Natural Language Processing

Anoop Sarkar
anoopsarkar.github.io/nlp-class

Simon Fraser University

October 9, 2018

Natural Language Processing

Anoop Sarkar
anoopsarkar.github.io/nlp-class

Simon Fraser University

Part 1: Multi-layer perceptrons

Log linear model

- Let there be m features, $f_k(\mathbf{x}, y)$ for $k = 1, \ldots, m$
- Define a parameter vector $\mathbf{v} \in \mathbb{R}^m$
- A log-linear model for classification into labels $y \in \mathcal{Y}$:

$$\Pr(y \mid \mathbf{x}; \mathbf{v}) = \frac{exp\left(\mathbf{v} \cdot \mathbf{f}(\mathbf{x}, y)\right))}{\sum_{y' \in \mathcal{Y}} exp\left(\mathbf{v} \cdot \mathbf{f}(\mathbf{x}, y')\right))}$$

Advantages

The feature representation $\mathbf{f}(\mathbf{x}, y)$ can represent any aspect of the input that is useful for classification.

Disadvantages

The feature representation $\mathbf{f}(\mathbf{x}, y)$ has to be designed by hand which is time-consuming and error-prone.

Neural Networks

Advantages

- Neural networks replace hand-engineered features with **representation learning**
- Empirical results across many different domains show that learned representations give significant improvements in accuracy
- Neural networks allow end to end training for complex NLP tasks and do not have the limitations of multiple chained pipeline models

Disadvantages

For many tasks linear models are much faster to train compared to neural network models

Alternative Form of Log linear model

Log-linear model:

$$\Pr(y \mid \mathbf{x}; \mathbf{v}) = \frac{exp\left(\mathbf{v} \cdot \mathbf{f}(\mathbf{x}, y)\right))}{\sum_{y' \in \mathcal{Y}} exp\left(\mathbf{v} \cdot \mathbf{f}(\mathbf{x}, y')\right))}$$

Alternative form using functions:

$$\Pr(y \mid x; v) = \frac{exp\left(v(y) \cdot f(x) + \gamma_y\right)}{\sum_{y' \in \mathcal{Y}} exp\left(v(y') \cdot f(x) + \gamma'_y\right))}$$

- Feature vector $f(x)$ maps input x to $\mathbb{R}^d$
- Parameters $v(y) \in \mathbb{R}^d$ and γ_y for each $y \in \mathcal{Y}$
- We use v to refer to the parameter vectors and bias values:

$$v = \{(v(y), \gamma_y) : y \in \mathcal{Y}$$

Representation Learning

Replace hand-engineered features f with learned features ϕ:

$$\Pr(y \mid x; \theta, v) = \frac{exp\left(v(y) \cdot \phi(x; \theta) + \gamma_y\right)}{\sum_{y' \in \mathcal{Y}} exp\left(v(y') \cdot \phi(x; \theta) + \gamma'_y\right))}$$

- Replace $f(x)$ with $\phi(x; \theta) \in \mathbb{R}^d$ where θ are new parameters
- Parameters θ are learned from training data
- Using θ the model ϕ maps input x to $\mathbb{R}^d$: a learned representation of x
- x is assumed to be already represented as a vector of size d
- We will use feedforward neural networks to define $\phi(x; \theta)$
- $\phi(x; \theta)$ will be a **non-linear** mapping to $\mathbb{R}^d$ while f is a **linear** model

A Single Neuron

A single neuron maps input $x \in \mathbb{R}^d$ to output h:

$$h = g(w \cdot x + b)$$

- Weight vector $w \in \mathbb{R}^d$, a bias $b \in \mathbb{R}$ are the parameters of the model learned from training data
- Transfer function $g : \mathbb{R} \to \mathbb{R}$
- It is important that g is a **non-linear** transfer function
- Linear $g(z) = \alpha \cdot z + \beta$ for constants α, β

The ReLU Transfer Function

Rectified Linear Unit (ReLU):

$$g(z) = \{z \text{ if } z \geq 0 \text{ or } 0 \text{ if } z < 0\}$$

or equivalently $g(z) = \max\{0, z\}$

Derivative of ReLU:

$$\frac{dg(z)}{dz} = \{1 \text{ if } z > 0 \text{ or } 0 \text{ if } z < 0\}$$

undefined if $z = 0$

The tanh Transfer Function

tanh transfer function:

$$g(z) = \frac{e^{2z} - 1}{e^{2z} + 1}$$

Derivative of tanh:

$$\frac{dg(z)}{dz} = (1 - g(z))^2$$

undefined if $z = 0$

Derivatives w.r.t. parameters

Derivatives w.r.t. w:

Given

$$h = g(w \cdot x + b)$$

derivatives w.r.t. $w_1, \ldots, w_j, \ldots w_d$:

$$\frac{dh}{dw_j}$$

Derivatives w.r.t. b:

derivatives w.r.t. b:

$$\frac{dh}{db}$$

Chain Rule of Differentiation

Introduce an intermediate variable $z \in \mathbb{R}$

$$z = w \cdot x + b$$

$$h = g(z)$$

Then by the chain rule:

$$\frac{dh}{dw_j} = \frac{dh}{dz}\frac{dz}{dw_j} = \frac{dg(z)}{dz} \times x_j$$

Derivatives w.r.t. b:

derivatives w.r.t. b:

$$\frac{dh}{db}$$

Acknowledgements

Many slides borrowed or inspired from lecture notes by Michael Collins, Chris Dyer, Kevin Knight, Philipp Koehn, Adam Lopez, Graham Neubig and Luke Zettlemoyer from their NLP course materials.

All mistakes are my own.

A big thank you to all the students who read through these notes and helped me improve them.