Distributed representation of text

He He

New York University

September 22, 2022

Logistics

- HW1 P1.2 clarification

  2. [3 points] Recall that for multinomial Naive Bayes, we have the input $X = (X_1, \ldots, X_n)$ where n is the number of words in an example. In general, n changes with each example but we can ignore that for now. We assume that $X_i \mid Y = y \sim \text{Categorical}(\theta_{w_1,y}, \ldots \theta_{w_m,y})$ where $Y \in \{0, 1\}$, $w_i \in \mathcal{V}$, and $m = |\mathcal{V}|$ is the vocabulary size. Further, $Y \sim \text{Bernoulli}(\theta_1)$. Show that the multinomial Naive Bayes model has a linear decision boundary, i.e. show that $h(x)$ can be written in the form $w \cdot x + b = 0$. [**RECALL:** The categorical distribution is a multinomial distribution with one trial. Its PMF is

  $$p(x_1, \ldots, x_m) = \prod_{i=1}^{m} \theta_i^{x_i} \ ,$$

  where $x_i = \mathbb{1}[x = i]$, $\sum_{i=1}^{m} x_i = 1$, and $\sum_{i=1}^{m} \theta_i = 1$.]

  x is the BoW vector.

- HW1 due by Sep 29 (one week from now)

Table of Contents

Last week

Generative vs discriminative models for text classification

- (Multinomial) naive Bayes
  - Assumes conditional independence
  - Very efficient in practice (closed-form solution)
- Logistic regression
  - Works with all kinds of features
  - Wins with more data

Feature vector of text input

- BoW representation
- N-gram features (usually $n \leq 3$)

Control the complexity of the hypothesis class

- Feature selection
- Norm regularization

Table of Contents

Objective

Goal: come up with a good representation of text

- What is a representation?
  - Feature map: ϕ: text $\to \mathbb{R}^d$, e.g., BoW, handcrafted features
  - "Representation" often refers to learned features of the input

Objective

Goal: come up with a good representation of text

- What is a representation?
  - Feature map: ϕ: text $\rightarrow \mathbb{R}^d$, e.g., BoW, handcrafted features
  - "Representation" often refers to learned features of the input
- What is a good representation?

Objective

Goal: come up with a good representation of text

- What is a representation?
  - Feature map: ϕ: text $\rightarrow \mathbb{R}^d$, e.g., BoW, handcrafted features
  - "Representation" often refers to learned features of the input
- What is a good representation?
  - Leads to good task performance (often requires less training data)
  - Enables a notion of distance over text: $d(\phi(a), \phi(b))$ is small for semantically similar texts a and b

Distance functions

Let's check if BoW is a good representation.

Euclidean distance

For $a, b \in \mathbb{R}^d$,

$$d(a, b) = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2} \ .$$

Distance functions

Let's check if BoW is a good representation.

Euclidean distance

For $a, b \in \mathbb{R}^d$,

$$d(a, b) = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2} .$$

What if b repeats each sentence in a twice?

Distance functions

Let's check if BoW is a good representation.

Euclidean distance

For $a, b \in \mathbb{R}^d$,

$$d(a, b) = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2} \, .$$

What if b repeats each sentence in a twice? ($b_i = 2a_i$)

Distance functions

Let's check if BoW is a good representation.

Euclidean distance

For $a, b \in \mathbb{R}^d$,

$$d(a, b) = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2} \,.$$

What if b repeats each sentence in a twice? $(b_i = 2a_i)$

Cosine similarity

For $a, b \in \mathbb{R}^d$,

$$\text{sim}(a, b) = \frac{a \cdot b}{\|a\| \|b\|} = \cos \alpha$$

Angle between two vectors

Example: information retrieval

Given a set of documents and a query, use the BoW representation and cosine similarity to find the most relevant document.

What are potential problems?

Example: information retrieval

Given a set of documents and a query, use the BoW representation and cosine similarity to find the most relevant document.

What are potential problems?

- Similarity may be dominated by common words
- Only considers the surface form (e.g., do not account for synonyms)

Example: information retrieval

Given a set of documents and a query, use the BoW representation and cosine similarity to find the most relevant document.

What are potential problems?

- Similarity may be dominated by common words
- Only considers the surface form (e.g., do not account for synonyms)

Example:

Q: Who has watched Tenet?

She has just watched Joker.

Tenet was shown here last week.

TFIDF

Key idea: upweight words that carry more information about the document

Feature map ϕ: document $\rightarrow \mathbb{R}^{|\mathcal{V}|}$

TFIDF:

$$\phi_i(d) = \underbrace{\text{count}(w_i, d)}_{\text{term frequency}} \times \underbrace{\log \frac{\#\ \text{documents}}{\#\ \text{documents containing}\ w_i}}_{\text{inverse document frequency}} .$$

- **Term frequency (TF)**: count of each word type in the document (same as BoW)
- Reweight by **inverse document frequency (IDF)**: how specific is the word type to any particular document
    - Higher for words that only occur in a few documents

Pointwise mutual information

$$\text{PMI}(x;y) \stackrel{\text{def}}{=} \log \frac{p(x,y)}{p(x)p(y)} = \log \frac{p(x \mid y)}{p(x)} = \log \frac{p(y \mid x)}{p(y)}$$

- Symmetric: $\text{PMI}(x;y) = \text{PMI}(y;x)$
- Range: $(-\infty, \min(-\log p(x), -\log p(y)))$
- Estimates:

$$\hat{p}(x \mid y) = \frac{\text{count}(x,y)}{\text{count}(y)} \quad \hat{p}(x) = \frac{\text{count}(x)}{\sum_{x' \in \mathcal{X}} \text{count}(x')}$$

- Positive PMI: $\text{PPMI}(x;y) \stackrel{\text{def}}{=} \max(0, \text{PMI}(x;y))$
- Application in NLP: measure association between words

Justification for TFIDF

Assumptions:

$$p(d) = \frac{1}{\# \text{ of documents}} \tag{1}$$

$$p(d \mid w) = \frac{1}{\# \text{ documents containing } w} \tag{2}$$

Then, we have

$$\mathrm{PMI}(w; d) = \log \frac{p(d \mid w)}{p(d)} = \mathrm{idf}(w, d) \ .$$

IDF measures the association between a term/word and the document

Table of Contents

The distributional hypothesis

"You shall know a word by the company it keeps." (Firth, 1957)

Word guessing! (example from Eisenstein's book)

Everybody likes tezgüino.

The distributional hypothesis

"You shall know a word by the company it keeps." (Firth, 1957)

Word guessing! (example from Eisenstein's book)

Everybody likes tezgüino.

We make tezgüino out of corn.

The distributional hypothesis

"You shall know a word by the company it keeps." (Firth, 1957)

Word guessing! (example from Eisenstein's book)

Everybody likes tezgüino.

We make tezgüino out of corn.

A bottle of tezgüino is on the table.

The distributional hypothesis

"You shall know a word by the company it keeps." (Firth, 1957)

Word guessing! (example from Eisenstein's book)

Everybody likes tezgüino.

We make tezgüino out of corn.

A bottle of tezgüino is on the table.

Don't have tezgüino before you drive.

The distributional hypothesis

"You shall know a word by the company it keeps." (Firth, 1957)

Word guessing! (example from Eisenstein's book)

> Everybody likes tezgüino.
>
> We make tezgüino out of corn.
>
> A bottle of tezgüino is on the table.
>
> Don't have tezgüino before you drive.

Idea: Represent a word by its neighbors.

Choose the context

What are the neighbors? (What type of co-occurence are we interested in?)

Example:

- word $\times$ word

Choose the context

What are the neighbors? (What type of co-occurence are we interested in?)

Example:

- word $\times$ word
- word $\times$ document

Choose the context

What are the neighbors? (What type of co-occurence are we interested in?)

Example:

- word $\times$ word
- word $\times$ document
- note $\times$ song

Choose the context

What are the neighbors? (What type of co-occurence are we interested in?)

Example:

- word $\times$ word
- word $\times$ document
- note $\times$ song
- person $\times$ movie

Choose the context

What are the neighbors? (What type of co-occurence are we interested in?)

Example:

- word × word
- word × document
- note × song
- person × movie

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Figure 6.2 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

Figure: Jurafsky and Martin.

Choose the context

What are the neighbors? (What type of co-occurence are we interested in?)

Example:

- word × word
- word × document
- note × song
- person × movie

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Figure 6.2 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

Figure: Jurafsky and Martin.

Construct a matrix where

- Row and columns represent two sets of objects
- Each entry is the (adjusted) co-occurence counts of the two objects

Reweight counts

Upweight informative words by IDF or PPMI

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Figure 6.9 A tf-idf weighted term-document matrix for four words in four Shakespeare

Figure: Jurafsky and Martin.

Each row/column gives us a word/document representation.

Using cosine similarity, we can cluster documents, find synonyms, discover word meanings...

Dimensionality reduction

Motivation: want a lower-dimensional, dense representation for efficiency

Dimensionality reduction

Motivation: want a lower-dimensional, dense representation for efficiency

Recall **SVD**: a $m \times n$ matrix $A_{m \times n}$ (e.g., a word-document matrix), can be decomposed to

$$U_{m \times m} \Sigma_{m \times n} V_{n \times n}^T ,$$

where U and V are orthogonal matrices, and Σ is a diagonal matrix.

Dimensionality reduction

Motivation: want a lower-dimensional, dense representation for efficiency

Recall **SVD**: a $m \times n$ matrix $A_{m\times n}$ (e.g., a word-document matrix), can be decomposed to

$$U_{m\times m}\Sigma_{m\times n}V^T_{n\times n} \ ,$$

where U and V are orthogonal matrices, and Σ is a diagonal matrix.

Interpretation:

$$AA^T = (U\Sigma V^T)(V\Sigma U^T) = U\Sigma^2 U^T \ .$$

- σ_i^2 are eigenvalues of AA^T
- Connection to PCA: If columns of A have zero mean (i.e. AA^T is the covariance matrix), then columns of U are principle components of the column space of A.

SVD for the word-document matrix

[board]

- Run truncated SVD of the word-document matrix $A_{m \times n}$
- Each row of $U_{m \times k}$ corresponds to a word vector of dimension k
- Each coordinate of the word vector corresponds to a cluster of documents (e.g., politics, music etc.)

Summary

Vector space models

1. Design the matrix, e.g. word $\times$ document, people $\times$ movie.
2. Reweight the raw counts, e.g. TFIDF, PMI.
3. Reduce dimensionality by (truncated) SVD.
4. Use word/person/etc. vectors in downstream tasks.

Key idea:

- Represent an object by its connection to other objects.
- For NLP, the word meaning can be represented by the context it occurs in.
- Infer latent features using co-occurence statistics

Table of Contents

Learning word embeddings

Goal: map each word to a vector in $\mathbb{R}^d$ such that *similar* words also have *similar* word vectors.

Learning word embeddings

Goal: map each word to a vector in $\mathbb{R}^d$ such that *similar* words also have *similar* word vectors.

Can we formalize this as a prediction problem?

- Needs to be self-supervised since our data is unlabeled.

Learning word embeddings

Goal: map each word to a vector in $\mathbb{R}^d$ such that *similar* words also have *similar* word vectors.

Can we formalize this as a prediction problem?

- Needs to be self-supervised since our data is unlabeled.

Intuition: words can be inferred from context

- Similar words occur in similar contexts
- Predict the context given a word f : word $\rightarrow$ context
- Words that tend to occur in similar contexts will have similar representation

The skip-gram model

Task: given a word, predict its neighboring words within a window

The quick brown fox jumps over the lazy dog

Assume conditional independence of the context words:

$$p(w_{i-k}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+k} \mid w_i) = \prod_{j=i-k, j \neq i}^{i+k} p(w_j \mid w_i)$$

How to model $p(w_j \mid w_i)$?

The skip-gram model

Task: given a word, predict its neighboring words within a window

The quick brown fox jumps over the lazy dog

Assume conditional independence of the context words:

$$p(w_{i-k}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+k} \mid w_i) = \prod_{j=i-k, j \neq i}^{i+k} p(w_j \mid w_i)$$

How to model $p(w_j \mid w_i)$? Multiclass classification

The skip-gram model

Use logistic regression to predict context from words

$$p(w_j \mid w_i) = \frac{\exp\left[\theta_{w_j} \cdot \phi(w_i)\right]}{\sum_{w \in \mathcal{V}} \exp\left[\theta_w \cdot \phi(w_i)\right]}$$
$$= \frac{\exp\left[\phi_{\text{ctx}}(w_j) \cdot \phi_{\text{wrd}}(w_i)\right]}{\sum_{w \in \mathcal{V}} \exp\left[\phi_{\text{ctx}}(w_j) \cdot \phi_{\text{wrd}}(w_i)\right]}$$

Classification $\rightarrow$ learning word vectors:

- $\phi(w_i)$: input features $\rightarrow$ context representation
- θ_{w_j}: weight vector for class w_j $\rightarrow$ word represenation of w_j

Implementation:

- Matrix form: $\phi \colon w \mapsto A_{d \times |\mathcal{V}|} \phi_{\text{one-hot}}(w)$, ϕ can be implemented as a dictionary
- Learn parameters by MLE and SGD (Is the objective convex?)
- ϕ_{wrd} is taken as the word embedding

Negative sampling (HW1 P2)

Challenge in MLE: computing the normalizer is expensive (try calculate the gradient)!

Key idea: solve a binary classification problem instead

Is the (word, context) pair real or fake?

positive examples +

w	c_{pos}
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

w	c_{neg}	w	c_{neg}
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

$$p_\theta(\text{real} \mid w, c) = \frac{1}{1 + e^{-\phi_{\text{ctx}}(c) \cdot \phi_{\text{wrd}}}}$$

The continuous bag-of-words model

Task: given the context, predict the word in the middle

The quick brown fox jumps over the lazy dog

Similary, we can use logistic regression for the prediction

$$p(w_i \mid w_{i-k}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+k})$$

How to represent the context (input)?

The continuous bag-of-words model

The context is a sequence of words.

$$c = w_{i-k}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+k}$$

$$p(w_i \mid c) = \frac{\exp\left[\theta_{w_i} \cdot \phi_{\text{BoW}}(c)\right]}{\sum_{w \in \mathcal{V}} \exp\left[\theta_w \cdot \phi_{\text{BoW}}(c)\right]}$$

$$= \frac{\exp\left[\phi_{\text{wrd}}(w_i) \cdot \sum_{w' \in c} \phi_{\text{ctx}}(w')\right]}{\sum_{w \in \mathcal{V}} \exp\left[\phi_{\text{wrd}}(w) \cdot \sum_{w' \in c} \phi_{\text{ctx}}(w')\right]}$$

- $\phi_{\text{BoW}}(c)$ sums over representations of each word in c
- Implementation is similar to the skip-gram model.

Semantic properties of word embeddings

Find similar words: top-k nearest neighbors using cosine similarity

- Size of window influences the type of similarity
- Shorter window produces syntactically similar words, e.g., Hogwarts and Sunnydale (fictional schools)
- Longer window produces topically related words, e.g., Hogwarts and Dumbledore (Harry Porter entities)

Semantic properties of word embeddings

Solve word analogy problems: a is to b as a' is to what?

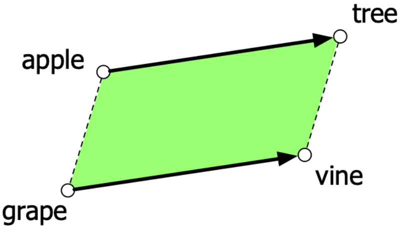


Figure: Parallelogram model (from J&H).

- man : woman :: king : queen
  $\phi_{\text{wrd}}(\text{man}) - \phi_{\text{wrd}}(\text{woman}) \approx \phi_{\text{wrd}}(\text{king}) - \phi_{\text{wrd}}(\text{queen})$
- man : woman :: king : ?
  $\arg\max_{w \in \mathcal{V}} \text{sim}(-\phi_{\text{wrd}}(\text{man}) + \phi_{\text{wrd}}(\text{woman}) + \phi_{\text{wrd}}(\text{king}), w)$
- Caveat: must exclude the three input words
- Does not work for general relations

Comparison

vector space models	word embeddings
matrix factorization	prediction problem
fast to train	slow (with large corpus) but more flexible
interpretable components	hard to interprete but has intriguing properties

- Both uses the distributional hypothesis.
- Both generalize beyond text: using co-occurence between any types of objects
  - Learn product embeddings from customer orders
  - Learn region embeddings from images

Summary

Key idea: formalize word representation learning as a self-supervised prediction problem

Prediction problems:

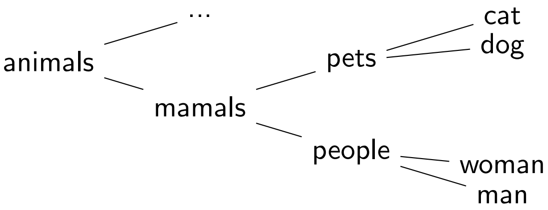
- Skip-gram: Predict context from words
- CBOW: Predict word from context
- Other possibilities:
  - Predict $\log \hat{p}(\text{word} \mid \text{context})$, e.g. GloVe
  - Contextual word embeddings (later)

Table of Contents

Brown clustering

Developed at IBM by Peter Brown et al. in early 90s.

Idea: hierarchically clustering words (initially used for language modeling)



- Use the path encoding to root as the word representation
- In practice, trees are constructed by hierarchical clustering and internal nodes are latent

Example clusters

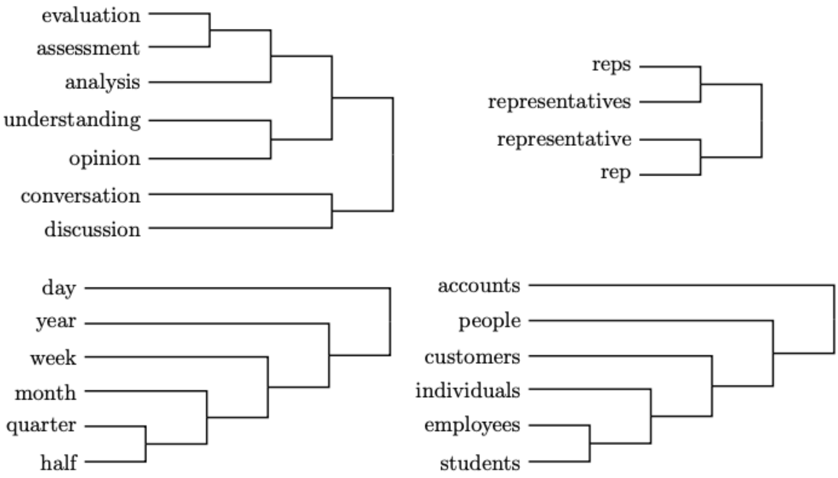


Figure: From Eisenstein

Evaluate word vectors

Intrinsic evaluation

- Evaluate on the proxy task (related to the learning objective)
- Word similarity/analogy datasets (e.g., WordSim-353, SimLex-999)

Extrinsic evaluation

- Evaluate on the real/downstream task we care about
- Use word vectors as features in NER, parsing etc.

Table of Contents

Feature learning

Linear predictor with handcrafted features: $f(x) = w \cdot \phi(x)$.

Can we learn intermediate features?

Example:

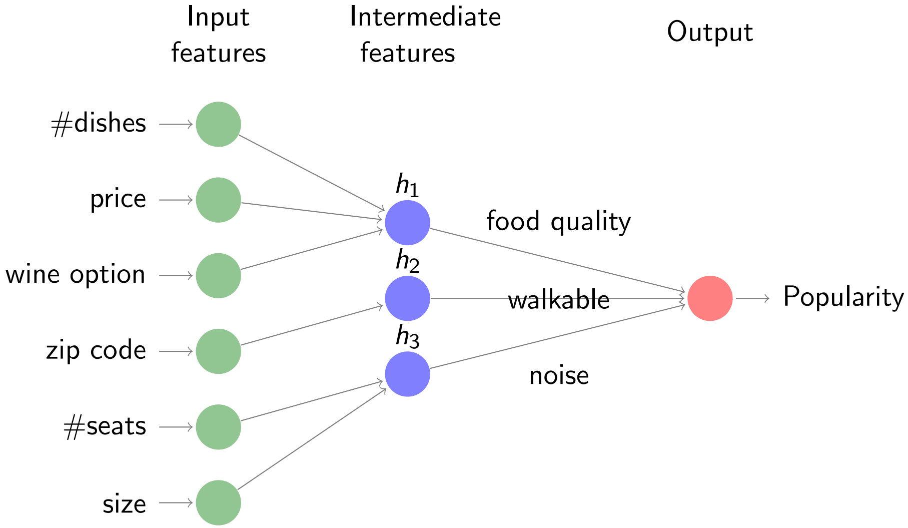
- Predict popularity of restaurants.
- Raw input: #dishes, price, wine option, zip code, #seats, size
- Decompose into subproblems:

$$h_1([\#\text{dishes, price, wine option}]) = \text{food quality}$$

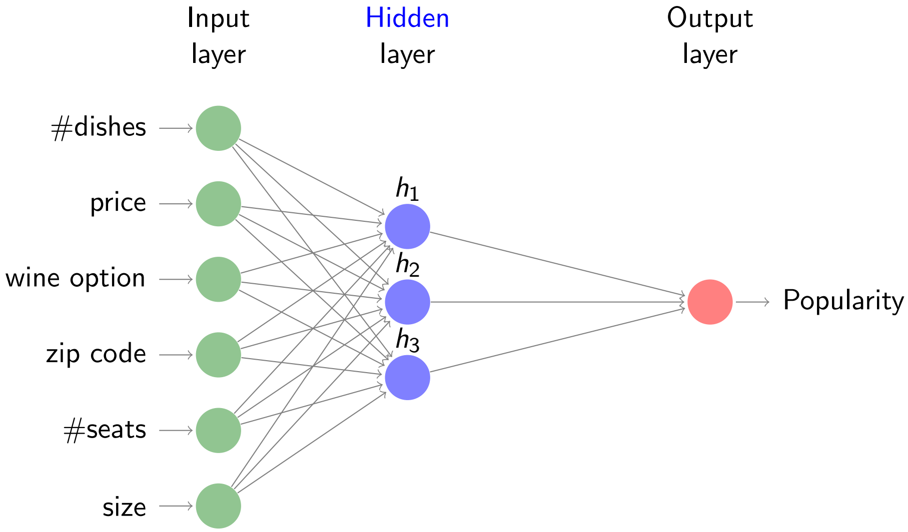
$$h_2([\text{zip code}]) = \text{walkable}$$

$$h_3([\#\text{seats, size}]) = \text{nosie}$$

Predefined subproblems



Learning intermediate features



Neural networks

Key idea: automatically learn the intermediate features.

Feature engineering: Manually specify $\phi(x)$ based on domain knowledge and learn the weights:

$$f(x) = w^T \phi(x).$$

Feature learning: Automatically learn both the features (K hidden units) and the weights:

$$h(x) = [h_1(x), \ldots, h_K(x)], \quad f(x) = w^T h(x)$$

Activation function

- How should we parametrize h_i's? Can it be linear?

Activation function

- How should we parametrize h_i's? Can it be linear?

$$h_i(x) = \sigma(v_i^T x). \tag{3}$$

- σ is the *nonlinear* **activation function**.

Activation function

- How should we parametrize h_i's? Can it be linear?

$$h_i(x) = \sigma(v_i^T x). \tag{3}$$

- σ is the *nonlinear* **activation function**.
- What might be some activation functions we want to use?

Activation function

- How should we parametrize h_i's? Can it be linear?

$$h_i(x) = \sigma(v_i^T x). \tag{3}$$

- σ is the *nonlinear* **activation function**.
- What might be some activation functions we want to use?
  - sign function? Non-differentiable.

Activation function

- How should we parametrize h_i's? Can it be linear?

$$h_i(x) = \sigma(v_i^T x). \tag{3}$$

- σ is the *nonlinear* **activation function**.
- What might be some activation functions we want to use?
  - sign function? Non-differentiable.
  - *Differentiable* approximations: sigmoid functions.
    - E.g., logistic function, hyperbolic tangent function.

Activation function

- How should we parametrize h_i's? Can it be linear?

$$h_i(x) = \sigma(v_i^T x). \tag{3}$$

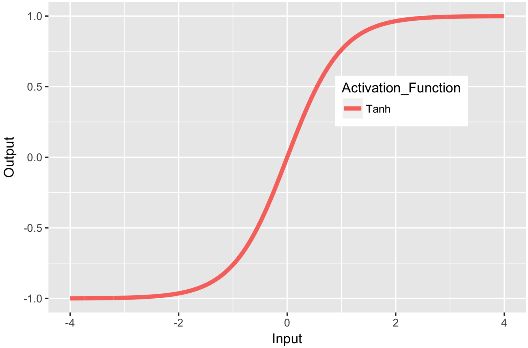
- σ is the *nonlinear* **activation function**.
- What might be some activation functions we want to use?
  - sign function? Non-differentiable.
  - *Differentiable* approximations: sigmoid functions.
    - E.g., logistic function, hyperbolic tangent function.
- Two-layer neural network (one hidden layer and one output layer) with K hidden units:

$$f(x) = \sum_{k=1}^{K} w_k h_k(x) = \sum_{k=1}^{K} w_k \sigma({v_k}^T x) \tag{4}$$

Activation Functions

- The **hyperbolic tangent** is a common activation function:

$$\sigma(x) = \tanh(x).$$

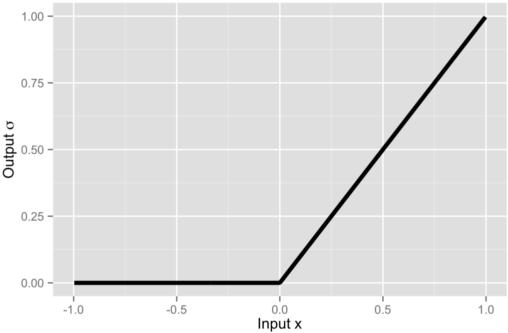


Activation Functions

- More recently, the **rectified linear** (**ReLU**) function has been very popular:

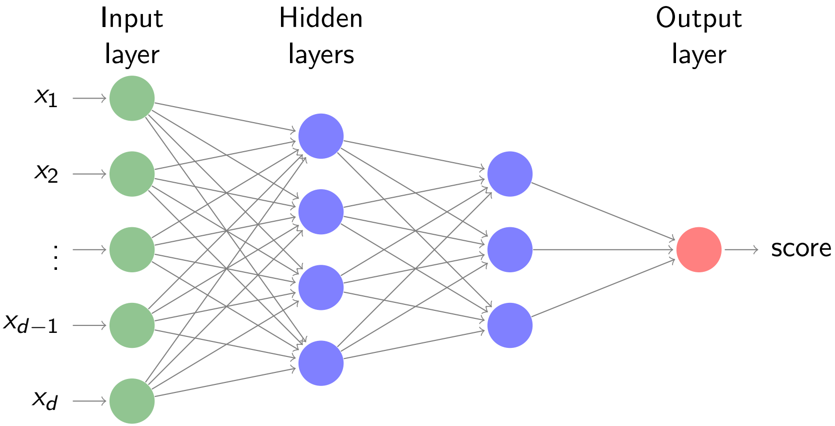
$$\sigma(x) = \max(0, x).$$

- Much faster to calculate, and to calculate its derivatives.
- Also often seems to work better.



Multilayer perceptron / Feed-forward neural networks

- Wider: more hidden units.
- Deeper: more hidden layers.



Multilayer Perceptron: Standard Recipe

- Each subsequent hidden layer takes the output $o \in \mathbb{R}^m$ of previous layer and produces

$$h^{(j)}(o^{(j-1)}) = \sigma\left(W^{(j)} o^{(j-1)} + b^{(j)}\right), \text{ for } j = 2, \ldots, L$$

where $W^{(j)} \in \mathbb{R}^{m \times m}$, $b^{(j)} \in \mathbb{R}^m$.

Multilayer Perceptron: Standard Recipe

- Each subsequent hidden layer takes the output $o \in \mathbb{R}^m$ of previous layer and produces

$$h^{(j)}(o^{(j-1)}) = \sigma\left(W^{(j)} o^{(j-1)} + b^{(j)}\right), \text{ for } j = 2, \ldots, L$$

where $W^{(j)} \in \mathbb{R}^{m \times m}$, $b^{(j)} \in \mathbb{R}^m$.

- Last layer is an *affine* mapping (no activation function):

$$a(o^{(L)}) = W^{(L+1)} o^{(L)} + b^{(L+1)},$$

where $W^{(L+1)} \in \mathbb{R}^{k \times m}$ and $b^{(L+1)} \in \mathbb{R}^k$.

Multilayer Perceptron: Standard Recipe

- Each subsequent hidden layer takes the output $o \in \mathbb{R}^m$ of previous layer and produces
$$h^{(j)}(o^{(j-1)}) = \sigma\left(W^{(j)}o^{(j-1)} + b^{(j)}\right), \text{ for } j = 2, \ldots, L$$
where $W^{(j)} \in \mathbb{R}^{m \times m}$, $b^{(j)} \in \mathbb{R}^m$.
- Last layer is an *affine* mapping (no activation function):
$$a(o^{(L)}) = W^{(L+1)}o^{(L)} + b^{(L+1)},$$
where $W^{(L+1)} \in \mathbb{R}^{k \times m}$ and $b^{(L+1)} \in \mathbb{R}^k$.
- The full neural network function is given by the *composition* of layers:
$$f(x) = \left(a \circ h^{(L)} \circ \cdots \circ h^{(1)}\right)(x) \tag{5}$$

Multilayer Perceptron: Standard Recipe

- Each subsequent hidden layer takes the output $o \in \mathbb{R}^m$ of previous layer and produces

$$h^{(j)}(o^{(j-1)}) = \sigma\left(W^{(j)}o^{(j-1)} + b^{(j)}\right), \text{ for } j = 2, \ldots, L$$

  where $W^{(j)} \in \mathbb{R}^{m \times m}$, $b^{(j)} \in \mathbb{R}^m$.

- Last layer is an *affine* mapping (no activation function):

$$a(o^{(L)}) = W^{(L+1)}o^{(L)} + b^{(L+1)},$$

  where $W^{(L+1)} \in \mathbb{R}^{k \times m}$ and $b^{(L+1)} \in \mathbb{R}^k$.

- The full neural network function is given by the *composition* of layers:

$$f(x) = \left(a \circ h^{(L)} \circ \cdots \circ h^{(1)}\right)(x) \qquad (5)$$

- Last layer typically gives us a score. How to do classification?