Language-Agnostic Tensor Processing Exercises

[redacted]

[redacted]

[redacted]

Exercises aiming at improving your "vectorized"-way-of-coding.

Here we'll be dealing with N-dimensional arrays or tensors, in particular:

- $N = 1$ - vectors,
- $N = 2$ - matrices.

These might be represented as NumPy array objects, Theano or TensorFlow tensors, R atomic vectors/matrix/arrays, Matlab vectors/matrices, vector/matrix classes in C++ Eigen or Armadillo libraries etc.

Most of the functions can be implemented based on built-in, "vectorized" operations (arithmetic operators, aggregation functions, vector subsetting etc.) – no explicit `for`-loops required. If your library does not provide you with such operations, pick a different one.

> Note that many machine learning algorithms actually take numeric matrix inputs or outputs, as real-world objects can be represented as sequences of feature vectors (if there are given as data frames, a conversion is done). Therefore, we will often assume that a matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ represents n points in $\mathbb{R}^d$. In such a case $\mathbf{x}_{i,\cdot}$ will refer to the i-th row (point, observation) and $\mathbf{x}_{\cdot,j}$ - the j-th column (variable, coordinate).

A *missing value* in a numeric vector is represented as `NA_real_` in R or `NaN` in all other cases.

If not stated explicitly otherwise, by the term "distance" we mean the Euclidean metric, but you can play with other metrics too.

1. Let `x` be a numeric vector:

   - Print all values in $[-2, -1] \cup [1, 2]$.
   - Print the number and the proportion of nonnegative elements in `x`.
   - Compute the arithmetic mean of absolute values.
   - Determine elements in `x` which are the least and the most distant from 0.
   - Determine 3 elements in `x` which are the most distant from the arithmetic mean of `x`.
   - Create a vector `x2`, which is a version of `x` with all outliers removed, i.e., all observations x_i such that $x_i \notin [Q_1 - 1.5IQR, Q_3 + 1.5IQR]$, where $IQR = Q_3 - Q_1$ denotes the interquartile range and Q_1 and Q_3 denote the 1st and 3rd sample quartiles, respectively.
   - Create a vector `y` such that `y[i]` is equal to `"nonnegative"` if the corresponding `x[i]>=0` and `"negative"` otherwise.
   - Create a vector `y` such that `y[i]` is equal to `"small"` if the corresponding `x[i]<-1`, `"large"` if `x[i]>1` and `"medium"` otherwise.

   Generate `x` like:

   ```
   set.seed(123)
   x <- round(rnorm(20, 0, 1), 2)
   ```

   or:

   ```
   import numpy as np
   np.random.seed(6)
   ```

```
x = np.round(np.random.normal(size=20), 2)
```

2. Write a function which standardizes the values in a given numeric vector, i.e., rescales its elements so that the resulting vector is of mean and standard deviation of 1. Note: this is also called "Z-score computing".

3. Write a function which clamps all values in a given vector to the unit interval, i.e., set all values less than 0 to be equal to 0 and all values greater than 1 to be equal to 1.

4. Compute the inner product of two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, i.e., $\mathbf{x}^T\mathbf{y} = \sum_{i=1}^n x_i y_i$.

5. Compute the root-mean-squared-error between two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\mathrm{RMSE}(\mathbf{x}, \mathbf{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^n (y_i - x_i)^2}$.

6. Compute the mean-absolute-error between two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\mathrm{MAE}(\mathbf{x}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^n |y_i - x_i|$.

7. Compute the median-absolute-error between two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\mathrm{MedAE}(\mathbf{x}, \mathbf{y}) = \mathrm{Median}\left(|y_1 - x_1|, \dots, |y_n - x_n|\right)$.

8. Compute the (normalized) Hamming distance between two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$, $\mathrm{Ham}(\mathbf{x}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^n \mathbb{I}(x_i \neq y_i)$, where the indicator function $\mathbb{I}$ yields 1 if a given logical condition is met and 0 otherwise.

9. Compute the cross-entropy loss between two vectors $\mathbf{y} \in \{0,1\}^n$ and $\hat{\mathbf{y}} \in (0,1)^n$, i.e., $\mathrm{CEnt}(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{n}\left(\sum_{i=1}^n y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right)$.

10. Let `x` and `y` be two vectors of the same length, n. Compute the Pearson linear correlation coefficient, given by:

$$r(\mathbf{x}, \mathbf{y}) = \frac{1}{n-1}\sum_{i=1}^n \frac{x_i - \bar{x}}{s_x}\frac{y_i - \bar{y}}{s_y}.$$

11. Let `x` and `y` be two vectors of the same length, n. Compute the Spearman rank correlation coefficient, given by:

$$\varrho(\mathbf{x}, \mathbf{y}) = 1 - \frac{6\sum_{i=1}^n d_i^2}{n(n^2-1)},$$

    where $d_i = R(\mathbf{x})_i - R(\mathbf{y})_i$, $i = 1, \dots, n$, and $R(\mathbf{x})_i$ denotes the rank of x_i in `x`.

12. Let `x` and `y` be two vectors of the same length, n. Compute the Kendall rank correlation coefficient, given by:

$$\tau(\mathbf{x}, \mathbf{y}) = \frac{2c}{n(n-1)/2} - 1,$$

    where c denotes the number of concordant pairs, i.e., such that for $1 \le i < j \le n$ it either holds $x_i > x_j$ and $y_i > y_j$, or $x_i < x_j$ and $y_i < y_j$. Assume all values in `x` and `y` are unique.

13. Write a function that determines the mode, i.e., the most frequently occurring value in a given vector. If the mode is not-unique, return a randomly chosen one. Hint: Find a built-in function that counts the number of occurrences of each unique value in `t`.

14. Implement `lead(x, n)` and `lag(x, n)`. The former function gets rid of the first n observations in `x` and adds `n` missing values at the end of the resulting vector, e.g., `lead([1, 2, 3, 4, 5], 2) == [3, 4, 5, NaN, NaN]`, i.e., the observations are left-shifted. On the other hand, `lag([1, 2, 3, 4, 5], 2) == [NaN, NaN, 1, 2, 3]`, i.e., we get a right-shift.

15. Implement `cumall()` and `cumany()` - cumulative versions of `all()` and `any()`, e.g., `cumall([True, True, True, False, True, False]) == [True, True, True, False, False, False]` and `cumany([False, False, True, False, True]) == [False, False, True, True, True]`.

16. Write a function to compute the k-moving average of a given vector `x` of length n, where $k = 2l + 1$ for some l. Return a vector `t` of length n with the l first and l last observations set to `NaN`. Each other t_i should be equal to the arithmetic mean of $x_{i-l}, \dots, x_i, \dots, x_{i+l}$.

17. Write a function that takes as arguments: (a) an integer n, (b) a numeric vector `x` of length k and no duplicated elements, (c) a vector of probabilities `p` of length k; verify that $p_i \geq 0$ for all i and $\sum_{i=1}^{k} p_i \simeq 1$. Based on a random number generator for the uniform distribution on the unit interval, generate n independent random variates from a distribution of a random variable X such that $\Pr(X = x_i) = p_i$ for $i = 1, \dots, k$. Hint: to obtain a single value, (a) generate $u \in [0, 1]$, (b) find $m \in \{1, \dots, k\}$ such that $u \in (\sum_{j=1}^{m-1} p_j, \sum_{j=1}^{m} p_j]$, (c) the result is then x_m.

18. Write a function that takes as arguments: (a) an increasingly sorted vector `x` of length n, (b) any vector `y` of length n, (c) a vector `z` of length k and elements in $[x_1, x_n)$. Let f be the piecewise linear spline that interpolates the points $(x_1, y_1), \dots, (x_n, y_n)$. Return a vector `w` of length k such that $w_i = f(z_i)$.

19. Write a function that estimates the value of π. Generate m (given) random points from the square $[-1, 1]^2$. Let k denote the number of points that lie in the circle of radius 1 centered at $(0, 0)$. The expected value $4k/m$ (the area of the square $\times$ estimated proportion of the points in the circle) is exactly π.

20. Let `t` be vector of n integers in $\{1, \dots, k\}$ (or $\{0, \dots, k-1\}$ if your language relies on 0-based array indexing). Write a function to one-hot-encode each t_i. Return a 0-1 matrix R of size $n \times k$ such that $r_{i,j} = 1$ if and only if $t_i = j$. By the way, such a representation is useful when solving, e.g., a multiclass classification problem by means of k binary classifiers. For example, if `t=[1, 2, 3, 2]` and $k = 4$, then:

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

21. Given an $n \times k$ matrix with elements in $\mathbb{R}$, apply the softmax function to each row, i.e., $x_{i,j} \mapsto \frac{\exp(x_{i,j})}{\sum_{l=1}^{k} \exp(x_{i,l})}$. Then one-hot decode the values in each row, i.e., find the column number with the value most close to 1. Return a vector of size n.

22. Let `t` be vector of n integers in $\{1, \dots, k\}$ (or $\{0, \dots, k-1\}$). Compute a 0-1 matrix R of size $n \times k$ such that $r_{i,j} = 1$ if and only if $j \geq t_i$. For example, if `t=[1, 2, 3, 2]` and $k = 4$, then:

$$R = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}.$$

23. For a given $\mathbf{X} \in \mathbb{R}^{n \times d}$, determine the bounding hyperrectangle of the n points. Return a matrix $\mathbf{B} \in \mathbb{R}^{2 \times d}$ with $b_{1,j} = \min_i x_{i,j}$ and $b_{2,j} = \max_i x_{i,j}$.

24. Write a function which standardizes the values in each column of a given matrix (separately).

25. Given a matrix with n rows and m columns (e.g., the first 4 rows from the `iris` dataset), compute the correlation matrix, i.e., an $m \times m$ matrix $\mathbf{C}$ with $c_{i,j}$ denoting the Pearson coefficient for the i-th and the j-th column.

26. Assume that an $n \times d$ matrix $\mathbf{X}$ represents n points in $\mathbb{R}^d$. Write a function that determines the pairwise distances between all the points in $\mathbf{X}$ and a given $\mathbf{y} \in \mathbb{R}^d$. Return a vector $\mathbf{d} \in \mathbb{R}^n$ with $d_i = \|\mathbf{x}_{i,\cdot} - \mathbf{y}\|_2$.

27. Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\mathbf{Y} \in \mathbb{R}^{m \times d}$ represent two sets of points in $\mathbb{R}^d$. Return an index vector $\mathbf{r}$ of length m such that r_i indicates the point in $\mathbf{X}$ with the least distance to the i-th point in $\mathbf{Y}$, i.e., $r_i = \arg\min_j \|\mathbf{x}_{j,\cdot} - \mathbf{y}_{i,\cdot}\|_2$.

[redacted]