have one training example.

Training the SoftMax Matrix

Since the Softmax Matrix is only dependent on one layer of the network, the calculation of the derivative of the loss function with respect to W_{sm} is rather straightforward. Consider index i of our labels:

$$\left(\frac{\partial L}{\partial W_{sm}}\right)_i = \frac{\partial L}{\partial o_i}\frac{\partial o_i}{\partial W_{sm}}$$

Where $o_i = ((W_{sm})_i \cdot x_s)$. We see that $\frac{\partial L}{o_i} = \sum_{j=1}^{k} \frac{-y_h}{\hat{y}_j}\frac{\partial \hat{y}_j}{\partial o_i}$. We know that by the construction of softmax,

$$\frac{\partial \hat{y}_j}{\partial o_i} = \hat{y}_j(1-\hat{y}_j), j = i$$

and

$$= -\hat{y}_i\hat{y}_j, j \neq i$$

and thus

$$\begin{aligned}
\frac{\partial L}{o_i} &= -y_i(1-\hat{y}_i) - \sum_{j\neq i} \frac{y_j}{\hat{y}_j}(-\hat{y}_j\hat{y}_i) \\
&= -y_i(1-\hat{y}_i) + \sum_{j\neq i} y_j\hat{y}_i \\
&= -y_i + y_i\hat{y}_i + \hat{y}_i\sum_{j\neq i} y_j \\
&= -y_i + \hat{y}_i\sum_{j=1}^{k} y_j \\
&= \hat{y}_i - y_i.
\end{aligned}$$

Note that

$$\frac{\partial o_i}{\partial W_{sm}} = x_s$$

and thus the ith row of th gradient will be

$$\frac{\partial L}{\partial W_{sm\, i}} = (\hat{y}_i - y_i) \odot x_s,$$

where $\odot$ is the outer product. This leaves the full gradient to be

$$\underset{k\times 1}{(\hat{y}-y)} \odot \underset{d\times 1}{x_s}.$$

This translates into our code in our training algorithm as

```
softmaxMatGradient = ((predictionVec - correctLabel)
                        * givenSentenceTree.langVec.transpose())
```

Which is a simplification given that

$$(\hat{y}-y) \odot x_s = (\hat{y}-y)\cdot(x_s)^T.$$

Language Matrix

Consider our language matrix W_l. We see that

$$\frac{\partial L}{\partial W_l} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial x_s} \frac{\partial x_s}{\partial W_l} = \left((\hat{y} - y)^T \cdot W_{sm}\right)^T \odot \frac{\partial x_s}{\partial W_l}.$$

The question is, how do we handle the generation of $\frac{\partial x_s}{\partial W_l}$? Each phrase in the sentence parse tree is dependent on W_l! Thus, the relation xPy be "there exists a path from x to p" and the logical $Ph(x)$ to be true if x is a phrase and false otherwise. Take $T(x_s) = \{x_p \in \mathbb{R}^d | x_s P x_p \wedge Ph(x_p)\}$. Note that this implies that $x_s \in T(x_s)$. We can see know that

$$\frac{\partial x_s}{W_l} = \sum_{x_p \in T(x_s)} \frac{\partial x_s}{\partial x_p} \frac{\partial x_p}{\partial W_l}.$$

Thus,

$$\frac{\partial L}{\partial W_l} = ((\hat{y} - y)^T \cdot W_{sm})^T \odot \left(\sum_{x_p \in T(x_s)} \frac{\partial x_s}{\partial x_p} \frac{\partial x_p}{\partial W_l} \right).$$

This process of calculating this gradient is stored in the function `buildLanguageGradient`, where we first calculate $(\hat{y} - y) \cdot W_{sm}$ using

```
softmaxLayerDeriv = np.dot((predictedLabel - correctLabel).T,
                           self.softmaxWeightMat)
```

, Then get all of our gradient paths (as in, all paths from x_s to each $x_p \in T(x_s)$) using the function `getLangaugeChainRulePaths`. Then for each chain rule path generated, we calculate $\frac{\partial x_s}{\partial x_p} \frac{\partial x_p}{\partial W_l}$ for the given $x_p \in T(x_s)$ at the end of the path using `languageDerivRecursion(langGradientPath)`.

Word Embedding Matrix

The word embedding matrix follows a similar path, although it is slightly trickier. consider $Wo(x)$ being true is x is a word and false otherwise. let $Word(x_s) = \{x_w \in \mathbb{R}^d | x_s P x_w \wedge Wo(w)\}$ (i.e. $Word(x_s)$ is the set of all words that make up the sentence s). we see that for a given word $v \in V$, if $v \notin Word(x_s)$, we know that $\frac{\partial L}{\partial W_e^v} = 0$, and thus the column of the gradient of W_e that is indexed for word v will be d-dimensional 0 vector if v is not in our sentence. Hence, let us consider $v \in Word(x_s)$. We see that

$$\frac{\partial L}{\partial W_e^v} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial x_s} \frac{\partial x_s}{W_e^v} = ((\hat{y} - y)^T \cdot W_{sm})^T \odot (\frac{\partial x_s}{\partial W_e^v})$$

This is again another problem where it is possible for several parts of the sentence tree that could be dependent on W_e^v since the word v could show up multiple times in a sentence. Hence, consider $D(x_p)$ to be the set of the **direct** children of p (i.e. $D(x_p) \leq 2$ in a parse tree). We can see that

$$\frac{\partial x_s}{\partial W_e^v} = \sum_{x_p \in T(x_s) | x_v \in D(x_p)} \frac{\partial x_s}{\partial x_p} \frac{\partial x_p}{\partial x_v} \frac{\partial x_v}{\partial W_e^v}.$$

(as one can see, $x_v = W_e^v$ so the last partial is somewhat redundant). Hence,

$$\frac{\partial L}{\partial W_e^v} = ((\hat{y} - y)^T \cdot W_{sm})^T \cdot \left(\sum_{x_p \in T(x_s) | x_v \in D(x_p)} \frac{\partial x_s}{\partial x_p} \frac{\partial x_p}{\partial x_v} \frac{\partial x_v}{\partial W_e^v} \right).$$

And thus,

$$\frac{\partial L}{\partial W_e} = \left[\begin{array}{cccc} \frac{\partial L}{\partial W_e^{w1}} & \frac{\partial L}{\partial W_e^{w2}} & \cdots & \frac{\partial L}{\partial W_e^{wV}} \end{array} \right].$$