CS 6501 Natural Language Processing

Feed-forward Neural Networks

Yangfeng Ji

October 3, 2018

Department of Computer Science
University of Virginia

UNIVERSITY of VIRGINIA | ENGINEERING

Overview

Introduction

Classification

Decision function

$$\Psi(\boldsymbol{x}, y) = \boldsymbol{w}_y^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}) \tag{1}$$

- $\boldsymbol{x}$: data point
- y: label
- $\boldsymbol{w}_y$: classification weights with respect to label y
- $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})$: feature function
- $\boldsymbol{\theta}$: parameter of feature function

Example: Feature engineering

How to construct $f(x, \theta)$?

Example sentence

I love drinking coffee

- Unigram: `I`, `love`, `drinking`, `coffee`
- Bigram: `I love`, `love drinking`, ...
- POS tags: $\langle$ I, IN$\rangle$, ...
- Production rules: S $\rightarrow$ NP VP, ...
- ...

Example: Feature engineering

How to construct $f(\boldsymbol{x}, \boldsymbol{\theta})$?

Example sentence

I love drinking coffee

Vocab	I	love	drinking	hate	coffee	tea
$\boldsymbol{x}^\top$	[1	1	1	0	1	0]

$$
\begin{aligned}
f(\boldsymbol{x}, \boldsymbol{\theta}) &= \mathbf{V}\boldsymbol{x} \\
\Psi(\boldsymbol{x}, y) &= \boldsymbol{w}_y^\top(\mathbf{V}\boldsymbol{x})
\end{aligned}
$$

where $\boldsymbol{\theta} = \mathbf{V}$

An Alternative View

Vocab	`I`	`love`	`drinking`	`hate`	`coffee`	`tea`
$\boldsymbol{x}^\top$	[1	1	1	0	1	0]
$\mathbf{V}$	$[v_{\text{I}}$	v_{love}	v_{drinking}	v_{hate}	v_{coffee}	$v_{\text{tea}}]$

An Alternative View

$$\begin{array}{c|cccccc} \text{Vocab} & \texttt{I} & \texttt{love} & \texttt{drinking} & \texttt{hate} & \texttt{coffee} & \texttt{tea} \\ \boldsymbol{x}^\top & [1 & 1 & 1 & 0 & 1 & 0] \\ \mathbf{V} & [v_{\text{I}} & v_{\text{love}} & v_{\text{drinking}} & v_{\text{hate}} & v_{\text{coffee}} & v_{\text{tea}}] \end{array}$$

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = v_{\text{I}} + v_{\text{love}} + v_{\text{drinking}} + v_{\text{coffee}} \tag{2}$$

Linear Functions

Looking for a more powerful model then $f(\boldsymbol{x}, \boldsymbol{\theta}) = \mathbf{V}\boldsymbol{x}$?

How about

$$f(\boldsymbol{x}, \boldsymbol{\theta}) \quad = \quad \mathbf{U}\mathbf{V}\boldsymbol{x}$$

Linear Functions

Looking for a more powerful model then $f(\boldsymbol{x}, \boldsymbol{\theta}) = \mathbf{V}\boldsymbol{x}$?

How about

$$\begin{aligned} f(\boldsymbol{x}, \boldsymbol{\theta}) &= \mathbf{U}\mathbf{V}\boldsymbol{x} \\ &= (\mathbf{U}\mathbf{V})\boldsymbol{x} \end{aligned}$$

Not really, maybe a little. Essentially, it is still a linear function with a single matrix decomposed as $\mathbf{UV}$.

Nonlinearity

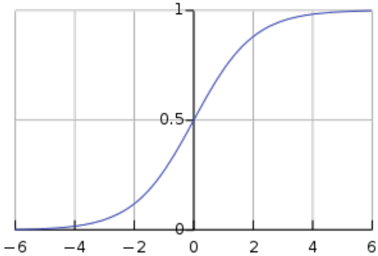
Add a nonlinear function $\boldsymbol{h}$

$$\begin{aligned} f(\boldsymbol{x}, \boldsymbol{\theta}) &= \boldsymbol{h}(\mathbf{V}\boldsymbol{x}) \\ \Psi(\boldsymbol{x}, y) &= \boldsymbol{w}_y^\top \boldsymbol{h}(\mathbf{V}\boldsymbol{x}) \end{aligned}$$

Now, it is a neural network!

Example: Sigmoid function

$$\boldsymbol{h}(\boldsymbol{t}) = \frac{1}{1 - e^{-\boldsymbol{t}}}$$

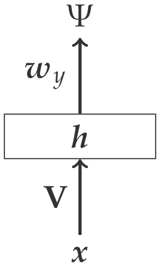


Feed-forward Neural Networks

A Simple Feed-forward Network

A fully-connected feed-forward neural network

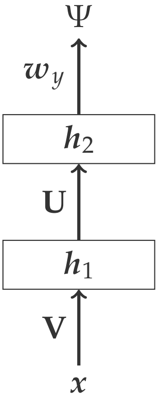
$$\Psi(\boldsymbol{x}, y) = \boldsymbol{w}_y^\top \boldsymbol{h}(\mathbf{V}\boldsymbol{x}) \tag{3}$$



Another Feed-forward Network

$$\Psi(x, y) = w_y^\top \cdot \underbrace{h_2(\mathbf{U} \cdot h_1(\mathbf{V} \cdot x))}_{f(x,\theta)} \qquad (4)$$

where h_1 and h_2 are nonlinear functions without parameters (hidden units).

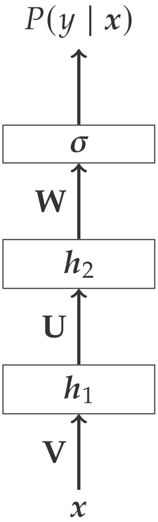


Softmax Function

Normalize the score function to make a probability

$$P(y \mid \boldsymbol{x}) = \boldsymbol{\sigma}(\Psi(\boldsymbol{x}, y)) = \frac{\exp(\Psi(\boldsymbol{x}, y))}{\sum_{y'} \exp(\Psi(\boldsymbol{x}, y'))} \tag{5}$$

- Main advantage is on training
- This is not a probabilistic model



Loss Function: Cross entropy

Binary classification on a single data point $\boldsymbol{x}$ with $y \in \{0, 1\}$

$$\ell = -y \log P(y = 1 \mid \boldsymbol{x}) - (1 - y) \log(1 - P(y = 1 \mid \boldsymbol{x})) \quad (6)$$

Loss Function: Cross entropy

Binary classification on a single data point $\boldsymbol{x}$ with $y \in \{0, 1\}$

$$\ell = -y \log P(y = 1 \mid \boldsymbol{x}) - (1 - y) \log(1 - P(y = 1 \mid \boldsymbol{x})) \quad (6)$$

- if $y = 1$:

$$\ell = -\log P(y = 1 \mid \boldsymbol{x})$$

- if $y = 0$:

$$\ell = -\log(1 - P(y = 1 \mid \boldsymbol{x})) = -\log P(y = 0 \mid \boldsymbol{x})$$

Loss Function: Cross entropy

K-class: convert label to K-dimensional one-hot vector with $\boldsymbol{y}_k = 1$, if k is the label

$$\begin{aligned} \ell &= -\sum_{k=1}^{K} \boldsymbol{y}_k \log P(\boldsymbol{y}_k = 1 \mid \boldsymbol{x}) \\ &= -\log P(\boldsymbol{y}_k = 1 \mid \boldsymbol{x}) \end{aligned} \tag{7}$$

Loss Function: Cross entropy

K-class: convert label to K-dimensional one-hot vector with $\boldsymbol{y}_k = 1$, if k is the label

$$\begin{aligned} \ell &= -\sum_{k=1}^{K} \boldsymbol{y}_k \log P(\boldsymbol{y}_k = 1 \mid \boldsymbol{x}) \\ &= -\log P(\boldsymbol{y}_k = 1 \mid \boldsymbol{x}) \end{aligned} \tag{7}$$

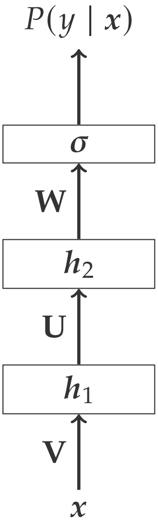
Essentially, it is the same as negative log-likelihood (NLL) in logistic regression.

Back Propagation

Online Learning

Training NNs with one example at a time (redefine $\boldsymbol{\theta}$ as $\{\mathbf{W}, \mathbf{U}, \mathbf{V}\}$)

$$\ell(\boldsymbol{\theta}) = -\log P(\boldsymbol{y}_k = 1 \mid \boldsymbol{x}) \tag{8}$$



Gradient based Learning

Stochastic gradient descent

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \cdot \frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{9}$$

For a subset of $\boldsymbol{\theta}$, e.g., $\boldsymbol{w}_k$

$$\boldsymbol{w}_k \leftarrow \boldsymbol{w}_k - \eta \cdot \frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{w}_k} \tag{10}$$

Gradient based Learning (cont.)

Recall the defition of $P(\boldsymbol{y} \mid \boldsymbol{x})$

$$\begin{aligned}\log P(\boldsymbol{y}_k \mid \boldsymbol{x}) =& \boldsymbol{w}_k^\top \cdot \boldsymbol{h}_2(\mathbf{U} \cdot \boldsymbol{h}_1(\mathbf{V} \cdot \boldsymbol{x})) \\ & - \log \sum_{k'} \exp(\boldsymbol{w}_{k'}^\top \cdot \boldsymbol{h}_2(\mathbf{U} \cdot \boldsymbol{h}_1(\mathbf{V} \cdot \boldsymbol{x})))\end{aligned} \tag{11}$$

Gradient based Learning (cont.)

Recall the defition of $P(\boldsymbol{y} \mid \boldsymbol{x})$

$$\begin{aligned}\log P(\boldsymbol{y}_k \mid \boldsymbol{x}) =& \boldsymbol{w}_k^\top \cdot \boldsymbol{h}_2(\mathbf{U} \cdot \boldsymbol{h}_1(\mathbf{V} \cdot \boldsymbol{x})) \\ & - \log \sum_{k'} \exp(\boldsymbol{w}_{k'}^\top \cdot \boldsymbol{h}_2(\mathbf{U} \cdot \boldsymbol{h}_1(\mathbf{V} \cdot \boldsymbol{x})))\end{aligned} \tag{11}$$

Gradient wrt $\boldsymbol{w}_k$

$$\begin{aligned}\frac{\partial \ell}{\partial \boldsymbol{w}_k} &= -\frac{\partial}{\partial \boldsymbol{w}_k} \log P(\boldsymbol{y} \mid \boldsymbol{x}) \\ &= -\boldsymbol{h}_2(\mathbf{U} \cdot \boldsymbol{h}_1(\mathbf{V} \cdot \boldsymbol{x}))) \\ &\quad + P(\boldsymbol{y}_k \mid \boldsymbol{x}) \cdot \boldsymbol{h}_2(\mathbf{U} \cdot \boldsymbol{h}_1(\mathbf{V} \cdot \boldsymbol{x}))\end{aligned} \tag{12}$$

Basic Derivatives

- $\frac{d(a \cdot z)}{dz} = a$
- $\frac{d \log(z)}{dz} = \frac{1}{z}$
- $\frac{de^z}{dz} = e^z$

Basic Derivatives

- $\frac{d(a \cdot z)}{dz} = a$
- $\frac{d \log(z)}{dz} = \frac{1}{z}$
- $\frac{de^z}{dz} = e^z$
- Chain rule: $\frac{df(g(z))}{dz} = \frac{df(g(z))}{dg(z)} \cdot \frac{dg(z)}{dz}$

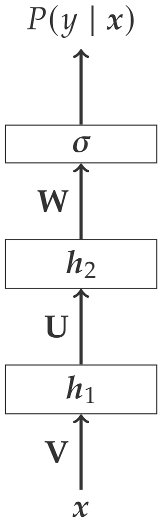
One More Example

Given

$$\begin{aligned} \ell &= -\log P(y \mid \boldsymbol{x}) \\ &= -\log \sigma(\mathbf{W} \cdot \boldsymbol{h}_2(\mathbf{U} \cdot \boldsymbol{h}_1(\mathbf{V} \cdot \boldsymbol{x}))) \end{aligned}$$

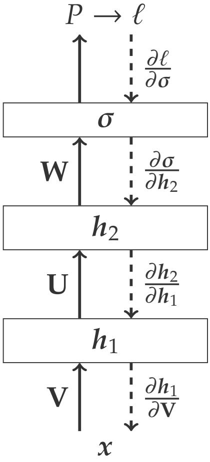
with the chain rule, we have

$$\frac{\partial \ell}{\partial \mathbf{V}} = \frac{\partial \ell}{\partial \boldsymbol{\sigma}} \cdot \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{h}_2} \cdot \frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{h}_1} \cdot \frac{\partial \boldsymbol{h}_1}{\partial \mathbf{V}}$$



Back Propagation

$$\frac{\partial \ell}{\partial \mathbf{V}} = \frac{\partial \ell}{\partial \boldsymbol{\sigma}} \cdot \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{h}_2} \cdot \frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{h}_1} \cdot \frac{\partial \boldsymbol{h}_1}{\partial \mathbf{V}} \tag{13}$$

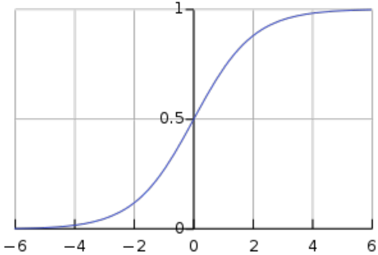


Problems of Gradients

$$\frac{\partial \ell}{\partial \mathbf{V}} = \frac{\partial \ell}{\partial \boldsymbol{\sigma}} \cdot \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{h}_2} \cdot \frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{h}_1} \cdot \frac{\partial \boldsymbol{h}_1}{\partial \mathbf{V}} \tag{14}$$

Vanishing gradients, if $\|\frac{\partial \cdot}{\partial \cdot}\| \ll 1$

$$\|\frac{\partial \ell}{\partial \mathbf{V}}\| \to 0 \tag{15}$$



Solution: initialize the parameters carefully

Problems of Gradients (cont.)

Exploding gradients, if $\|\frac{\partial \cdot}{\partial \cdot}\| > 1$

$$\|\frac{\partial \ell}{\partial \mathbf{V}}\| > M \tag{16}$$

Problems of Gradients (cont.)

Exploding gradients, if $\|\frac{\partial \cdot}{\partial \cdot}\| > 1$

$$\|\frac{\partial \ell}{\partial \mathbf{V}}\| > M \tag{16}$$

Solution: norm clipping [Pascanu et al., 2013]

$$\tilde{\boldsymbol{g}} \leftarrow \lambda \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|} \tag{17}$$

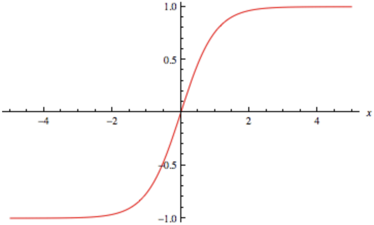
where $\boldsymbol{g} = \frac{\partial \ell}{\partial \mathbf{V}}$ and $1 < \lambda \leq 5$.

Further Comments

Choices of Hidden Units

Tanh function

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$



Sigmoids

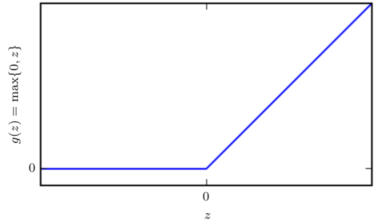
1. Symmetric sigmoids such as hyperbolic tangent often converge faster than the standard logistic function.
2. A recommended sigmoid [19] is: $f(x) = 1.7159 \tanh\left(\frac{2}{3}x\right)$. Since the tanh function is sometimes computationally expensive, an approximation of it by a ratio of polynomials can be used instead.
3. Sometimes it is helpful to add a small linear term, e.g. $f(x) = \tanh(x) + ax$ so as to avoid flat spots.

[LeCun et al., 2012]

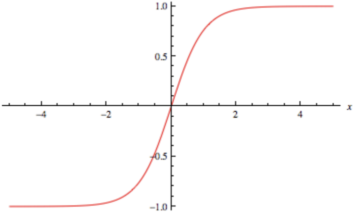
Choices of Hidden Units (cont.)

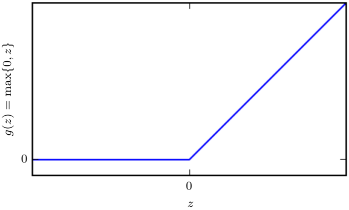
Rectified linear unit

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



Comparison





- Gradient
- Input range

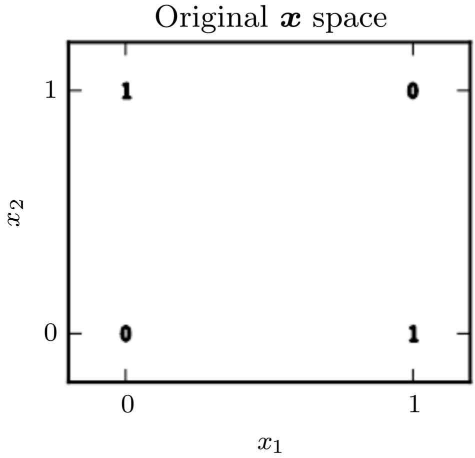
Mini-batch

Mini-batch size K

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \cdot \frac{1}{K} \sum_{k=1}^{K} \frac{\partial \ell_{k}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{18}$$

- Typically $10 \leq K \leq 100$ [Hinton, 2012]
- Larger K
  - gives more reliable estiamte of gradient
  - takes advantage of matrix-vector muptiplies on GPU
- Should work together with learning late η

XOR Problem



Summary

1. Introduction
2. Feed-forward Neural Networks
3. Back Propagation
4. Further Comments

Reference

Hinton, G. E. (2012).
A practical guide to training restricted boltzmann machines.
In *Neural networks: Tricks of the trade*, pages 599–619. Springer.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012).
Efficient backprop.
In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013).
On the difficulty of training recurrent neural networks.
In *International Conference on Machine Learning*, pages 1310–1318.