Pseudoinverse and Least Squares

In [1]:

```
#keep
import numpy as np
import numpy.linalg as la

np.set_printoptions(precision=4, linewidth=100)
```

In [2]:

```
#keep
A = np.random.randn(5, 3)
```

Now compute the SVD of A. Note that `numpy.linalg.svd` returns V^T:

In [3]:

```
U, singval, VT = la.svd(A)
V = VT.T
```

Let's first understand the shapes of these arrays:

In [4]:

```
#keep
print(U.shape)
print(singval.shape)
print(V.shape)
```

```
(5, 5)
(3,)
(3, 3)
```

Check the orthogonality of U and V:

In [5]:

```
U.T.dot(U)
```

Out[5]:

```
array([[  1.0000e+00,  -1.6053e-16,  -1.9500e-16,   2.1064e-17,   0.
0000e+00],
       [ -1.6053e-16,   1.0000e+00,  -8.5663e-17,   7.0205e-18,   1.
1102e-16],
       [ -1.9500e-16,  -8.5663e-17,   1.0000e+00,   1.2369e-18,   0.
0000e+00],
       [  2.1064e-17,   7.0205e-18,   1.2369e-18,   1.0000e+00,   2.
7756e-17],
       [  0.0000e+00,   1.1102e-16,   0.0000e+00,   2.7756e-17,   1.
0000e+00]])
```

In [6]:

```
V.T.dot(V)
```

Out[6]:

```
array([[  1.0000e+00,  -3.4694e-17,  -7.8063e-18],
       [ -3.4694e-17,   1.0000e+00,  -5.5511e-17],
       [ -7.8063e-18,  -5.5511e-17,   1.0000e+00]])
```

Now build the matrix Σ:

In [7]:

```
Sigma = np.zeros(A.shape)
Sigma[:3, :3] = np.diag(singval)
Sigma
```

Out[7]:

```
array([[ 2.4071,  0.    ,  0.    ],
       [ 0.    ,  2.2117,  0.    ],
       [ 0.    ,  0.    ,  0.4908],
       [ 0.    ,  0.    ,  0.    ],
       [ 0.    ,  0.    ,  0.    ]])
```

Now piece A back together from the factorization:

In [8]:

```
U.dot(Sigma).dot(V.T) - A
```

Out[8]:

```
array([[  2.2204e-16,   5.5511e-16,  -2.4425e-15],
       [  2.2204e-16,  -6.1062e-16,   7.7716e-16],
       [  0.0000e+00,   1.1102e-16,  -4.1633e-16],
       [ -1.1102e-16,   6.9389e-17,   5.5511e-17],
       [ -1.1102e-16,   2.4980e-16,  -1.6653e-16]])
```

Next, compute the pseudoinverse:

In [9]:

```
SigmaInv = np.zeros((3,5))
SigmaInv[:3, :3] = np.diag(1/singval)
SigmaInv
```

Out[9]:

```
array([[ 0.4154,  0.    ,  0.    ,  0.    ,  0.    ],
       [ 0.    ,  0.4521,  0.    ,  0.    ,  0.    ],
       [ 0.    ,  0.    ,  2.0373,  0.    ,  0.    ]])
```

In [10]:

```
A_pinv = V.dot(SigmaInv).dot(U.T)
```

Now use the pseudoinverse to "solve" $Ax = b$ for our tall-and-skinny A:

In [11]:

```
#keep
b = np.random.randn(5)
```

In [12]:

```
A_pinv.dot(b)
```

Out[12]:

```
array([ 0.504 , -0.3182,  0.9697])
```