Python Basics with Numpy

- **imgae2vector**

    -
      ```
      " " "
      Argument:
          image -- a numpy array of shape (length, height, depth)

          Returns:
          v -- a vector of shape (length*height*depth, 1)
       " " "
      v = image.reshape(image.shape[0]*image.shape[1]*image.shape[2], 1)
      ```

    -
      ```
      " " "
      Argument:
          image -- a numpy array of shape (examples, length, height, depth)

          Returns:
          v -- a vector of shape (examples*length*height*depth, examples)
       " " "
      v = image.reshape(image.shape[0], -1).T
      ```
- **normalization**
    - `np.linalg.nrom(x, ord, axis, keepdim)`
    - `x=x/x_norm`
- **Softmax**:
    - for $x \in \mathbb{R}^{1\times n}$, $softmax(x) = softmax(\begin{bmatrix} x_1 & x_2 & \ldots & x_n \end{bmatrix}) = \begin{bmatrix} \frac{e^{x_1}}{\sum_j e^{x_j}} & \frac{e^{x_2}}{\sum_j e^{x_j}} & \cdots & \frac{e^{x_n}}{\sum_j e^{x_j}} \end{bmatrix}$
    - for a matrix $x \in \mathbb{R}^{m\times n}$, x_{ij} maps to the element in the i^{th} row and j^{th} column of x, thus we have:

$$softmax(x) = softmax\begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & x_{m3} & \dots & x_{mn} \end{bmatrix} = \begin{bmatrix} \frac{e^{x_{11}}}{\sum_j e^{x_{1j}}} & \frac{e^{x_{12}}}{\sum_j e^{x_{1j}}} & \frac{e^{x_{13}}}{\sum_j e^{x_{1j}}} & \dots & \frac{e^{x_{1n}}}{\sum_j e^{x_{1j}}} \\ \frac{e^{x_{21}}}{\sum_j e^{x_{2j}}} & \frac{e^{x_{22}}}{\sum_j e^{x_{2j}}} & \frac{e^{x_{23}}}{\sum_j e^{x_{2j}}} & \dots & \frac{e^{x_{2n}}}{\sum_j e^{x_{2j}}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{e^{x_{m1}}}{\sum_j e^{x_{mj}}} & \frac{e^{x_{m2}}}{\sum_j e^{x_{mj}}} & \frac{e^{x_{m3}}}{\sum_j e^{x_{mj}}} & \dots & \frac{e^{x_{mn}}}{\sum_j e^{x_{mj}}} \end{bmatrix} = \begin{pmatrix} softmax\text{(first row of x)} \\ softmax\text{(second row of x)} \\ ... \\ softmax\text{(last row of x)} \end{pmatrix} \tag{1}$$

    -
      ```
      def softmax(x):

          x_exp = np.exp(x)

          x_sum = np.sum(x_exp, axis=1,keepdims=True)

          s = x_exp/x_sum

          return s
      ```