Rappel (très) informel d'algèbre linéaire

Pascal Vincent

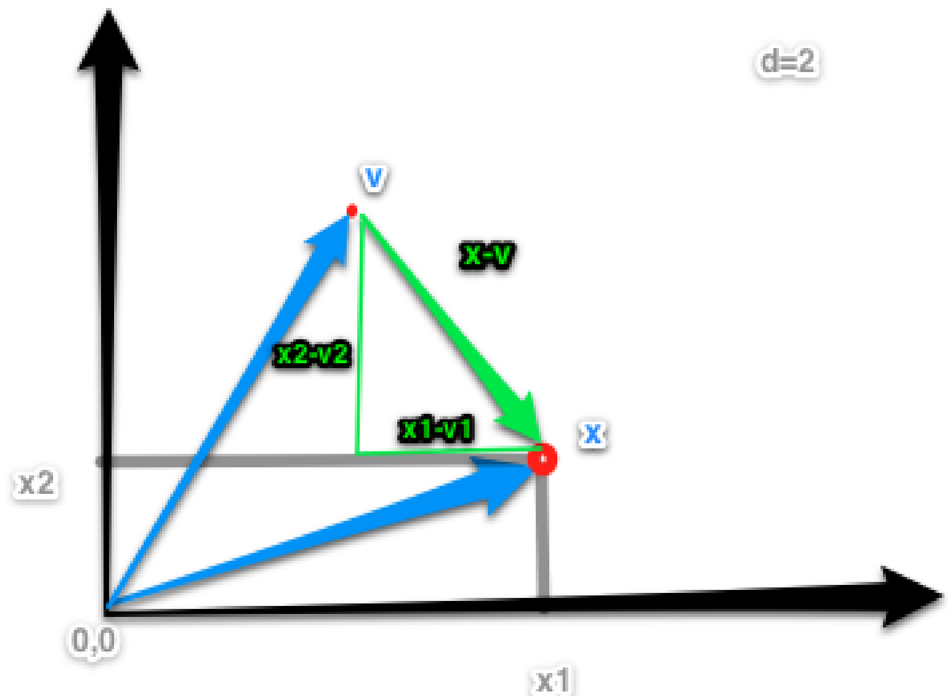
September 7, 2016

1 Scalaire?

un nombre réel

$a \in \mathbb{R}$

2 Vecteur x de dimension d



Représentations d'un vecteur:

- une flèche
- un point dans un espace de dimension d
- une liste de d valeurs (nommés coposantes et éléments du vecteur: les d coordonnées du point)
- $x \in \mathbb{R}^d$

- $x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix}$

- Par convention les vecteurs sont des **“vecteurs colonne”**

- **Opération de transposition:** intervertit ligne et colonne
  $x^T = (x_1, x_2, \ldots, x_d)$

- **En Java:**
  `double[] x = new double[d];`
  $x_1 \equiv$ `x[0]`

- **En Python/numpy:**
  `x = numpy.zeros(d)`

3 Matrices

2 dimensions: nombre de lignes n, nombre de colonnes d

- Soit A une matrice $n \times d$

- $A \in \mathbb{R}^{n \times d}$

$$A = \begin{pmatrix} A_{11} & \ldots & A_{1d} \\ \vdots & \ddots & \vdots \\ A_{n1} & \cdots & A_{nd} \end{pmatrix}$$

  En Python/numpy:
  `x = numpy.zeros((n,d))`

- A_{ij} représente l’élément à la ligne i, colonne j
  En Python/numpy:
  `A[i,j]`
  Attention, numérotation des indices à partir de 0 (alors qu’en math c’est à partir de 1):
  $A_{11} \equiv$ `A[0,0]`

- Colonne j de la matrice:
  $A_{:,j} = \begin{pmatrix} A_{1j} \\ \vdots \\ A_{nj} \end{pmatrix}$
  En python/numpy: `A[:,j]`

- Ligne i de la matrice (vue comme un vecteur colonne):
  $A_{i:} = A_i = \begin{pmatrix} A_{i1} \\ \vdots \\ A_{id} \end{pmatrix}$
  En python/numpy: A[i,:] ou bien simplement A[i]

ATTENTION: Remarque de notation, en apprentissage machine.

- L'ensemble de donnée est souvent stocké sous la forme d'une ou plusieurs matrices et/ou vecteurs.
- Par ex, si on a n exemples d'entrée en dimension d on peut les avoir dans une matrice X qui est $n \times d$. Associée à un vecteur de cibles y de n éléments.
- D'après la notation plus haut, le $i^{ème}$ exemple d'entrée pourrait être noté: $X_{i,:}$ ou X_i
- Parfois on va plutôt noter notre ensemble de donnée

  $$D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

  Dans ce cas x_i est un vecteur de dimension d. Et la$j^{ème}$ composante de ce vecteur sera logiquement x_{ij}
- **MAIS PARFOIS** x est aussi utilisé pour représenter **un** vecteur exemple, dans ce cas x_i représente plutôt la ième composante (scalaire) de l'exemple x. Il faut donc porter attention au contexte!
- Pour éviter cette confusion, vous verrez parfois l'ensemble de donnée noté:

  $$D = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$$

  Dans ce cas la $j^{ème}$ composante du $i^{ème}$ exemple sera logiquement notée $x_j^{(i)}$.

4 Tenseurs

Les tenseurs sont des généralisation des vecteurs et matrices.

- Un tenseur d'ordre K correspond, en informatique à un tableau à K dimensions.
- Un scalaire est un tenseur d'ordre 0
- Un vecteur est un tenseur d'ordre 1
- Une matrice est un tenseir d'ordre 2 (lignes, colonnes)

- On peut s'imaginer un tenseur d'ordre 3 comme ayant linges,colonnes,profondeur

En python/numpy, le type qui représente desvecteurs/matrices/tenseurs se nomme "ndarray".

Pour ex. créer un tenseur d'ordre 3 de n lignes, d colonnes, c cases de profond, rempli de zéros, on peut écrire:

T=numpy.zeros((n,d,c))

5 Opérations de base sur les vecteurs

5.1 Opérations terme à terme

$\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$

$$x - v = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} - \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_d \end{pmatrix} = \begin{pmatrix} x_1 - v_1 \\ x_2 - v_2 \\ \vdots \\ x_d - v_d \end{pmatrix}$$

$$x + v = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} + \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_d \end{pmatrix} = \begin{pmatrix} x_1 + v_1 \\ x_2 + v_2 \\ \vdots \\ x_d + v_d \end{pmatrix}$$

$$x \odot v = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \odot \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_d \end{pmatrix} = \begin{pmatrix} x_1 v_1 \\ x_2 v_2 \\ \vdots \\ x_d v_d \end{pmatrix}$$

5.2 Changement d'échelle (produit scalaire,vecteur)

$\mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$

$$\lambda x = \lambda \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} = \begin{pmatrix} \lambda x_1 \\ \lambda x_2 \\ \vdots \\ \lambda x_d \end{pmatrix}$$

5.3 Produit scalaire

$\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$

Produit scalaire (dot product) entre $x \in \mathbb{R}^d$ et $w \in \mathbb{R}^d$:

$$\begin{aligned}
\langle w, x\rangle &= w^T x \\
&= (w_1, w_2, \ldots, w_d) \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \\
&= w_1 x_1 + w_2 x_2 + \ldots + w_d x_d \\
&= \sum_{k=1}^{d} w_k x_k
\end{aligned}$$

Python numpy: numpy.dot(w,x)
Complexité algorithmique: O(d)

```
def dot(w,x):
    d = len(w)
    resultat = 0
    i = 0
    while i<d:
        resultat = resultat + w[i]*x[i]
        i = i+1

    return resultat
```

5.4 Produit externe

Ex: $\mathbb{R}^{d'} \times \mathbb{R}^d \to \mathbb{R}^{d' \times d}$

$$\begin{aligned}
wx^T &= \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_{d'} \end{pmatrix} (x_1, x_2, \ldots, x_d) \\
&= \begin{pmatrix} w_1 x_1 & w_1 x_2 & \ldots & w_1 x_d \\ w_2 x_1 & w_2 x_2 & \ldots & w_2 x_d \\ \vdots & \vdots & \ddots & \vdots \\ w_{d'} x_1 & w_{d'} x_2 & \cdots & w_{d'} x_d \end{pmatrix}
\end{aligned}$$

5.5 Norme Euclidienne ou L_2

Correspond à la “longueur” du vecteur

$$\|x\| = \|x\|_2 = \sqrt{\langle x, x\rangle} = \sqrt{x^T x} = \sqrt{x_1^2 + x_2^2 + \ldots + x_d^2} = \sqrt{\sum_{k=1}^{d} x_k^2}$$

Norme au carré:

$$\|x\|^2 = \|x\|_2^2 = \langle x, x\rangle = x^T x = x_1^2 + x_2^2 + \ldots + x_d^2 = \sum_{k=1}^{d} x_k^2$$

5.6 Norme L_p

$$\|x\|_p = \sqrt[p]{|x|_1^p + |x|_2^p + \ldots + |x|_d^p} = \left(\sum_{k=1}^{d} |x|_k^p\right)^{\frac{1}{p}}$$

5.7 Distance Euclidienne ou L2

$$\begin{aligned} d_{Euclid}(x, v) &= d_2(x, v) = \|x - v\|_2 \\ &= \sqrt{(x_1 - v_1)^2 + \ldots + (x_d - v_d)^2} \\ &= \sqrt{\sum_{i=1}^{d} (x_i - v_i)^2} \end{aligned}$$

5.8 Distance L_p

$$\begin{aligned} d_p(x, v) &= \|x - v\|_p \\ &= \left(\sum_{i=1}^{d} |x_i - v_i|^p\right)^{1/p} \end{aligned}$$

6 Opérations matrice vecteur

6.1 Produit matrice vecteur

$\mathbb{R}^{n\times d} \times \mathbb{R}^d \to \mathbb{R}^n$

$$\begin{aligned} Ax &= \begin{pmatrix} A_{11} & \ldots & A_{1d} \\ \vdots & \ddots & \vdots \\ A_{n1} & \cdots & A_{nd} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \\ &= \begin{pmatrix} \langle A_{1:}, x\rangle \\ \langle A_{2:}, x\rangle \\ \vdots \\ \langle A_{n:}, x\rangle \end{pmatrix} \\ &= x_1 A_{:,1} + x_2 A_{:,2} + \ldots + x_d A_{:,d} \end{aligned}$$

Donne un vecteur
Complexité algorithmique: $O(nd)$

6.2 Forme quadratique

$$\begin{aligned} x^T W x &= (x_1, \ldots, x_d) \begin{pmatrix} A_{11} & \cdots & A_{1d} \\ \vdots & \ddots & \vdots \\ A_{d1} & \cdots & A_{dd} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \\ &= \sum_{i=1}^{d} \sum_{j=1}^{d} A_{ij} x_i x_j \end{aligned}$$

Donne un scalaire
Complexité algorithmique: $O(d^2)$