CS 4501/6501 Interpretable Machine Learning

Neural Networks and Deep Learning

Yangfeng Ji

Department of Computer Science
University of Virginia

Overview

1. What is a white box looks like?
2. What is a neural network?
3. Why we need neural network models?
4. Why neural network is a black box?

What is a white box looks like?

Linear Models

Directly modeling a linear classifier as

$$h_y(\boldsymbol{x}) = \boldsymbol{w}_y^\top \boldsymbol{x} + b_y \tag{1}$$

- $\boldsymbol{x} \in \mathbb{N}^V$: vector, bag-of-words representation
- $\boldsymbol{w}_y \in \mathbb{R}^V$: vector, classification weights associated with label y
- $b_y \in \mathbb{R}$: scalar, label bias in the training set y

Logistic Regression

Rewrite the linear decision function in the log probabilistic form

$$\log P(y \mid \boldsymbol{x}) \propto \underbrace{\boldsymbol{w}_y^\top \boldsymbol{x} + b_y}_{h_y(\boldsymbol{x})} \tag{2}$$

Logistic Regression

Rewrite the linear decision function in the log probabilistic form

$$\log P(y \mid \boldsymbol{x}) \propto \underbrace{\boldsymbol{w}_y^\top \boldsymbol{x} + b_y}_{h_y(\boldsymbol{x})} \tag{2}$$

or, the probabilistic form is

$$P(y \mid \boldsymbol{x}) \propto \exp(\boldsymbol{w}_y^\top \boldsymbol{x} + b_y) \tag{3}$$

Logistic Regression

Rewrite the linear decision function in the log probabilistic form

$$\log P(y \mid \boldsymbol{x}) \propto \underbrace{\boldsymbol{w}_y^\top \boldsymbol{x} + b_y}_{h_y(\boldsymbol{x})} \tag{2}$$

or, the probabilistic form is

$$P(y \mid \boldsymbol{x}) \propto \exp(\boldsymbol{w}_y^\top \boldsymbol{x} + b_y) \tag{3}$$

To make sure $P(y \mid \boldsymbol{x})$ is a valid definition of probability, we need to make sure $\sum_y P(y \mid \boldsymbol{x}) = 1$,

$$P(y \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_y^\top \boldsymbol{x} + b_y)}{\sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w}_{y'}^\top \boldsymbol{x} + b_{y'})} \tag{4}$$

Alternative Form

Rewriting $\boldsymbol{x}$ and $\boldsymbol{w}$ as

- $\boldsymbol{x}^{\mathsf{T}} = [x_1, x_2, \cdots, x_V, 1]$
- $\boldsymbol{w}_y^{\mathsf{T}} = [w_1, w_2, \cdots, w_V, b_y]$

allows us to have a more concise form

$$P(y \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_y^{\mathsf{T}}\boldsymbol{x})}{\sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w}_{y'}^{\mathsf{T}}\boldsymbol{x})} \tag{5}$$

Alternative Form

Rewriting $\boldsymbol{x}$ and $\boldsymbol{w}$ as

- $\boldsymbol{x}^{\top} = [x_1, x_2, \cdots, x_V, 1]$
- $\boldsymbol{w}_y^{\top} = [w_1, w_2, \cdots, w_V, b_y]$

allows us to have a more concise form

$$P(y \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_y^{\top}\boldsymbol{x})}{\sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w}_{y'}^{\top}\boldsymbol{x})} \tag{5}$$

Comments:

- $\frac{\exp(a)}{\sum_{a'} \exp(a')}$ is the softmax function
- This form works with any size of $\mathcal{Y}$ — it does not have to be a binary classification problem.

Binary Classifier

Assume $\mathcal{Y} = \{\textsc{neg}, \textsc{pos}\}$, then the corresponding logistic regression classifier with $Y = \textsc{Pos}$ is

$$P(Y = \textsc{Pos} \mid \boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{w}^\top \boldsymbol{x})} \tag{6}$$

where $\boldsymbol{w}$ is the only parameter.

Binary Classifier

Assume $\mathcal{Y} = \{\textsc{neg}, \textsc{pos}\}$, then the corresponding logistic regression classifier with $Y = \textsc{Pos}$ is

$$P(Y = \textsc{Pos} \mid \boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x})} \tag{6}$$

where $\boldsymbol{w}$ is the only parameter.

- $P(Y = \textsc{Neg} \mid \boldsymbol{x}) = 1 - P(Y = \textsc{Pos} \mid \boldsymbol{x})$

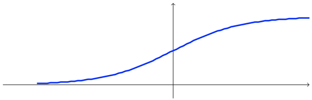
Binary Classifier

Assume $\mathcal{Y} = \{\textsc{neg}, \textsc{pos}\}$, then the corresponding logistic regression classifier with $Y = \textsc{Pos}$ is

$$P(Y = \textsc{Pos} \mid \boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{w}^{\top}\boldsymbol{x})} \tag{6}$$

where $\boldsymbol{w}$ is the only parameter.

- $P(Y = \textsc{Neg} \mid \boldsymbol{x}) = 1 - P(Y = \textsc{Pos} \mid \boldsymbol{x})$
- $\frac{1}{1+\exp(-z)}$ is the Sigmoid function



Two Questions on Building LR Models

... of building a logistic regression classifier

$$P(y \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_y^{\top}\boldsymbol{x})}{\sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w}_{y'}^{\top}\boldsymbol{x})} \tag{7}$$

- How to learn the parameters $\boldsymbol{W} = \{\boldsymbol{w}_y\}_{y \in \mathcal{Y}}$?

Two Questions on Building LR Models

... of building a logistic regression classifier

$$P(y \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_y^\top \boldsymbol{x})}{\sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w}_{y'}^\top \boldsymbol{x})} \tag{7}$$

- How to learn the parameters $\boldsymbol{W} = \{\boldsymbol{w}_y\}_{y \in \mathcal{Y}}$?
- Can $\boldsymbol{x}$ be better than the bag-of-words representations?

Review: (Log)-likelihood Function

With a collection of training examples $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$, the likelihood function of $\{\boldsymbol{w}_y\}_{y\in\mathcal{Y}}$ is

$$L(\boldsymbol{W}) = \prod_{i=1}^{m} P(y^{(i)} \mid \boldsymbol{x}^{(i)}) \tag{8}$$

and the log-likelihood function is

$$\ell(\{\boldsymbol{w}_y\}) = \sum_{i=1}^{m} \log P(y^{(i)} \mid \boldsymbol{x}^{(i)}) \tag{9}$$

Log-likelihood Function of a LR Model

With the definition of a LR model

$$P(y \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_y^\top \boldsymbol{x})}{\sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w}_{y'}^\top \boldsymbol{x})} \tag{10}$$

the log-likelihood function is

$$\begin{aligned} \ell(\boldsymbol{W}) &= \sum_{i=1}^{m} \log P(y^{(i)} \mid \boldsymbol{x}^{(i)}) && (11) \\ &= \sum_{i=1}^{m} \left\{ \boldsymbol{w}_{y^{(i)}}^\top \boldsymbol{x}^{(i)} - \log \sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w}_{y'}^\top \boldsymbol{x}^{(i)}) \right\} && (12) \end{aligned}$$

Given the training examples $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$, $\ell(\boldsymbol{W})$ is a function of $\boldsymbol{W} = \{\boldsymbol{w}_y\}$.

Optimization with Gradient

MLE is equivalent to minimize the Negative Log-Likelihood (NLL) as

$$\begin{aligned}\mathrm{NLL}(\boldsymbol{W}) &= -\ell(\boldsymbol{W}) \\ &= \sum_{i=1}^{m}\Big\{-\boldsymbol{w}_{y^{(i)}}^{\top}\boldsymbol{x}^{(i)} + \log\sum_{y'\in\mathcal{Y}}\exp(\boldsymbol{w}_{y'}^{\top}\boldsymbol{x})\Big\}\end{aligned}$$

then, the parameter $\boldsymbol{w}_y$ associated with label y can be updated as

$$\boldsymbol{w}_y \leftarrow \boldsymbol{w}_y - \eta \cdot \frac{\partial \mathrm{NLL}(\{\boldsymbol{w}_y\})}{\partial \boldsymbol{w}_y}, \quad \forall y \in \mathcal{Y} \tag{13}$$

where η is called **learning rate**.

Optimization with Gradient (II)

Two questions answered by the update equation

(1) which direction?
(2) how far it should go?

[Jurafsky and Martin, 2022]

Optimization with Gradient (II)

Two questions answered by the update equation

(1) which direction?

(2) how far it should go?

$$\boldsymbol{w}_y \leftarrow \boldsymbol{w}_y - \underbrace{\eta}_{(2)} \cdot \underbrace{\frac{\partial \mathrm{NLL}(\{\boldsymbol{w}_y\})}{\partial \boldsymbol{w}_y}}_{(1)} \tag{14}$$

[Jurafsky and Martin, 2022]

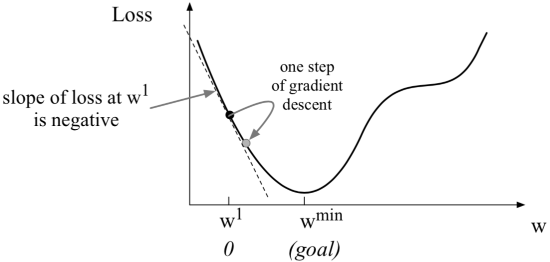
Optimization with Gradient (II)

Two questions answered by the update equation

(1) which direction?

(2) how far it should go?

$$w_y \leftarrow w_y - \underbrace{\eta}_{(2)} \cdot \underbrace{\frac{\partial \text{NLL}(\{w_y\})}{\partial w_y}}_{(1)} \tag{14}$$



[Jurafsky and Martin, 2022]

Training Procedure

Steps for parameter estimation, given the current parameter $\{\boldsymbol{w}_y\}$

1. Compute the derivative

$$\frac{\partial \mathrm{NLL}(\{\boldsymbol{w}_y\})}{\partial \boldsymbol{w}_y}, \quad \forall y \in \mathcal{Y}$$

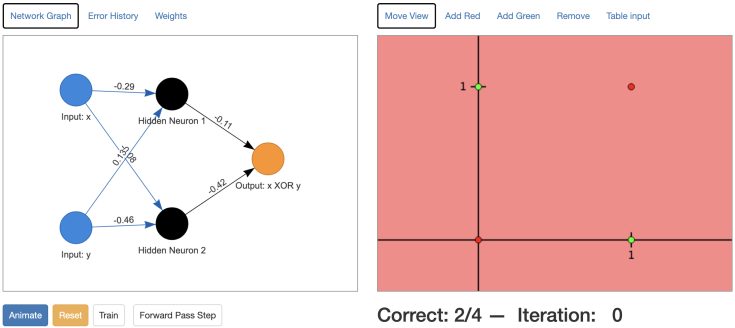
2. Update parameters with

$$\boldsymbol{w}_y \leftarrow \boldsymbol{w}_y - \eta \cdot \frac{\partial \mathrm{NLL}(\{\boldsymbol{w}_y\})}{\partial \boldsymbol{w}_y}, \quad \forall y \in \mathcal{Y}$$

3. If not done, return to step 1

Neural Network Demo

A simple demo with 2-dimensional inputs



`https://phiresky.github.io/neural-network-demo/`

Sentiment Classification

A few training examples from Yelp Review

Label	Text
5	`Love the staff, love the meat, love the place. Prepare for a long line around lunch or dinner hours ...`
5	`Super simple place but amazing nonetheless. It's been around since the 30's and they still serve the same thing ...`

Sentiment Classification

A few training examples from Yelp Review

Label	Text
5	`Love the staff, love the meat, love the place. Prepare for a long line around lunch or dinner hours ...`
5	`Super simple place but amazing nonetheless. It's been around since the 30's and they still serve the same thing ...`
1	`Actually I would like to give them a big fat zero. Any vet's office that would tell ...`

Sentiment Classification

A few training examples from Yelp Review

Label	Text
5	Love the staff, love the meat, love the place. Prepare for a long line around lunch or dinner hours ...
5	Super simple place but amazing nonetheless. It's been around since the 30's and they still serve the same thing ...
1	Actually I would like to give them a big fat zero. Any vet's office that would tell ...
2	OK so first off the the burger was great as far as the taste. But I got super sick after eating it ...

Classification Configuration and Performance

Vocabulary size	17,490
Training size	40K
Development size	5K
Classifier	Logistic regression
Regularization parameter	$C = 1$
Training accuracy	88.48%
Development accuracy	61.22%

You can find the demo code via the link

Interpretability: Global

RATING	FEATURES
1	worst awful horrible disgusting disgusted joke terrible zero luck pathetic
2	meh mediocre tacky nope renovations cheddar sand-which underwhelmed passable tasteless

Interpretability: Global

RATING	FEATURES
1	worst awful horrible disgusting disgusted joke terrible zero luck pathetic
2	meh mediocre tacky nope renovations cheddar sandwhich underwhelmed passable tasteless
3	feelings mains healthier remind hearty bleu overrated unsure smelling rules

Interpretability: Global

RATING	FEATURES
1	worst awful horrible disgusting disgusted joke terrible zero luck pathetic
2	meh mediocre tacky nope renovations cheddar sandwhich underwhelmed passable tasteless
3	feelings mains healthier remind hearty bleu overrated unsure smelling rules
4	default hankering drawback bojangles pleasantly hazelnut customize gratuity excellent tremendously
5	phenomenal incredible amazing gem excellent pleasantly hesitate master magnificent spotless

Interpretability: Local

The prediction on the following is 5

```
I love the service here , they ’ re on it !
```

After pre-processing, we remove the high-frequency word `I` and the punctuation

Interpretability: Local

The prediction on the following is 5

`I love the service here , they ’ re on it !`

After pre-processing, we remove the high-frequency word `I` and the punctuation

FEATURE	CLASSIFICATION WEIGHT
`love`	0.85
`service`	0.04
`on`	0.01
`they`	-0.00
`it`	-0.02
`the`	-0.06
`re`	-0.13
`here`	-0.15

What is a neural network?

Logistic Regression

- An unified form for $y \in \{-1, +1\}$

$$p(Y = +1 \mid \boldsymbol{x}) = \frac{1}{1 + \exp(-\langle \boldsymbol{w}, \boldsymbol{x} \rangle)} \tag{15}$$

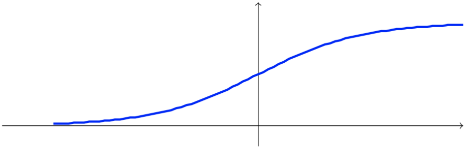
Logistic Regression

- An unified form for $y \in \{-1, +1\}$

$$p(Y = +1 \mid \boldsymbol{x}) = \frac{1}{1 + \exp(-\langle \boldsymbol{w}, \boldsymbol{x} \rangle)} \tag{15}$$

- The sigmoid function $\sigma(a)$ with $a \in \mathbb{R}$

$$\sigma(a) = \frac{1}{1 + \exp(-a)} \tag{16}$$

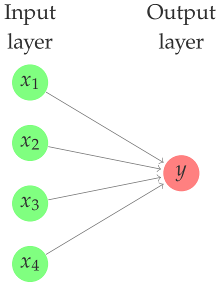


Graphical Representation

- A specific example of LR

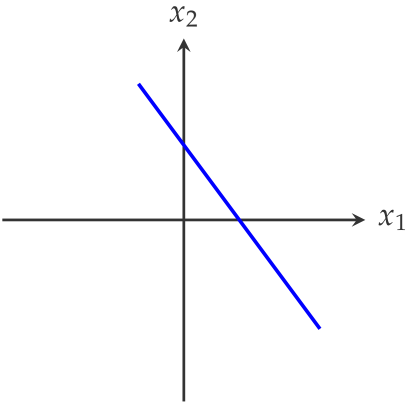
$$p(Y = 1 \mid \boldsymbol{x}) = \sigma(\sum_{j=1}^{4} w_j \boldsymbol{x}_{\cdot,j}) \tag{17}$$

- The graphical representation of this LR model is



Capacity of a LR

Logistic regression gives a linear decision boundary



From LR to Neural Networks

Build upon logistic regression, a simple neural network can be constructed as

$$
\begin{aligned}
z_k &= \sigma\Big(\sum_{j=1}^{d} w_{k,j}^{(1)} x_{\cdot,j}\Big) \quad k \in [K] && (18) \\
P(y = 1 \mid \boldsymbol{x}) &= \sigma\Big(\sum_{k=1}^{K} w_k^{(o)} z_k\Big) && (19)
\end{aligned}
$$

- $\boldsymbol{x} \in \mathbb{R}^d$: d-dimensional input
- $y \in \{-1, +1\}$ (binary classification problem)

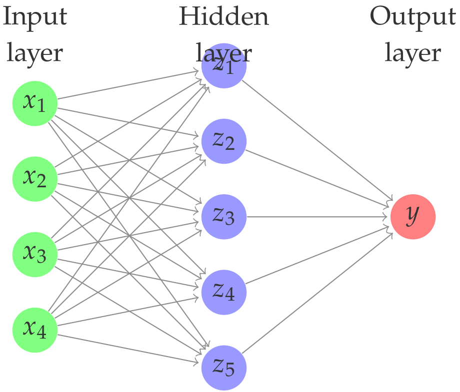
From LR to Neural Networks

Build upon logistic regression, a simple neural network can be constructed as

$$
\begin{aligned}
z_k &= \sigma\Big(\sum_{j=1}^{d} w_{k,j}^{(1)} x_{\cdot,j}\Big) \quad k \in [K] && (18) \\
P(y = 1 \mid \boldsymbol{x}) &= \sigma\Big(\sum_{k=1}^{K} w_k^{(o)} z_k\Big) && (19)
\end{aligned}
$$

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

Graphical Representation

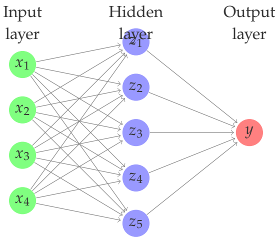


- Depth: 2 (two-layer neural network)
- Width: 5 (the maximal number of units in each layer)

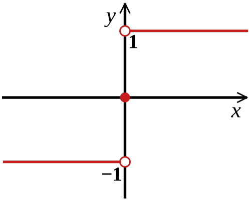
Hypothesis Space

The hypothesis space of neural networks is usually defined by the architecture of the network, which includes

- the nodes in the network,
- the connections in the network, and
- the activation function (e.g., σ)

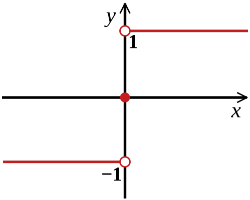


Other Activation Functions

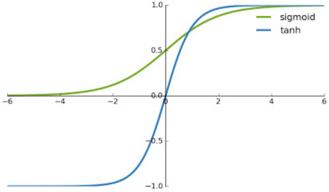


(a) Sign function

Other Activation Functions

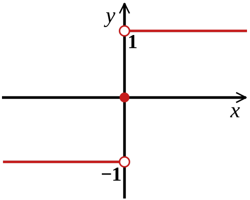


(a) Sign function

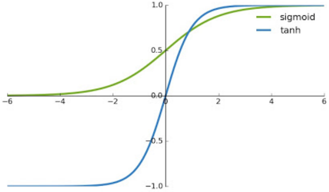


(b) Tanh function

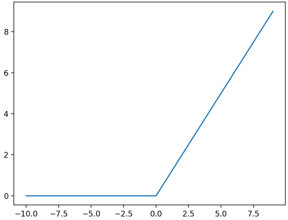
Other Activation Functions



(a) Sign function



(b) Tanh function



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

- Element-wise formulation

$$
\begin{aligned}
z_k &= \sigma(\sum_{j=1}^{d} w_{k,j}^{(1)} x_j) \quad k \in [K] \qquad (20) \\
P(y = +1 \mid \boldsymbol{x}) &= \sigma(\sum_{k=1}^{K} w_k^{(o)} z_k) \qquad (21)
\end{aligned}
$$

Mathematical Formulation

- Element-wise formulation

$$z_k = \sigma(\sum_{j=1}^{d} w_{k,j}^{(1)} x_j) \quad k \in [K] \tag{20}$$

$$P(y = +1 \mid \boldsymbol{x}) = \sigma(\sum_{k=1}^{K} w_k^{(o)} z_k) \tag{21}$$

- Matrix-vector formulation

$$\boldsymbol{z} = \sigma(\mathbf{W}\boldsymbol{x}) \tag{22}$$

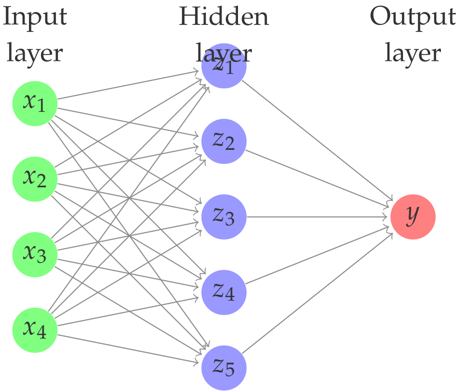
$$P(y = +1 \mid \boldsymbol{x}) = \sigma(\boldsymbol{w}^{\top}\boldsymbol{z}) \tag{23}$$

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

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 $\boldsymbol{W} \in \mathbb{R}^{5\times4}$

$$\boldsymbol{W}\boldsymbol{x} = \begin{bmatrix} 0.1 & 0.3 & 0.7 & 0.9 \\ 0.2 & 0.8 & 0.3 & 0.5 \\ 0.4 & 0.8 & 0.6 & 0.1 \\ 0.7 & 0.2 & 0.9 & 0.2 \\ 0.4 & 0.5 & 0.8 & 0.9 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \tag{24}$$

$$\tag{25}$$

Example

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

$$\boldsymbol{W}\boldsymbol{x} = \begin{bmatrix} 0.1 & 0.3 & 0.7 & 0.9 \\ 0.2 & 0.8 & 0.3 & 0.5 \\ 0.4 & 0.8 & 0.6 & 0.1 \\ 0.7 & 0.2 & 0.9 & 0.2 \\ 0.4 & 0.5 & 0.8 & 0.9 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \tag{24}$$

$$= \begin{bmatrix} 0.3 \\ 0.8 \\ 0.8 \\ 0.2 \\ 0.5 \end{bmatrix} + \begin{bmatrix} 0.9 \\ 0.5 \\ 0.1 \\ 0.2 \\ 0.9 \end{bmatrix} \tag{25}$$

Example

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

$$\boldsymbol{W}\boldsymbol{x} = \begin{bmatrix} 0.1 & 0.3 & 0.7 & 0.9 \\ 0.2 & 0.8 & 0.3 & 0.5 \\ 0.4 & 0.8 & 0.6 & 0.1 \\ 0.7 & 0.2 & 0.9 & 0.2 \\ 0.4 & 0.5 & 0.8 & 0.9 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \tag{24}$$

$$= \begin{bmatrix} 0.3 \\ 0.8 \\ 0.8 \\ 0.2 \\ 0.5 \end{bmatrix} + \begin{bmatrix} 0.9 \\ 0.5 \\ 0.1 \\ 0.2 \\ 0.9 \end{bmatrix} \tag{25}$$

- 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*

Output Layer

For the same text classification task, we use the following configuration:

- Input dimension: $\boldsymbol{x} \in \mathbb{R}^{17K}$
- Hidden layer: $\boldsymbol{h} \in \mathbb{R}^{32}$
- Output layer: $y \in \{1, \ldots, 5\}$

You can find an extremely simple implementation via the same link, the dev accuracy is 65%

Why we need neural network models?

Distributed Representation

- Bag-of-words representations

VOCAB	coffee	love	like	$\cdots$	tea	you
love	(0	1	0	$\cdots$	0	0)

[1] I made up those numbers for illustration purpose

Distributed Representation

- Bag-of-words representations

Vocab	coffee	love	like	⋯	tea	you
`love`	(0	1	0	⋯	0	0)

- Distributed representations[1]

`love`	(0.1	0.8	-1.0	0.3)

[1] I made up those numbers for illustration purpose

Distributed Representation

- Bag-of-words representations

VOCAB	coffee	love	like	⋯	tea	you
love	(0	1	0	⋯	0	0)
like	(0	0	1	⋯	0	0)

- Distributed representations[1]

love	(0.1	0.8	-1.0	0.3)

[1] I made up those numbers for illustration purpose

Distributed Representation

- Bag-of-words representations

VOCAB	coffee	love	like	···	tea	you
love	(0	1	0	···	0	0)
like	(0	0	1	···	0	0)

- Distributed representations[1]

love	(0.1	0.8	-1.0	0.3)
like	(0.2	0.7	-0.9	0.3)

[1] I made up those numbers for illustration purpose

Distributed Representation

- Bag-of-words representations

Vocab	coffee	love	like	⋯	tea	you
love	(0	1	0	⋯	0	0)
like	(0	0	1	⋯	0	0)

- Distributed representations[1]

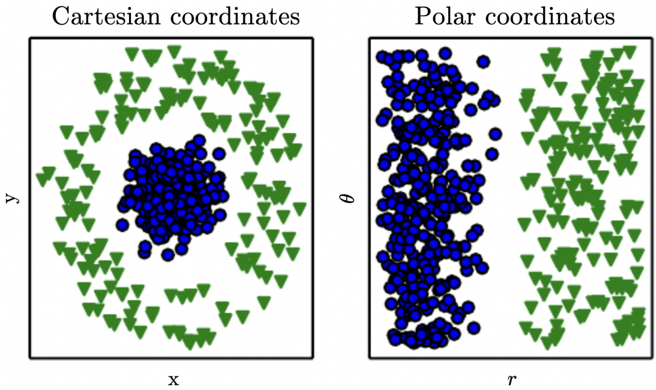
love	(0.1	0.8	-1.0	0.3)
like	(0.2	0.7	-0.9	0.3)

- Distributed representations allow simple algebraic operations for semantic meanings
  - E.g., the cosine value between two word embeddings measures their semantic similarity

[1] I made up those numbers for illustration purpose

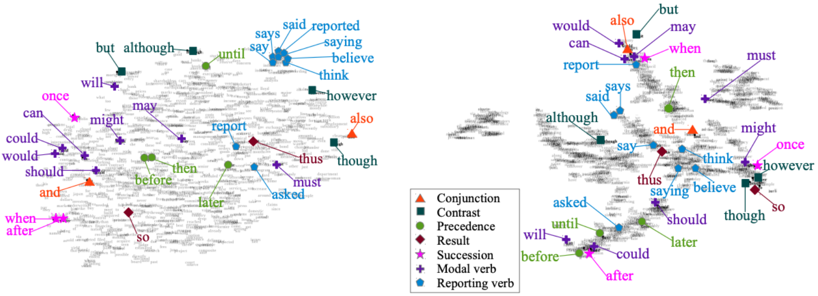
The Value of Different Representations

Different representations lead to different levels of challenge in machine learning



Advantage of Representation Learning

Driven by supervision signals, the model can learn some task-specific information and encoded in word embeddings



Similar advantage exists in any other supervised learning tasks [Bengio et al., 2013]

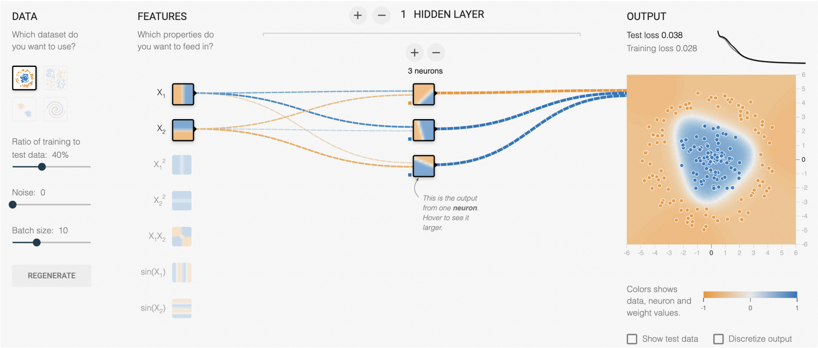
Neural Network as Universal Approximators

Neural network as universal approximators

- With arbitrary width and bounded depth [Cybenko, 1989]
- With arbitrary depth and limited width [Kidger and Lyons, 2020]

Model Capacity via Function Composition

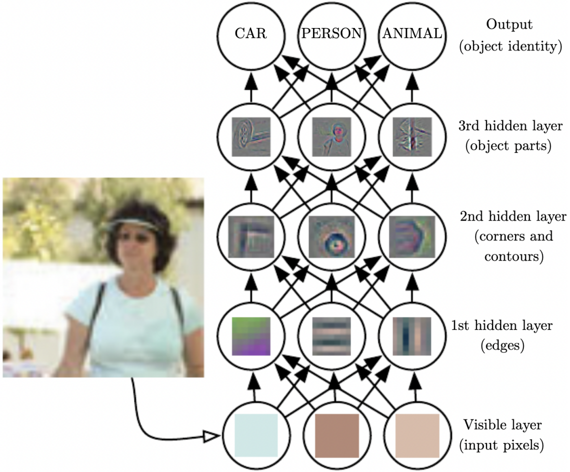
A Toy Example about Function Composition:



https://playground.tensorflow.org/

Model Capacity via Function Composition

An example of function composition to extract high-level features

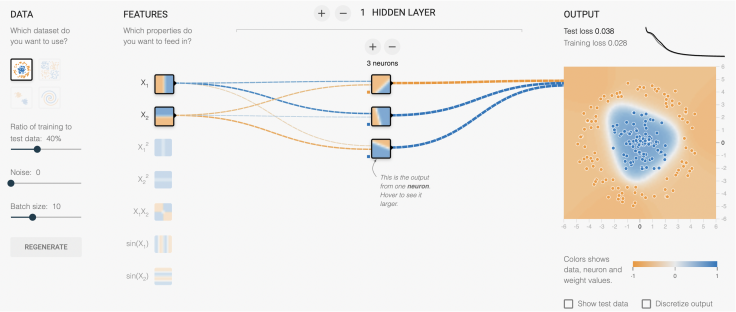


[Goodfellow et al., 2016]

Why neural network is a black box?

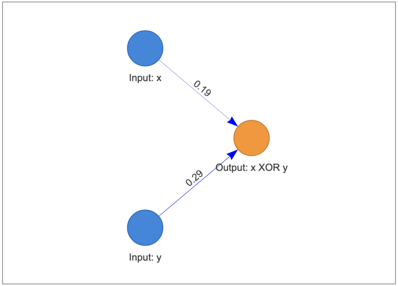
Is Neural Network a Black Box?

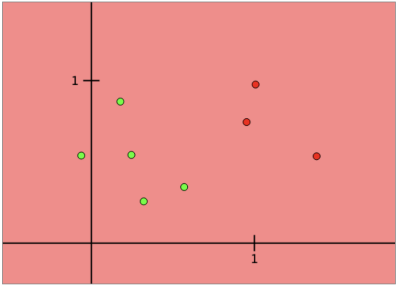
Not exactly, we can analyze what it learns when the model is small



Example

Model interpretability: model predictions can be interpreted as *certain* rules associated with inputs[2]



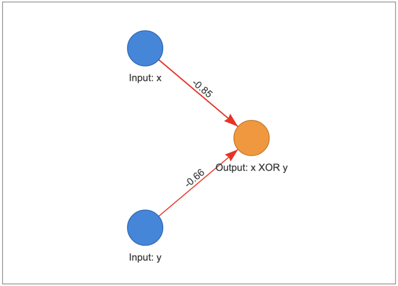


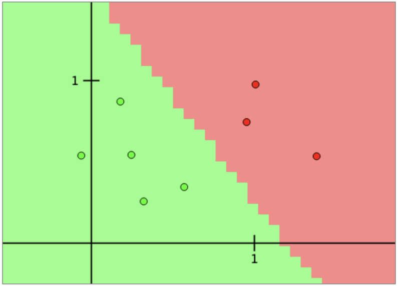
This is equivalent to a logistic regression model

[2]This is by no means a formal definition of interpretability

Example

Model interpretability: model predictions can be interpreted as *certain* rules associated with inputs[2]

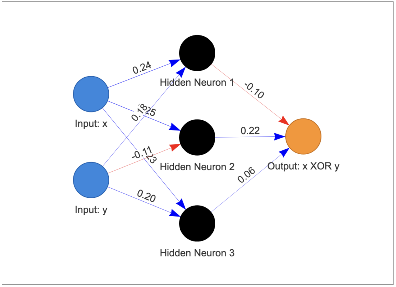


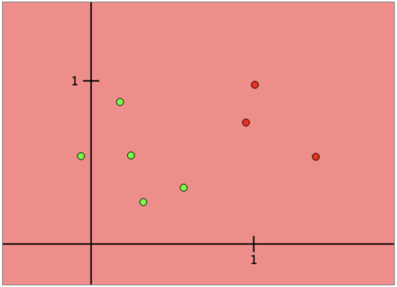


This is equivalent to a logistic regression model

[2]This is by no means a formal definition of interpretability

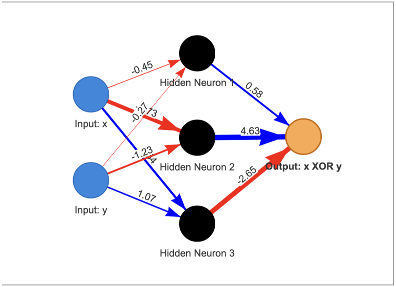
With randomly initialized weights (classifying all examples as negative), the neural network (with one hidden layer) can easily learn a classifier with 100%

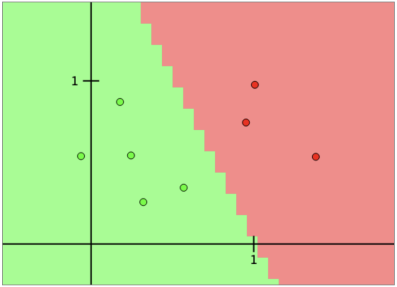




With a Simple Neural Network

With randomly initialized weights (classifying all examples as negative), the neural network (with one hidden layer) can easily learn a classifier with 100%

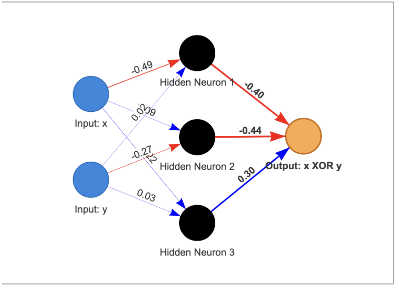


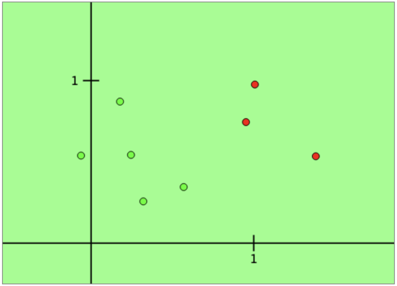


With the visualization, it is not difficult to identify the second and the third neurons are important.

Distributed Representations (II)

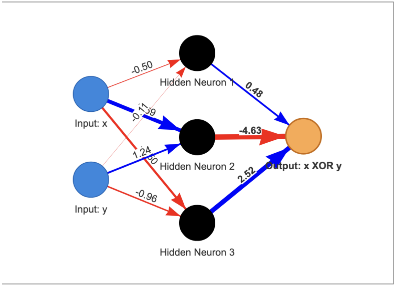
With another set of randomly initialized weights (clasifying all examples as negative), the learned classifier gives a very similar decision boundary

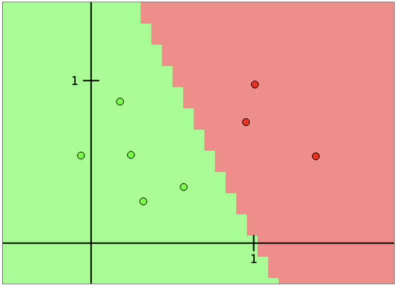




Distributed Representations (II)

With another set of randomly initialized weights (clasifying all examples as negative), the learned classifier gives a very similar decision boundary





With the visualization, it is not difficult to identify the second and the third neurons are important.

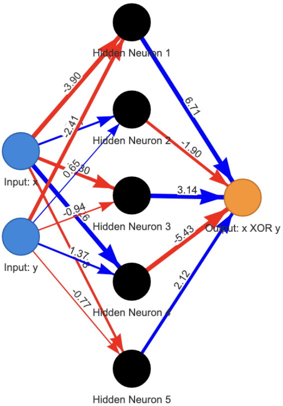
Distributed Representation (III)

- The contributions of the second and third neurons from the these two neural networks are contradicted with each other

Distributed Representation (III)

- The contributions of the second and third neurons from the these two neural networks are contradicted with each other
- Actually, it is still explainable, if we also consider the contribution from the previous layer

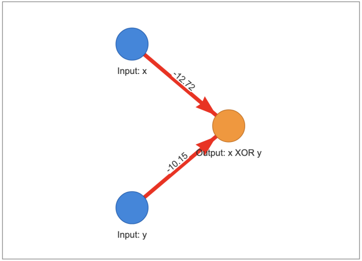
Try to Explain the Following Two Models?

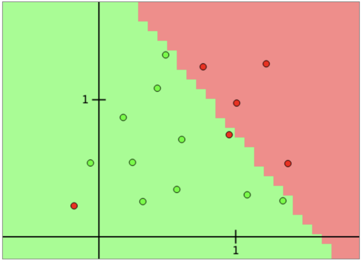


- The neural network has more parameters than the task actually needs
- The contributions of hidden neurons are *randomly* distributed

Number of Layers

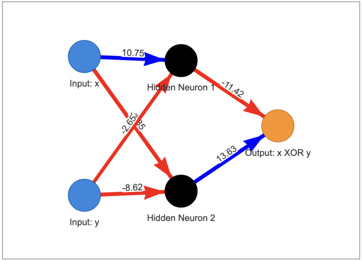
Similar challenge for interpreting the contributions when we increase the number of layers

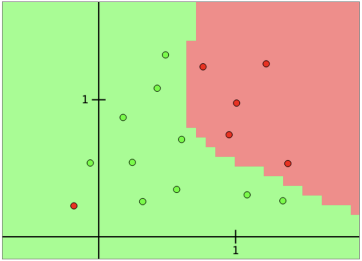




Number of Layers

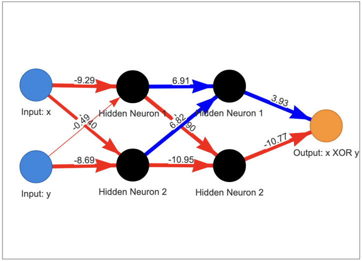
Similar challenge for interpreting the contributions when we increase the number of layers

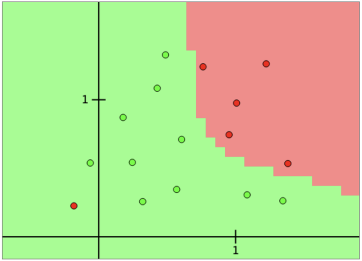




Number of Layers

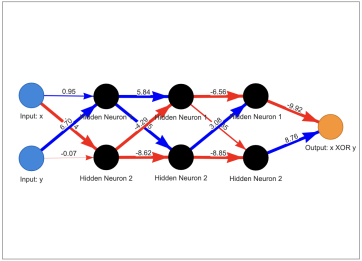
Similar challenge for interpreting the contributions when we increase the number of layers

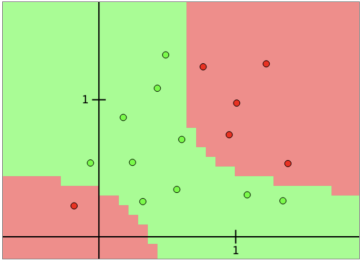




Number of Layers

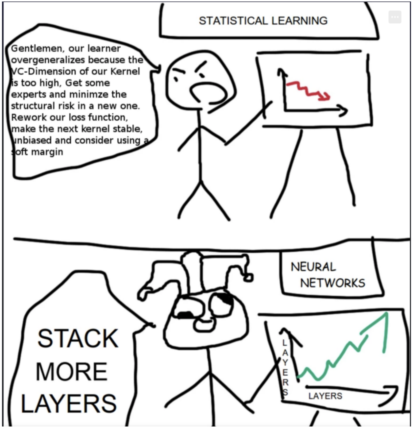
Similar challenge for interpreting the contributions when we increase the number of layers





Number of Layers

Similar challenge for interpreting the contributions when we increase the number of layers



Question

Can we train a neural network that maintains good performance and is also interpretable?

Reference

Bengio, Y., Courville, A., and Vincent, P. (2013).
Representation learning: A review and new perspectives.
IEEE transactions on pattern analysis and machine intelligence, 35(8):1798–1828.

Cybenko, G. (1989).
Approximation by superpositions of a sigmoidal function.
Mathematics of control, signals and systems, 2(4):303–314.

Goodfellow, I., Bengio, Y., and Courville, A. (2016).
Deep Learning.
MIT Press.
`http://www.deeplearningbook.org`.

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.

Jurafsky, D. and Martin, J. H. (2022).
Speech and Language Processing.
Online.

Kidger, P. and Lyons, T. (2020).
Universal approximation with deep narrow networks.
In *Conference on learning theory*, pages 2306–2327. PMLR.