Part-of-Speech Tagging

$+$

Neural Networks 2

CS 287

Review: Bilinear Model

Bilinear model,

$$\hat{\mathbf{y}} = f((\mathbf{x}^0 \mathbf{W}^0)\mathbf{W}^1 + \mathbf{b})$$

- $\mathbf{x}^0 \in \mathbb{R}^{1 \times d_0}$ start with one-hot.
- $\mathbf{W}^0 \in \mathbb{R}^{d_0 \times d_{\mathrm{in}}}$, $d_0 = |\mathcal{F}|$
- $\mathbf{W}^1 \in \mathbb{R}^{d_{\mathrm{in}} \times d_{\mathrm{out}}}, \mathbf{b} \in \mathbb{R}^{1 \times d_{\mathrm{out}}}$; model parameters

Notes:

- Bilinear parameter interaction.
- $d_0 >> d_{\mathrm{in}}$, e.g. $d_0 = 10000, d_{\mathrm{in}} = 50$

Review: Bilinear Model: Intuition

$$(\mathbf{x}^0\mathbf{W}^0)\mathbf{W}^1 + \mathbf{b}$$

$$\begin{bmatrix} 0 & \ldots & {\color{red}1} & \ldots & 0 \end{bmatrix} \begin{bmatrix} w^0_{1,1} & \ldots & w^0_{0,d_{\text{in}}} \\ & \vdots & \\ & \vdots & \\ & \vdots & \\ {\color{red}w^0_{k,1}} & {\color{red}\ldots} & {\color{red}w^0_{k,d_{\text{in}}}} \\ & \vdots & \\ & \vdots & \\ & \vdots & \\ w^0_{d_0,1} & \ldots & w^0_{d_0,d_{\text{in}}} \end{bmatrix} {\color{red}\begin{bmatrix} w^1_{1,1} & \ldots & \ldots & w^1_{0,d_{\text{out}}} \\ & \ddots & \ddots & \\ w^1_{d_{\text{in}},0} & \ldots & \ldots & w^1_{d_{\text{in}},d_{\text{out}}} \end{bmatrix}}$$

Review: Window Model

Goal: predict t_5.

- Windowed word model.

$$w_1 \;\; w_2 \;\; [w_3 \;\; w_4 \;\; w_5 \;\; w_6 \;\; w_7] \;\; w_8$$

- w_3, w_4; left context
- w_5; Word of interest
- w_6, w_7; right context
- d_{win}; size of window ($d_{\mathrm{win}} = 5$)

Review: Dense Windowed BoW Features

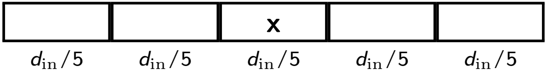
- $f_1, \ldots, f_{d_{\mathrm{win}}}$ are words in window
- Input representation is the concatenation of embeddings

$$\mathbf{x} = [v(f_1)\ v(f_2)\ \ldots\ v(f_{d_{\mathrm{win}}})]$$

Example: Tagging

$$w_1\ w_2\ [w_3\ w_4\ w_5\ w_6\ w_7]\ w_8$$

$$\mathbf{x} = [v(w_3)\ v(w_4)\ v(w_5)\ v(w_6)\ v(w_7)]$$



Rows of $\mathbf{W}^1$ encode position specific weights.

Quiz

We are doing tagging with a windowed bilinear model with hinge-loss and no capitalization features. The model has $d_{\mathrm{win}} = 5$, $d_{\mathrm{in}} = 50$, $d_{\mathrm{out}} = 40$, and vocabulary size 10000.

We are given the input window:

```
The dog walked to the
```

Unfortunately we incorrectly classify `walked` as `NN` as opposed to `VP`, in a bilinear model with a hinge-loss .

What is the maximum number of parameters that receive a non-zero gradient?

Answer:

$$
\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}
\begin{bmatrix}
w^0_{1,1} & \cdots & w^0_{0,d_{\text{in}}} \\
w^0_{the,1} & \cdots & w^0_{the,d_{\text{in}}} \\
\vdots & & \\
w^0_{dog,1} & \cdots & w^0_{dog,d_{\text{in}}} \\
\vdots & & \\
w^0_{walked,1} & \cdots & w^0_{walked,d_{\text{in}}} \\
\vdots & & \\
w^0_{to,1} & \cdots & w^0_{to,d_{\text{in}}} \\
\vdots & & \\
w^0_{the,1} & \cdots & w^0_{the,d_{\text{in}}} \\
\vdots & & \\
w^0_{d_0,1} & \cdots & w^0_{d_0,d_{\text{in}}}
\end{bmatrix}
\begin{bmatrix}
w^1_{1,1} & \cdots & w^1_{1,NN} & \cdots & w^1_{1,VP} & w^1_{0,d_{\text{out}}} \\
 & \ddots & \ddots & & & \\
w^1_{d_{\text{in}},0} & \cdots & w^1_{d_{\text{in}},NN} & \cdots & w^1_{d_{\text{in}},VP} & w^1_{d_{\text{in}},d_{\text{out}}}
\end{bmatrix}
$$

$$\mathbf{W}^0 = 5 \times d_{\text{in}}$$

$$\mathbf{W}^1 = d_{\text{in}} \times 2$$

Part-of-Speech Tagging

Consider the following windowed model, and assume for now a linear model.

$$w_1 \ \texttt{the} \ w_3 \ w_4 \ w_5$$

- What information do we have about the tag of w_3?
- What weight should the features values associated with `the` in position w_2 take?

Part-of-Speech Tagging

Next Consider the following windowed model, and assume for now a linear model.

$$w_1 \ w_2 \ w_3 \ \texttt{dog} \ w_5$$

- What information do we have about the tag of w_3?
- What weight should the features values associated with `dog` in position w_4 take?

Part-of-Speech Tagging

Now finally consider the following windowed model, and assume for now a linear model.

$$w_1 \ \texttt{the} \ w_3 \ \texttt{dog} \ w_5$$

- What information do we have about the tag of w_3?
- What weight would we want if we combined both the features values?

Table

Contents

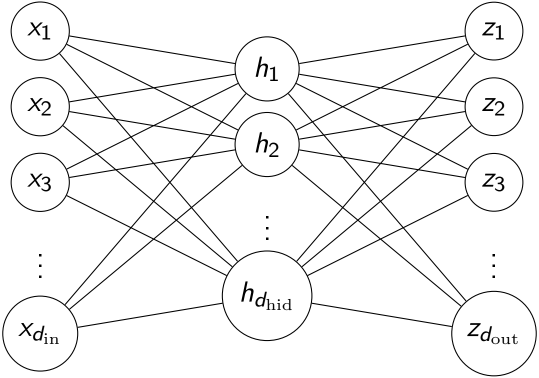
Neural Network

One-layer multi-layer perceptron architecture,

$$NN_{MLP1}(\mathbf{x}) = g(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)W^2 + \mathbf{b}^2$$

- $\mathbf{x}\mathbf{W} + \mathbf{b}$; *perceptron*
- $\mathbf{x}$ is the dense representation in $\mathbb{R}^{1 \times d_{\text{in}}}$
- $\mathbf{W}^1 \in \mathbb{R}^{d_{\text{in}} \times d_{\text{hid}}}$, $\mathbf{b}^1 \in \mathbb{R}^{1 \times d_{\text{hid}}}$; first affine transformation
- $\mathbf{W}^2 \in \mathbb{R}^{d_{\text{hid}} \times d_{\text{out}}}$, $\mathbf{b}^2 \in \mathbb{R}^{1 \times d_{\text{out}}}$; second affine transformation
- $g : \mathbb{R}^{d_{\text{hid}} \times d_{\text{hid}}}$ is an *activation non-linearity* (often pointwise)
- $g(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)$ is the *hidden layer*

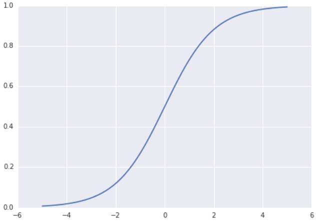
Schematic



Non-Linear Functions

Logistic sigmoid function:

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$



- $\sigma((\mathbf{xW}^1 + \mathbf{b}^1)_i)$
- Intuition: Each hidden dimension ("neuron") is result of logistic regression.
- These probabilities are "features" for next layer.

Feature Conjunctions

Consider the example ...

Non-Convexity

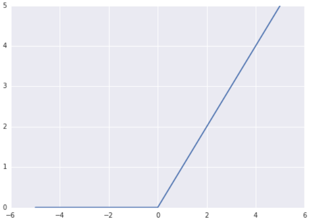
-

Why are these better?

Other Non-Linearities: ReLU

Rectified Linear Unit:

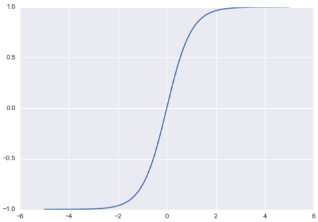
$$\mathrm{ReLU}(t) = \max\{0, t\}$$



Intuition:

Saturation

Saturation: Intuition


1.0
0.5
0.0
−0.5
−1.0
−6
−4
−2
0
2
4
6

Function Approximator

MLP1 is a universal approximator

Deep Neural Networks (DNNs)

Can stack MLPs,

$$NN_{MLP1}(\mathbf{x}) = g(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)W^2 + \mathbf{b}^2$$

$$NN_{MLP2}(\mathbf{x}) = g(NN_{MLP1}(\mathbf{x})\mathbf{W}^1 + \mathbf{b}^1)W^2 + \mathbf{b}^2$$

- Can have multiple hidden layers, etc.

Other types of networks

Highway Network (one example)

$$NN_{MLP2}(\mathbf{x}) = g(NN_{MLP1}(\mathbf{x})\mathbf{W}^1 + \mathbf{b}^1)W^2 + \mathbf{b}^2$$

Deep Neural Networks (DNNs)

Contents

Consider a vector-valued parameterized function $f(\mathbf{x};)$ where

- $f(\mathbf{x}) : \mathbb{R}^m \mapsto \mathbb{R}^n$; function
- $\in \mathbb{R}^d$; function parameters

Consider a scalar-valued loss function $L(\mathbf{x};)$ where

- $L(\mathbf{x}) : \mathbb{R}^n \mapsto \mathbb{R}$; function

Backpropagation

Forward Compute $L(f(\ldots f()))$

Backward

$$\frac{\partial L}{\partial f(\ldots f(x_i))} = \sum_{j=1}^{m} \frac{\partial f(\mathbf{x})_j}{\partial x_i} \frac{\partial L(f(\mathbf{x}))}{\partial f(\mathbf{x})_j}$$

Torch Implementation

$f(\mathbf{x})$ $\frac{\partial L}{\partial f(\mathbf{x})_j}$ $\frac{\partial L}{k}$

Torch Implementation

$f(x)$ $\frac{\partial L}{\partial x}$ $f(x)_j$

$\frac{\partial L}{k}$

Torch Names

- $\mathbf{x}$; *input*
- $f(\mathbf{x})$; *self.output* (saved on forward pass)
- $\frac{\partial L}{x_i}$; *self.gradInput*
- $\frac{\partial L}{f(\mathbf{x})_j}$; *gradOutput*
- ; *gradWeight*
- $\frac{\partial L}{}$; *gradWeight*

Max

Max

Contents