CS 6501 Natural Language Processing

Text Classification (II): Neural Classifiers

Yangfeng Ji

Information and Language Processing Lab
Department of Computer Science
University of Virginia

Overview

From Logistic Regression to Neural Networks

Logistic Regression

A quick review of logistic regression with $\boldsymbol{x} = [x_1, \ldots, x_d] \in \mathbb{R}^d$

- An basic form for binary classification $y \in \{-1, +1\}$

$$P(Y = +1 \mid \boldsymbol{x}) = \frac{1}{1 + \exp(-\langle \boldsymbol{w}, \boldsymbol{x} \rangle)} \tag{1}$$

Logistic Regression

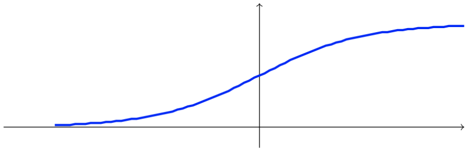
A quick review of logistic regression with $\boldsymbol{x} = [x_1, \ldots, x_d] \in \mathbb{R}^d$

- An basic form for binary classification $y \in \{-1, +1\}$

$$P(Y = +1 \mid \boldsymbol{x}) = \frac{1}{1 + \exp(-\langle \boldsymbol{w}, \boldsymbol{x} \rangle)} \tag{1}$$

- The sigmoid function $\sigma(a)$ with $a \in \mathbb{R}$, which is a monotonic nonlinear function

$$\sigma(a) = \frac{1}{1 + \exp(-a)} \tag{2}$$



Graphical Representation

- A specific example of LR with four input features

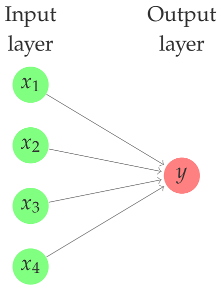
$$P(Y = 1 \mid \boldsymbol{x}) = \sigma(\sum_{j=1}^{4} w_j x_j) \tag{3}$$

Graphical Representation

- A specific example of LR with four input features

$$P(Y = 1 \mid \boldsymbol{x}) = \sigma(\sum_{j=1}^{4} w_j x_j) \tag{3}$$

- The graphical representation of this LR model is



From LR to a Simple Neural Network

Build upon logistic regression, a simple neural network with K hidden units $\{z_k\}_{k=1}^{K}$ can be constructed as

$$z_k = \sigma(\sum_{j=1}^{d} w_{k,j}^{(1)} x_j) \quad k \in \{1, \dots, K\} \tag{4}$$

(5)

- $\boldsymbol{x} \in \mathbb{R}^d$: d-dimensional input
- K is the number of hidden units, each of them has the same form as a LR.

From LR to a Simple Neural Network

Build upon logistic regression, a simple neural network with K hidden units $\{z_k\}_{k=1}^{K}$ can be constructed as

$$\begin{aligned} z_k &= \sigma(\sum_{j=1}^{d} w_{k,j}^{(1)} x_j) \quad k \in \{1, \ldots, K\} && (4) \\ P(y = +1 \mid \boldsymbol{x}) &= \sigma(\sum_{k=1}^{K} w_k^{(o)} z_k) && (5) \end{aligned}$$

- $\boldsymbol{x} \in \mathbb{R}^d$: d-dimensional input
- K is the number of hidden units, each of them has the same form as a LR.
- $y \in \{-1, +1\}$ (binary classification problem)

From LR to a Simple Neural Network

Build upon logistic regression, a simple neural network with K hidden units $\{z_k\}_{k=1}^{K}$ can be constructed as

$$\begin{aligned} z_k &= \sigma(\sum_{j=1}^{d} w_{k,j}^{(1)} x_j) \quad k \in \{1, \ldots, K\} && (4) \\ P(y = +1 \mid \boldsymbol{x}) &= \sigma(\sum_{k=1}^{K} w_k^{(o)} z_k) && (5) \end{aligned}$$

- $\boldsymbol{x} \in \mathbb{R}^d$: d-dimensional input
- K is the number of hidden units, each of them has the same form as a LR.
- $y \in \{-1, +1\}$ (binary classification problem)
- $\{w_{k,i}^{(1)}\}$ and $\{w_k^{(o)}\}$ are two sets of the parameters, and
- $\sigma(\cdot)$ is called the activation function

Mathematical Formulation

With the notations of matrix-vector multiplication, we can rewrite these equations into a more concise form

- Element-wise formulation

$$z_k = \sigma\left(\sum_{j=1}^{d} w_{k,j}^{(1)} x_j\right) \quad k \in [K] \tag{6}$$

$$P(y = +1 \mid \boldsymbol{x}) = \sigma\left(\sum_{k=1}^{K} w_k^{(o)} z_k\right) \tag{7}$$

Mathematical Formulation

With the notations of matrix-vector multiplication, we can rewrite these equations into a more concise form

- Element-wise formulation

$$z_k = \sigma(\sum_{j=1}^{d} w_{k,j}^{(1)} x_j) \quad k \in [K] \tag{6}$$

$$P(y = +1 \mid \boldsymbol{x}) = \sigma(\sum_{k=1}^{K} w_k^{(o)} z_k) \tag{7}$$

- Matrix-vector formulation

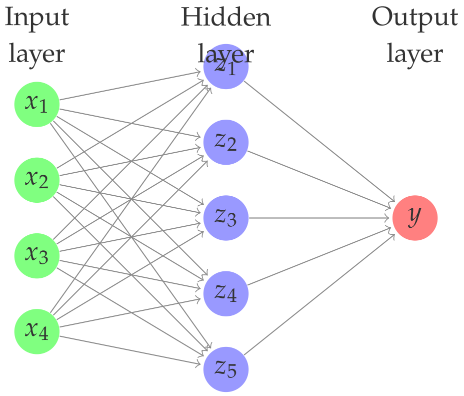
$$\boldsymbol{z} = \sigma(\mathbf{W}^{(1)} \boldsymbol{x}) \tag{8}$$

$$P(y = +1 \mid \boldsymbol{x}) = \sigma((\boldsymbol{w}^{(o)})^{\mathsf{T}} \boldsymbol{z}) \tag{9}$$

where $\mathbf{W}^{(1)} \in \mathbb{R}^{K \times d}$ and $\mathbf{w}^{(o)} \in \mathbb{R}^{K}$

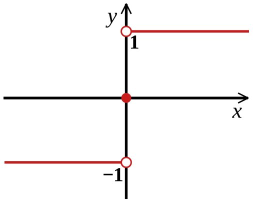
Graphical Representation

Assume the input dimension $d = 4$ and the number of hidden units (hidden dimension) $K = 5$, we can represent this neural network model with a similar graphical representation



Other Activation Functions

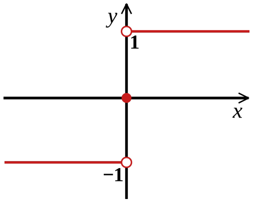
In deep learning, we have a few options about activation functions:



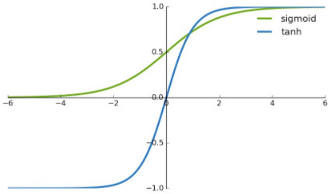
(a) Sign function

Other Activation Functions

In deep learning, we have a few options about activation functions:



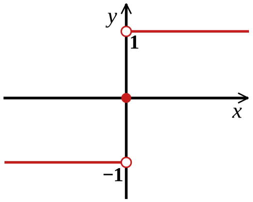
(a) Sign function



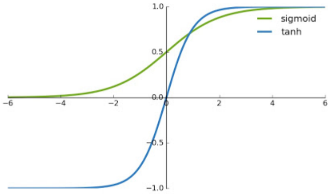
(b) Sigmoid and Tanh function

Other Activation Functions

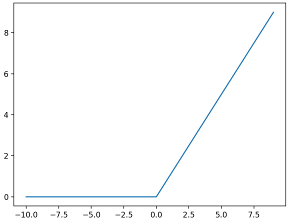
In deep learning, we have a few options about activation functions:



(a) Sign function



(b) Sigmoid and Tanh function



(c) ReLU function [Jarrett et al., 2009]

Learning Neural Networks

Neural Network Predictions

Consider the previously defined neural network model for binary classification $\mathcal{Y} = \{-1, +1\}$,

- Re-write the model definition into a compact form

$$P(Y = +1 \mid \boldsymbol{x}) = \sigma\left((\boldsymbol{w}^{(o)})^{\top}\sigma(\mathbf{W}^{(1)}\boldsymbol{x})\right) \tag{10}$$

  where $\{\boldsymbol{w}^{(o)}, \mathbf{W}^{(1)}\}$ are the parameters

Neural Network Predictions

Consider the previously defined neural network model for binary classification $\mathcal{Y} = \{-1, +1\}$,

- Re-write the model definition into a compact form

$$P(Y = +1 \mid \boldsymbol{x}) = \sigma\left((\boldsymbol{w}^{(o)})^{\top}\sigma(\mathbf{W}^{(1)}\boldsymbol{x})\right) \tag{10}$$

  where $\{\boldsymbol{w}^{(o)}, \mathbf{W}^{(1)}\}$ are the parameters

- Assume the ground-truth label is y, let's introduce an empirical distribution

$$Q(Y = y' \mid \boldsymbol{x}) = \delta(y', y) = \begin{cases} 1 & y' = y \\ 0 & y' \neq y \end{cases} \tag{11}$$

Cross Entropy

Given one data point, The loss function of a neural network is usually defined as the cross entropy of the prediction distribution p and the empirical distribution p

$$\begin{aligned} H(Q,P) &= -Q(Y=+1 \mid \boldsymbol{x}) \log P(Y=+1 \mid \boldsymbol{x}) \\ &\quad -Q(Y=-1 \mid \boldsymbol{x}) \log P(Y=-1 \mid \boldsymbol{x}) \end{aligned} \tag{12}$$

Cross Entropy

Given one data point, The loss function of a neural network is usually defined as the cross entropy of the prediction distribution p and the empirical distribution p

$$\begin{aligned} H(Q, P) &= -Q(Y = +1 \mid \boldsymbol{x}) \log P(Y = +1 \mid \boldsymbol{x}) \\ &\quad -Q(Y = -1 \mid \boldsymbol{x}) \log P(Y = -1 \mid \boldsymbol{x}) \end{aligned} \tag{12}$$

Since q is defined with a Delta function, depending on y, we have

$$H(Q, P) = \begin{cases} -\log P(Y = +1 \mid \boldsymbol{x}) & Y = +1 \\ -\log P(Y = -1 \mid \boldsymbol{x}) & Y = -1 \end{cases} \tag{13}$$

Learning

- Given a set of training example $S = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$, the loss function is defined as

$$L(\boldsymbol{\theta}) = -\sum_{i=1}^{m} \log p(y^{(i)} \mid \boldsymbol{x}^{(i)}) \tag{14}$$

  where $\boldsymbol{\theta}$ indicates all the parameters in a network.

Learning

- Given a set of training example $S = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$, the loss function is defined as

$$L(\boldsymbol{\theta}) = -\sum_{i=1}^{m} \log p(y^{(i)} \mid \boldsymbol{x}^{(i)}) \tag{14}$$

  where $\boldsymbol{\theta}$ indicates all the parameters in a network.

- For example, $\boldsymbol{\theta} = \{\boldsymbol{w}^{(o)}, \mathbf{W}^{(1)}\}$, for the previously defined two-layer neural network

Learning

- Given a set of training example $S = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$, the loss function is defined as

$$L(\boldsymbol{\theta}) = -\sum_{i=1}^{m} \log p(y^{(i)} \mid \boldsymbol{x}^{(i)}) \tag{14}$$

  where $\boldsymbol{\theta}$ indicates all the parameters in a network.
- For example, $\boldsymbol{\theta} = \{\boldsymbol{w}^{(o)}, \mathbf{W}^{(1)}\}$, for the previously defined two-layer neural network
- Just like learning a LR, we can use the gradient-based learning

Gradient-based Learning

A simple description of gradient-based learning

1. Compute the gradient of $\boldsymbol{\theta}$, $\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$

Gradient-based Learning

A simple description of gradient-based learning

1. Compute the gradient of $\boldsymbol{\theta}$, $\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$
2. Update the parameter with the gradient

$$\boldsymbol{\theta}^{(\text{new})} \leftarrow \boldsymbol{\theta}^{(\text{old})} - \eta \cdot \frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(\text{old})}} \tag{15}$$

where η is the learning rate

Gradient-based Learning

A simple description of gradient-based learning

1. Compute the gradient of $\boldsymbol{\theta}$, $\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$
2. Update the parameter with the gradient

$$\boldsymbol{\theta}^{(\text{new})} \leftarrow \boldsymbol{\theta}^{(\text{old})} - \eta \cdot \frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(\text{old})}} \tag{15}$$

   where η is the learning rate

3. Go back step 1 until it converges

Gradient Computation

Consider the two-layer neural network with one training example $(\boldsymbol{x}, y)$, to further simplify the computation, we assume $y = +1$

$$\log p(y \mid \boldsymbol{x}) = \log \sigma \left((\boldsymbol{w}^{(o)})^\top \sigma(\mathbf{W}^{(1)} \boldsymbol{x}) \right) \tag{16}$$

Gradient Computation

Consider the two-layer neural network with one training example $(\boldsymbol{x}, y)$, to further simplify the computation, we assume $y = +1$

$$\log p(y \mid \boldsymbol{x}) = \log \sigma \left((\boldsymbol{w}^{(o)})^\top \sigma(\mathbf{W}^{(1)} \boldsymbol{x}) \right) \tag{16}$$

The gradient with respect to $\boldsymbol{w}^{(o)}$ is

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{w}^{(o)}} = -\frac{\partial \log \sigma\left(\cdot \right)}{\partial \sigma\left(\cdot \right)} \tag{17}$$

Gradient Computation

Consider the two-layer neural network with one training example $(\boldsymbol{x}, y)$, to further simplify the computation, we assume $y = +1$

$$\log p(y \mid \boldsymbol{x}) = \log \sigma \left((\boldsymbol{w}^{(o)})^{\mathsf{T}} \sigma(\mathbf{W}^{(1)} \boldsymbol{x}) \right) \tag{16}$$

The gradient with respect to $\boldsymbol{w}^{(o)}$ is

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{w}^{(o)}} = -\frac{\partial \log \sigma\left(\cdot \right)}{\partial \sigma\left(\cdot \right)} \cdot \frac{\partial \sigma\left((\boldsymbol{w}^{(o)})^{\mathsf{T}} \sigma(\mathbf{W}^{(1)} \boldsymbol{x}) \right)}{\partial (\boldsymbol{w}^{(o)})^{\mathsf{T}} \sigma(\mathbf{W}^{(1)} \boldsymbol{x})} \tag{17}$$

Gradient Computation

Consider the two-layer neural network with one training example $(\boldsymbol{x}, y)$, to further simplify the computation, we assume $y = +1$

$$\log p(y \mid \boldsymbol{x}) = \log \sigma\left((\boldsymbol{w}^{(o)})^{\mathsf{T}} \sigma(\mathbf{W}^{(1)}\boldsymbol{x})\right) \tag{16}$$

The gradient with respect to $\boldsymbol{w}^{(o)}$ is

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{w}^{(o)}} = -\frac{\partial \log \sigma\left(\cdot\right)}{\partial \sigma\left(\cdot\right)} \cdot \frac{\partial \sigma\left((\boldsymbol{w}^{(o)})^{\mathsf{T}} \sigma(\mathbf{W}^{(1)}\boldsymbol{x})\right)}{\partial (\boldsymbol{w}^{(o)})^{\mathsf{T}} \sigma(\mathbf{W}^{(1)}\boldsymbol{x})} \cdot \frac{\partial (\boldsymbol{w}^{(o)})^{\mathsf{T}} \sigma(\mathbf{W}^{(1)}\boldsymbol{x})}{\partial \boldsymbol{w}^{(o)}} \tag{17}$$

Gradient Computation

Consider the two-layer neural network with one training example $(\boldsymbol{x}, y)$, to further simplify the computation, we assume $y = +1$

$$\log p(y \mid \boldsymbol{x}) = \log \sigma \left((\boldsymbol{w}^{(o)})^\top \sigma(\mathbf{W}^{(1)} \boldsymbol{x}) \right) \tag{16}$$

The gradient with respect to $\boldsymbol{w}^{(o)}$ is

$$\begin{aligned} \frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{w}^{(o)}} &= -\frac{\partial \log \sigma \left(\cdot \right)}{\partial \sigma \left(\cdot \right)} \cdot \frac{\partial \sigma \left((\boldsymbol{w}^{(o)})^\top \sigma(\mathbf{W}^{(1)} \boldsymbol{x}) \right)}{\partial (\boldsymbol{w}^{(o)})^\top \sigma(\mathbf{W}^{(1)} \boldsymbol{x})} \cdot \frac{\partial (\boldsymbol{w}^{(o)})^\top \sigma(\mathbf{W}^{(1)} \boldsymbol{x})}{\partial \boldsymbol{w}^{(o)}} \\ &= -\left\{ 1 - \sigma \left((\boldsymbol{w}^{(o)})^\top \sigma(\mathbf{W}^{(1)} \boldsymbol{x}) \right) \right\} \cdot \sigma(\mathbf{W}^{(1)} \boldsymbol{x}) \end{aligned} \tag{17}$$

Gradient Computation (II)

The gradient with respect to $W^{(1)}$ is

$$\begin{aligned}\frac{\partial L(\boldsymbol{\theta})}{\partial \mathbf{W}^{(1)}} &= -\frac{\partial \log \sigma\Big(\cdot\Big)}{\partial \sigma\Big(\cdot\Big)} \cdot \frac{\partial \sigma\Big((\boldsymbol{w}^{(o)})^{\mathsf{T}} \sigma(\mathbf{W}^{(1)}\boldsymbol{x})\Big)}{\partial (\boldsymbol{w}^{(o)})^{\mathsf{T}} \sigma(\mathbf{W}^{(1)}\boldsymbol{x})} \\ &\quad \cdot \frac{\partial (\boldsymbol{w}^{(o)})^{\mathsf{T}} \sigma(\mathbf{W}^{(1)}\boldsymbol{x})}{\partial \sigma(\mathbf{W}^{(1)}\boldsymbol{x})} \cdot \frac{\partial \sigma(\mathbf{W}^{(1)}\boldsymbol{x})}{\partial \mathbf{W}^{(1)}\boldsymbol{x}} \cdot \frac{\partial \mathbf{W}^{(1)}\boldsymbol{x}}{\partial \mathbf{W}^{(1)}}\end{aligned} \tag{18}$$

Gradient Computation (II)

The gradient with respect to $W^{(1)}$ is

$$\begin{aligned}\frac{\partial L(\boldsymbol{\theta})}{\partial \mathbf{W}^{(1)}} \quad = \quad & -\frac{\partial \log \sigma\Big(\,\cdot\,\Big)}{\partial \sigma\Big(\,\cdot\,\Big)} \cdot \frac{\partial \sigma\Big((\boldsymbol{w}^{(o)})^{\mathsf{T}} \sigma(\mathbf{W}^{(1)}\boldsymbol{x})\Big)}{\partial (\boldsymbol{w}^{(o)})^{\mathsf{T}} \sigma(\mathbf{W}^{(1)}\boldsymbol{x})} \\ & \cdot \frac{\partial (\boldsymbol{w}^{(o)})^{\mathsf{T}} \sigma(\mathbf{W}^{(1)}\boldsymbol{x})}{\partial \sigma(\mathbf{W}^{(1)}\boldsymbol{x})} \cdot \frac{\partial \sigma(\mathbf{W}^{(1)}\boldsymbol{x})}{\partial \mathbf{W}^{(1)}\boldsymbol{x}} \cdot \frac{\partial \mathbf{W}^{(1)}\boldsymbol{x}}{\partial \mathbf{W}^{(1)}} \end{aligned} \tag{18}$$

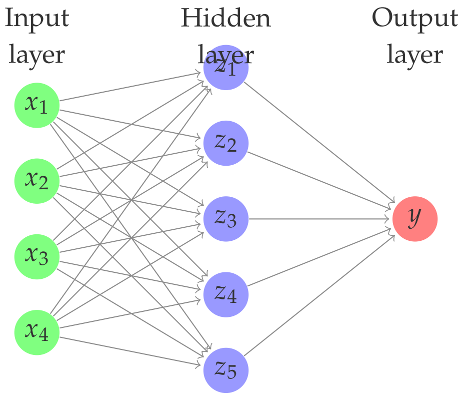
- Both of them are the applications of the chain rule in calculus plus some derivatives of basic functions
- In the literature of neural networks, it is called the back-propagation algorithm [Rumelhart et al., 1986]

Neural Text Classifiers

Network Architecture

We are going to build a simple neural network for text classification.
It includes three layers as the previous example

- Input layer
- Hidden layer
- Output layer



Example

Consider the following special case, where we have a 4-dimensional BoW representation $\boldsymbol{x} \in \mathbb{R}^4$ and a weight matrix $\boldsymbol{W} \in \mathbb{R}^{5\times4}$

$$\boldsymbol{W}\boldsymbol{x} = \begin{bmatrix} 0.1 & 0.3 & 0.7 & 0.9 \\ 0.2 & 0.8 & 0.3 & 0.5 \\ 0.4 & 0.8 & 0.6 & 0.1 \\ 0.7 & 0.2 & 0.9 & 0.2 \\ 0.4 & 0.5 & 0.8 & 0.9 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \qquad (19)$$

$$(20)$$

Example

Consider the following special case, where we have a 4-dimensional BoW representation $\boldsymbol{x} \in \mathbb{R}^4$ and a weight matrix $\boldsymbol{W} \in \mathbb{R}^{5\times 4}$

$$\boldsymbol{W}\boldsymbol{x} = \begin{bmatrix} 0.1 & 0.3 & 0.7 & 0.9 \\ 0.2 & 0.8 & 0.3 & 0.5 \\ 0.4 & 0.8 & 0.6 & 0.1 \\ 0.7 & 0.2 & 0.9 & 0.2 \\ 0.4 & 0.5 & 0.8 & 0.9 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \tag{19}$$

$$= \begin{bmatrix} 0.3 \\ 0.8 \\ 0.8 \\ 0.2 \\ 0.5 \end{bmatrix} + \begin{bmatrix} 0.9 \\ 0.5 \\ 0.1 \\ 0.2 \\ 0.9 \end{bmatrix} \tag{20}$$

Example

Consider the following special case, where we have a 4-dimensional BoW representation $\boldsymbol{x} \in \mathbb{R}^4$ and a weight matrix $\boldsymbol{W} \in \mathbb{R}^{5\times4}$

$$\begin{aligned}
\boldsymbol{W}\boldsymbol{x} &= \begin{bmatrix} 0.1 & 0.3 & 0.7 & 0.9 \\ 0.2 & 0.8 & 0.3 & 0.5 \\ 0.4 & 0.8 & 0.6 & 0.1 \\ 0.7 & 0.2 & 0.9 & 0.2 \\ 0.4 & 0.5 & 0.8 & 0.9 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} && (19) \\
&= \begin{bmatrix} 0.3 \\ 0.8 \\ 0.8 \\ 0.2 \\ 0.5 \end{bmatrix} + \begin{bmatrix} 0.9 \\ 0.5 \\ 0.1 \\ 0.2 \\ 0.9 \end{bmatrix} && (20)
\end{aligned}$$

- Each column vector in $\boldsymbol{W}$ corresponds one word in the BoW representation
- The column vectors can be considered as representations of words, in other words, *word embeddings*

Data Processing

Similar to building a logistic regression classifier, we need the following steps in data processing

- Tokenize texts
- Build a vocab
- Convert a text into a collection of word indices (instead of using BoW representations)

Data Processing

Similar to building a logistic regression classifier, we need the following steps in data processing

- Tokenize texts
- Build a vocab
- Convert a text into a collection of word indices (instead of using BoW representations)

In addition, we also need to divide the whole set of texts into small batches

- Create mini-batches

Input Layer

In the mini-batch setting, the model usually takes a subset of texts instead of one single text. The input $\boldsymbol{X}$ represents a word index matrix, where each column vector $\boldsymbol{x}_i$ is the word indices from a corresponding text with paddings

$$\boldsymbol{X} = \begin{bmatrix} 12 & 31 & \cdots & 11 & 2 \\ 5 & 16 & \cdots & 15 & 8 \\ 9 & 1 & \cdots & 21 & 10 \\ 1 & 1 & \cdots & 7 & 1 \end{bmatrix} \in \mathbb{R}^{L \times B} \tag{21}$$

- This is a slightly different representation, compared to BoW

Input Layer

In the mini-batch setting, the model usually takes a subset of texts instead of one single text. The input $\boldsymbol{X}$ represents a word index matrix, where each column vector $\boldsymbol{x}_i$ is the word indices from a corresponding text with paddings

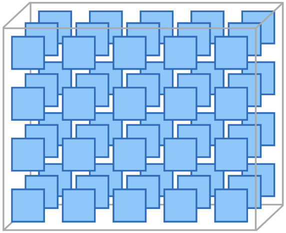
$$\boldsymbol{X} = \begin{bmatrix} 12 & 31 & \cdots & 11 & 2 \\ 5 & 16 & \cdots & 15 & 8 \\ 9 & 1 & \cdots & 21 & 10 \\ 1 & 1 & \cdots & 7 & 1 \end{bmatrix} \in \mathbb{R}^{L\times B} \tag{21}$$

- This is a slightly different representation, compared to BoW
- B: the mini batch size
- L: the biggest length of a text in the mini batch
- Add extra padding tokens will make sentences in the mini batch have the same length

Hidden Layer

With the word index matrix, the hidden layer of a neural network compute the representation of texts with the following steps

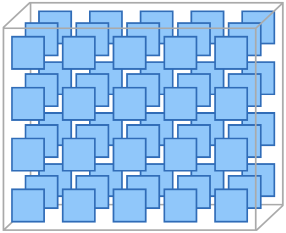
1. Create a mode-3 tensor $\mathcal{X} \in \mathbb{R}^{L \times B \times E}$ by representing each word with a vector (word embedding)



Hidden Layer

With the word index matrix, the hidden layer of a neural network compute the representation of texts with the following steps

1. Create a mode-3 tensor $\mathcal{X} \in \mathbb{R}^{L \times B \times E}$ by representing each word with a vector (word embedding)

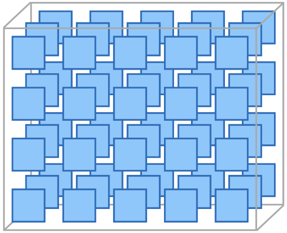


2. Create text representations by summing over the first dimension, which gives a matrix $\boldsymbol{H}$ with size $B \times E$

Hidden Layer

With the word index matrix, the hidden layer of a neural network compute the representation of texts with the following steps

1. Create a mode-3 tensor $\mathcal{X} \in \mathbb{R}^{L \times B \times E}$ by representing each word with a vector (word embedding)



2. Create text representations by summing over the first dimension, which gives a matrix $\boldsymbol{H}$ with size $B \times E$
3. Go through an element-wise activation function, e.g., the Sigmoid function.

Output Layer

With the representations of texts $\boldsymbol{H} \in \mathbb{R}^{B \times E}$, the output layer is nothing different from the logistic regression model.

Output Layer

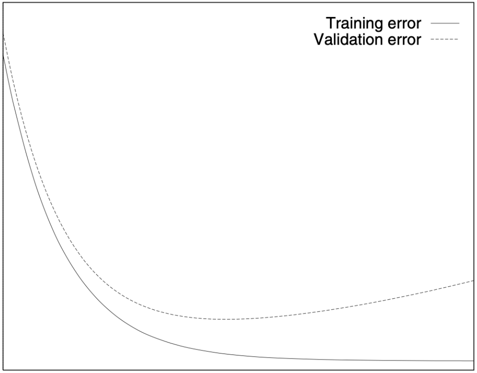
With the representations of texts $\boldsymbol{H} \in \mathbb{R}^{B \times E}$, the output layer is nothing different from the logistic regression model.

Checkout the demo code!

Neural Networks: Tricks of the Trade

Early Stopping

The idealized training and validation error curves: Vertical axis – errors, horizontal axis – time

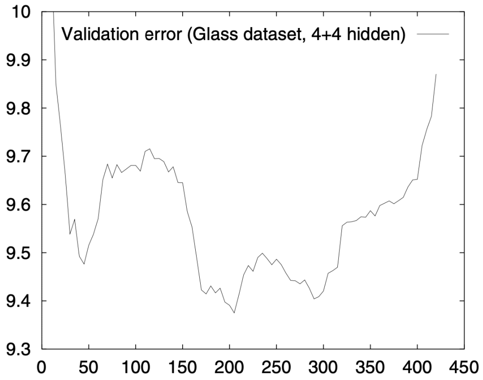


Stop training as soon as the validation error increases

[Montavon et al., 2012]

Early Stopping (II)

A real validation error during training



- Use check points and always the best model according to the validation performance
- Stop training if the validation error keep increasing for a *while* (?)

Regularization: Weight Decay

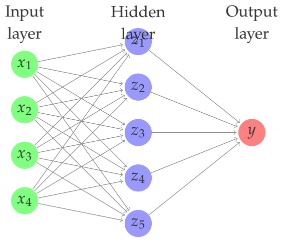
Add a small modification to the gradient of $\boldsymbol{\theta}$, such that

$$\boldsymbol{\theta}^{(\text{new})} \leftarrow \boldsymbol{\theta}^{(\text{old})} - \eta \cdot \left(\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(\text{old})}} + \alpha \boldsymbol{\theta}^{(\text{old})}\right) \tag{22}$$

- It is equivalent to the ℓ_2 regularization
- Proposed in [Plaut et al., 1986] based on a different intuition

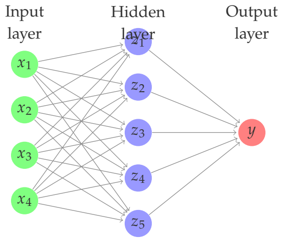
Regularization: Dropout

Randomly remove some components from the input layer during training



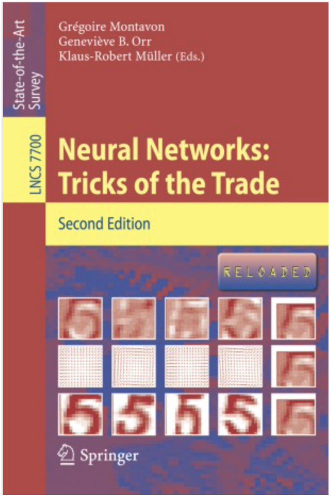
Regularization: Dropout

Randomly remove some components from the input layer during training



- Proposed in [Hinton et al., 2012] and later published as [Srivastava et al., 2014]
- Provides a regularization effect that reduce the correlation with noisy features
- Can also be applied to hidden layers

Neural Networks: Tricks of the Trade


State-of-the-Art Survey
LNCS 7700
Grégoire Montavon
Geneviève B. Orr
Klaus-Robert Müller (Eds.)
Neural Networks: Tricks of the Trade
Second Edition

Neural Networks: Tricks of the Trade

Example Applications

Multi-choice Question Answering

Question

George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat?

Choices

A : dry palms

B : wet palms

C : palms covered with oil

D : palms covered with lotion

Correct Answer (Label)

A

Multi-choice Question Answering

Question

George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat?

Choices

A : dry palms

B : wet palms

C : palms covered with oil

D : palms covered with lotion

Correct Answer (Label)

A

To form the training set with examples like this, what information should be in the input/output?

Option 1

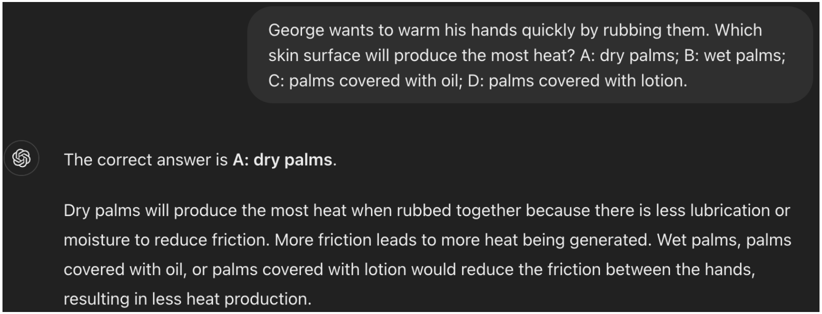
Input

George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? A: dry palms; B: wet palms; C: palms covered with oil; D: palms covered with lotion.

Output

A

ChatGPT


George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? A: dry palms; B: wet palms; C: palms covered with oil; D: palms covered with lotion.

The correct answer is **A: dry palms**.

Dry palms will produce the most heat when rubbed together because there is less lubrication or moisture to reduce friction. More friction leads to more heat being generated. Wet palms, palms covered with oil, or palms covered with lotion would reduce the friction between the hands, resulting in less heat production.

Option 2

Input

George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? A: dry palms; B: wet plams; C: palms covered with oil; D: palms covered with lotion.

Output

A: dry palms

Option 2

Input

George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? A: dry palms; B: wet plams; C: palms covered with oil; D: palms covered with lotion.

Output

A: dry palms

It can be used with generative models, but not classification models

Option 3

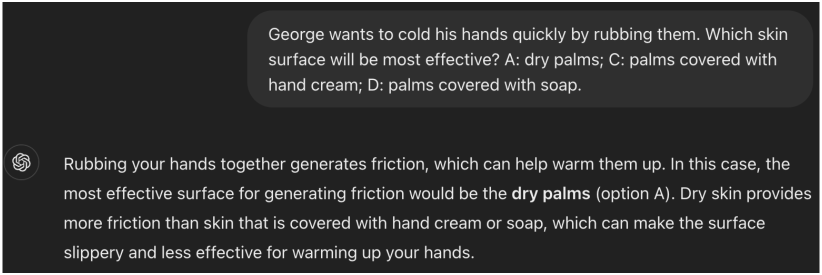
George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? A: dry palms	True
George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? B: wet palms	False
George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? C: palms covered with oil	False
George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? D: palms covered with lotion	False

Option 3

George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? A: dry palms	True
George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? B: wet palms	False
George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? C: palms covered with oil	False
George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? D: palms covered with lotion	False

We can even get rid of the `A, B, C, D` from the strings

ChatGPT Again



George wants to cold his hands quickly by rubbing them. Which skin surface will be most effective? A: dry palms; C: palms covered with hand cream; D: palms covered with soap.

Rubbing your hands together generates friction, which can help warm them up. In this case, the most effective surface for generating friction would be the **dry palms** (option A). Dry skin provides more friction than skin that is covered with hand cream or soap, which can make the surface slippery and less effective for warming up your hands.

Summary

1. From Logistic Regression to Neural Networks
2. Learning Neural Networks
3. Neural Text Classifiers
4. Neural Networks: Tricks of the Trade
5. Example Applications

Reference

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012).
Improving neural networks by preventing co-adaptation of feature detectors.
arXiv preprint arXiv:1207.0580.

Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009).
What is the best multi-stage architecture for object recognition?
In *Proceedings of the 12th International Conference on Computer Vision*, pages 2146–2153. IEEE.

Montavon, G., Orr, G., and Müller, K.-R. (2012).
Neural networks-tricks of the trade second edition.
Springer.

Plaut, D. C., Nowlan, S., and Hinton, G. (1986).
Experiments on learning by back propagation.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986).
Learning representations by back-propagating errors.
Nature, 323(6088):533–536.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014).
Dropout: a simple way to prevent neural networks from overfitting.
The journal of machine learning research, 15(1):1929–1958.