201600282 엄기산

In [3]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

In [4]:
```
#벡터 연습
import re, math, random # regexes, math functions, random numbers
import matplotlib.pyplot as plt # pyplot
from collections import defaultdict, Counter
from functools import partial, reduce # For python3, "reduce" function is added

import numpy as np

def vector_add(v, w):
    return [v_i + w_i for v_i, w_i in zip(v,w)]

def vector_subtract(v, w):
    return [v_i - w_i for v_i, w_i in zip(v,w)]

def vector_sum(vectors):
    return reduce(vector_add, vectors)

def vector_sum_modified(vectors):
    return [sum(value) for value in zip(*vectors)]

def scalar_multiply(c, v):
    return [c * v_i for v_i in v]

def vector_mean(vectors):
    n = len(vectors)
    return scalar_multiply(1/n, vector_sum(vectors))

def dot(v, w):
    return sum(v_i * w_i for v_i, w_i in zip(v, w))

def sum_of_squares(v):
    return dot(v, v)

def magnitude(v):
    return math.sqrt(sum_of_squares(v))

def squared_distance(v, w):
    return sum_of_squares(vector_subtract(v, w))

def distance(v, w):
    return math.sqrt(squared_distance(v, w))


v = [x for x in range(1, 11, 2)]
w = [y for y in range(11, 21, 2)]
scalar = 3

vector_add(v,w)
#%timeit vector_add(v, w)

vector_subtract(v, w)
#%timeit vector_subtract(v, w)

vectors = [v,w,v,w,v,w]
vector_sum(vectors)

vectors = [v,w,v,w,v,w]
```

```
vector_sum_modified(vectors)

scalar_multiply(scalar,v)

vector_mean([v,v,v,v])

dot(v, w)

magnitude(v)

distance(v,w)
```

Out[4]: [12, 16, 20, 24, 28]

Out[4]: [-10, -10, -10, -10, -10]

Out[4]: [36, 48, 60, 72, 84]

Out[4]: [36, 48, 60, 72, 84]

Out[4]: [3, 9, 15, 21, 27]

Out[4]: [1.0, 3.0, 5.0, 7.0, 9.0]

Out[4]: 415

Out[4]: 12.84523257866513

Out[4]: 22.360679774997898

In [5]:

```
#벡터연습 numpy
np.array(v) + np.array(w)
#%timeit np.array(v) + np.array(w)

np.array(v) - np.array(w)
#%timeit np.array(v) - np.array(w)

np.sum([v,w,v,w,v,w], axis=0)

scalar * np.array(v)

np.mean([v,v,v,v], axis=0)

np.dot(v,w)

np.linalg.norm(v)

np.linalg.norm(np.subtract(v,w))
```

Out[5]: array([12, 16, 20, 24, 28])

Out[5]: array([-10, -10, -10, -10, -10])

Out[5]: array([36, 48, 60, 72, 84])

Out[5]: array([3, 9, 15, 21, 27])

Out[5]: array([1., 3., 5., 7., 9.])

Out[5]: 415

Out[5]: 12.84523257866513

Out[5]: 22.360679774997898

In [6]:

```
#행렬연습
def shape(A):
```

```python
    num_rows = len(A)
    num_cols = len(A[0]) if A else 0
    return num_rows, num_cols

def get_row(A, i):
    return A[i]

def get_column(A, j):
    return [A_i[j] for A_i in A]

def make_matrix(num_rows, num_cols, entry_fn):
    return [[entry_fn(i, j) for j in range(num_cols)]
            for i in range(num_rows)]

def is_diagonal(i, j):
    return 1 if i == j else 0

def matrix_add(A, B):
    if shape(A) != shape(B):
        raise ArithmeticError("cannot add matrices with different shapes")

    num_rows, num_cols = shape(A)
    def entry_fn(i, j): return A[i][j] + B[i][j]

    return make_matrix(num_rows, num_cols, entry_fn)


example_matrix = [[1,2,3,4,5], [11,12,13,14,15], [21,22,23,24,25]]

shape(example_matrix)
get_row(example_matrix, 0)
get_column(example_matrix,3)

identity_matrix = make_matrix(5, 5, is_diagonal)

identity_matrix

friendships = [(0, 1), (0, 2), (1, 2), (1, 3), (2, 3), (3, 4), (4, 5), (5, 6), (5, 7),

friendships = [[0, 1, 1, 0, 0, 0, 0, 0, 0, 0],
               [1, 0, 1, 1, 0, 0, 0, 0, 0, 0],
               [1, 1, 0, 1, 0, 0, 0, 0, 0, 0],
               [0, 1, 1, 0, 1, 0, 0, 0, 0, 0],
               [0, 0, 0, 1, 0, 1, 0, 0, 0, 0],
               [0, 0, 0, 0, 1, 0, 1, 1, 0, 0],
               [0, 0, 0, 0, 0, 1, 0, 0, 1, 0],
               [0, 0, 0, 0, 0, 1, 0, 0, 1, 0],
               [0, 0, 0, 0, 0, 0, 1, 1, 0, 1],
               [0, 0, 0, 0, 0, 0, 0, 0, 1, 0]]

friendships[0][2] == 1
friendships[0][8] == 1

friends_of_five = [i for i, is_friend in enumerate(friendships[5]) if is_friend]
print(friends_of_five)

A = [[ 1., 0., 0.], [ 0., 1., 2.]]
B = [[ 5., 4., 3.], [ 2., 2., 2.]]

matrix_add(A,B)
```

Out[6]: (3, 5)

Out[6]: [1, 2, 3, 4, 5]

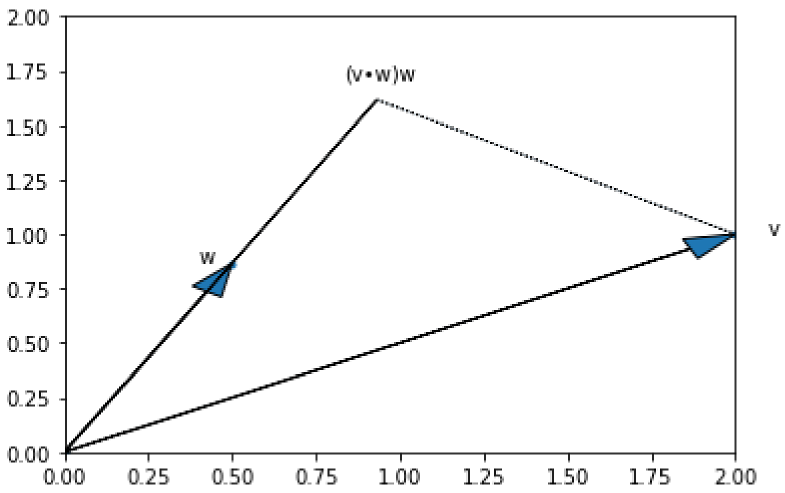
```
Out[6]: [4, 14, 24]

Out[6]: [[1, 0, 0, 0, 0],
         [0, 1, 0, 0, 0],
         [0, 0, 1, 0, 0],
         [0, 0, 0, 1, 0],
         [0, 0, 0, 0, 1]]

Out[6]: True

Out[6]: False

        [4, 6, 7]
Out[6]: [[6.0, 4.0, 3.0], [2.0, 3.0, 4.0]]
```

```
In [7]: #행렬연습 numpy
np.shape(example_matrix)
example_matrix = np.array(example_matrix)
example_matrix[0]
example_matrix[:,3]

np.identity(5)

np.add(A,B)

np.transpose(A)

matrix_a = np.array(A)

matrix_a

np.transpose(B)

np.dot(A,np.transpose(B))
```

```
Out[7]: (3, 5)

Out[7]: array([1, 2, 3, 4, 5])

Out[7]: array([ 4, 14, 24])

Out[7]: array([[1., 0., 0., 0., 0.],
               [0., 1., 0., 0., 0.],
               [0., 0., 1., 0., 0.],
               [0., 0., 0., 1., 0.],
               [0., 0., 0., 0., 1.]])
Out[7]: array([[6., 4., 3.],
               [2., 3., 4.]])
Out[7]: array([[1., 0.],
               [0., 1.],
               [0., 2.]])
Out[7]: array([[1., 0., 0.],
               [0., 1., 2.]])
Out[7]: array([[5., 2.],
               [4., 2.],
               [3., 2.]])
Out[7]: array([[ 5.,  2.],
               [10.,  6.]])
```

```
In [8]: #벡터 점곱 그래프
def make_graph_dot_product_as_vector_projection(plt):
    v = [2, 1]
    w = [math.sqrt(.25), math.sqrt(.75)]
    c = dot(v, w)
    vonw = scalar_multiply(c, w)
    o = [0,0]
```

```
    plt.arrow(0, 0, v[0], v[1],
              width=0.002, head_width=.1, length_includes_head=True)
    plt.annotate("v", v, xytext=[v[0] + 0.1, v[1]])
    plt.arrow(0 ,0, w[0], w[1],
              width=0.002, head_width=.1, length_includes_head=True)
    plt.annotate("w", w, xytext=[w[0] - 0.1, w[1]])
    plt.arrow(0, 0, vonw[0], vonw[1], length_includes_head=True)
    plt.annotate(u"(v•w)w", vonw, xytext=[vonw[0] - 0.1, vonw[1] + 0.1])
    plt.arrow(v[0], v[1], vonw[0] - v[0], vonw[1] - v[1],
              linestyle='dotted', length_includes_head=True)
    plt.scatter(*zip(v,w,o),marker='.')
    plt.axis([0,2,0,2])
    plt.show()

%matplotlib inline
make_graph_dot_product_as_vector_projection(plt)
```



In [15]:
```
#201600282 엄기산 LAB5


def my_matrix_dot(A,B):
    if len(A[0]) != len(B) :
        raise ArithmeticError("number of colums in the first matrix must be equal to

    num_rows = len(A)
    num_cols = len(B[0])

    def entry_fn1(i, j):
        ABij = 0
        for k in range(0,len(B)):
            ABij = ABij + A[i][k]*B[k][j]
        return ABij

    return make_matrix(num_rows, num_cols, entry_fn1)

def my_matrix_transpose(M):
    num_rows = len(M[0])
    num_cols = len(M)

    def entry_fn2(i,j):
        return M[j][i]

    return make_matrix(num_rows, num_cols, entry_fn2)

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

```
    [3,4],
    [5,6]])

dotAB=np.array(my_matrix_dot(A,B))
transA=np.array(my_matrix_transpose(A))
transB=np.array(my_matrix_transpose(B))
dotAB
transA
transB
```

Out[15]:
```
array([[22, 28],
       [49, 64]])
```

Out[15]:
```
array([[1, 4],
       [2, 5],
       [3, 6]])
```

Out[15]:
```
array([[1, 3, 5],
       [2, 4, 6]])
```

201600282 엄기산