Topics in Natural Language Processing

Nils Kujath

Linear Algebra Basics

- Let $n \in \mathbb{N}$. The set of all ordered n-tuples $(c_1, c_2, \ldots, c_n)$, where each component $c_i \in \mathbb{R}$, forms the **vector space** $\mathbb{R}^n$ over (the field) $\mathbb{R}$. We call each of these n-tuples in $\mathbb{R}^n$ a **vector** in $\mathbb{R}^n$. We will use bold lowercase letters to denote vectors, i.e., $\mathbf{v} \in \mathbb{R}^n$. We use the term **scalar** to refer to an **element** of the field $\mathbb{R}$—that is, one of the **components** of a vector in $\mathbb{R}^n$.
- An $m \times n$ **matrix** is a rectangular array of scalars arranged in m rows and n columns, with each entry taken from $\mathbb{R}$. We use uppercase bold letters to denote matrices, e.g., $\mathbf{A} \in \mathbb{R}^{m \times n}$. The scalar entry in row i and column j is denoted by $A_{[i,j]}$. The i-th row and the j-th column—each of which can be interpreted as a vector—are denoted by $\mathbf{A}_{[i,*]}$ and $\mathbf{A}_{[*,j]}$, respectively.
- A **tensor** is a multidimensional array of scalars that generalizes the concepts of scalars (order 0), vectors (order 1), and matrices (order 2). A tensor of order k has k indices and can be represented as an element of $\mathbb{R}^{d_1 \times d_2 \times \cdots \times d_k}$, where each $d_i \in \mathbb{N}$ specifies the size along the i-th mode. We typically denote tensors by boldface uppercase calligraphic letters, e.g., $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{d_1 \times \cdots \times d_k}$.

Linear Algebra Basics (cont'd)

- Let $\mathbf{v} = (1, 2, 3)$ and $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$.
- We can use Python's `numpy` package to represent tensors using the `ndarray` class, which provides a general-purpose implementation of *n*-dimensional arrays (note that Python indexes from 0):

```
import numpy as np

v = np.array([1,2,3])
A = np.array([[1,2,3],[4,5,6]])

print(f"Type of v: {type(v)}, Type of A: {type(A)}")
print(f"Shape of v: {v.shape}, Shape of A: {A.shape}")
print(f"v[0]: {v[0]}   A[1]: {A[1]} A[1,2]: {A[1][2]}")
```

```
Type of v: <class 'numpy.ndarray'>, Type of A: <class 'numpy.ndarray'>
Shape of v: (3,), Shape of A: (2, 3)
v[0]: 1   A[1]: [4 5 6]   A[1,2]: 6
```

Linear Algebra Basics (cont'd)

- Let $n \in \mathbb{N}$. Let $\mathbf{u} = (u_1, \ldots, u_n)$ and $\mathbf{v} = (v_1, \ldots, v_n)$ be two vectors in $\mathbb{R}^n$:
  - Their **sum** is defined componentwise: $\mathbf{u} + \mathbf{v} = (u_1 + v_1, \ldots, u_n + v_n) \in \mathbb{R}^n$, e.g.: $(1, 2, 3) + (2, 3, 4) = (3, 5, 7)$;
  - Similarly, their **difference** is defined componentwise: $\mathbf{u} - \mathbf{v} = (u_1 - v_1, \ldots, u_n - v_n) \in \mathbb{R}^n$, e.g.: $(1, 2, 3) - (4, 5, 6) = (-3, -3, -3)$.
- Let $n \in \mathbb{N}$ and $k \in \mathbb{R}$. Let $\mathbf{u} = (u_1, \ldots, u_n)$. $\mathbf{u}$ can be multiplied with a scalar k:
  - The **scalar multiplication** of $\mathbf{u}$ with k is defined componentwise: $k\mathbf{u} = (ku_1, \ldots, ku_n) \in \mathbb{R}^n$, e.g.: $2 \cdot (1, 2, 3) = (2, 4, 6)$.
- Let $n \in \mathbb{N}$ and $a \in \mathbb{R}$. Let $\mathbf{u} = (u_1, \ldots, u_n)$ and $\mathbf{v} = (v_1, \ldots, v_n)$ be two vectors in $\mathbb{R}^n$:
  - The two vectors $\mathbf{u}$ and $\mathbf{v}$ are said to be **colinear** iff $\exists a \in \mathbb{R}$ if there exists a scalar $a \neq 0$ such that we can multiply $\mathbf{v}$ with a and get $\mathbf{u}$ (i.e., $\mathbf{u} = a\mathbf{v}$), e.g.: $(1, 2, 3) = 0.5 \cdot (2, 4, 6)$
  - If $a > 0$, both vectors point in the same direction; If $a < 0$, both vectors point in opposite directions.