Vector-vector dot Product

```
import numpy as np
import matplotlib.pyplot as plt
xv = np.random.rand(10)
print(xv)
print(type(xv),xv.ndim,xv.shape)
for i in range(len(xv)):
  print(i, ":", xv[i])
for x1,x2 in zip(xv,xv):
  print(x1,x2)
def v_v(wv,xv):
  total = 0
  for w,v in zip(wv,xv):
    total += w*v
  return total

  # Test
  w = np.random.rand(10)
  x = np.ones_like(w)
  print(v_v(w,x))
  print(np.sum(w))
```

```
[0.99523548 0.3670788  0.90936806 0.82855821 0.77645853 0.86966391
 0.85404306 0.35273337 0.45532102 0.98936072]
<class 'numpy.ndarray'> 1 (10,)
0 : 0.9952354821243785
1 : 0.36707880220427
2 : 0.9093680641592986
3 : 0.8285582128456196
4 : 0.7764585272595899
5 : 0.8696639113073579
6 : 0.8540430638079562
7 : 0.3527333717350468
8 : 0.45532101812016945
9 : 0.9893607188075757
0.9952354821243785 0.9952354821243785
0.36707880220427 0.36707880220427
0.9093680641592986 0.9093680641592986
0.8285582128456196 0.8285582128456196
0.7764585272595899 0.7764585272595899
0.8696639113073579 0.8696639113073579
0.8540430638079562 0.8540430638079562
0.3527333717350468 0.3527333717350468
0.45532101812016945 0.45532101812016945
0.9893607188075757 0.9893607188075757
```

Matrix - Vector Product

```
def m_v(wm, xv):
  total = []
  for wv in wm:
    total.append(v_v(wv,xv))
  return total
#Test
w = np.random.rand(10,10)
x = np.ones(10)
print(m_v(w,x))
print(np.sum(w))
```

```
[5.593588432072418, 4.6555689132354185, 3.6628320377603165, 3.761801109314055, 3.5364956765908655, 4.176270259524638, 5.9584192509079035, 4.540755651243235, 4.0362355283909
45.32356018742772
```

Matrix - Matrix Product

```
def m_m(wm,xm):
  total = []
  for w,m in zip(wm, xm.T):
    total.append(v_v(w,x))
  return total
#Test
w = np.random.rand(10,10)
x = np.ones(10)
print(m_m(w,x))
print(np.sum(w))
```

```
[4.77312221625035, 4.558645705431309, 4.036302076584782, 6.2207255212131445, 5.399355246489991, 6.06040472358512, 5.1489951487896555, 5.04061965011452, 5.025739159133071, 4
51.00160524974011
```

Time it

```
import logging
import cProfile
import time

start = time.perf_counter()
t = np.matmul(w,x)
end = time.perf_counter()
print("np matmul:", end-start);
start = time.perf_counter()
t = m_m(w,x)
end = time.perf_counter()
print("m_m :", end - start)
```

```
np matmul: 0.0004063939999809918
m_m : 0.000293160000012449
```