Matrix: A = [1 2 ; 3 4 ; 5 6] Sid

Vector: v = [1 2 3] , v = [1 ; 2 ; 3]

size(A) = [r c] , length(v) size(A,1) : r

A = ones(r,c) size(A,2) : c

hist(—) // histogram

eye(n) // n × n identity matrix

load('file.dat')

who(s) // shows variables

clear —

save file.abc x , -ascii // text

: ever element along row/column

A(r,:), A(:,c) , A(r,c)

A(:) all elements into column vector

Computational Operations

A × B , A.× B

matrix multiplication

element wize operation

A' (transpose) max(), rand

sum(a) , prod(a) , floor(a), ceil(a)

pinv(A) // inverse

max(A,[], 1 row, 2 column

Plotting Data

```
plot(x, y)
hold on;
clf;
```

```
xlabel( ), ylabel( ), legend( )
print -dpng "file"
axis([x1 x2 y1 y2])
subplot(r, c, element #)
imagesc(A), colorbar, colormap gray
```

Control Statements

```
v = zeros(10,1)          <-->    indices = 1:10:
for i = start : end,              for i = indices
    v(i) = 2^i;
end;
```

```
i = 1                       while true,
while i <= _,                  v(i) = 999;
    ______;                    i = i+1;
    i = i+1;                   if i == 6,
end;                              break;
                               end;
                            end;
```

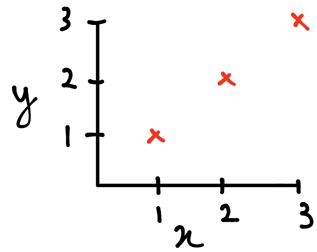
functions : file.m

- ```
  function y = squareNumber(x)
  y = x^2;
  ```

- ```
  function [y1, y2] = squareAndCubeNumber(x)
  y1 = x^2;
  y2 = x^3;
  ```

$$\begin{array}{cccc} X & & \theta & \\ \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} & \times & \begin{bmatrix} 0 \\ 1 \end{bmatrix} & = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \\ 3 \times 2 & \times & 2 \times 1 & \end{array}$$

$\theta' = [0 \ 1]$



Cost function :

$x = [1\ 1 ; 1\ 2 ; 1\ 3]$

$$\begin{array}{cc} x_0 & x_1 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{array}$$

$y = [1 ; 2 ; 3]$ $\begin{array}{c} 1 \\ 2 \\ 3 \end{array}$

theta = [0 ; 1] $\begin{array}{c} 0 \\ 1 \end{array}$

```
function J = costFunctionJ (X, y, theta)
    m= size (X, 1)   // 3 rows / training examples
    predictions = X * theta ;
    sqrErrors = (predictions - y).^2 ;
    J = 1/(2*m) * sum(sqrErrors) ;
```

Vectorization

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j$$

$$= \theta^T x$$

<u>unvectorized</u> :

```
prediction = 0.0 ;
for j = 1:n+1,
    prediction = prediction
        + theta(j) * x(j)
end ;
```

<u>vectorized</u>

```
prediction = theta' * x ;
```

Gradient Descent

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}$$

<u>Vectorized</u>:

$$\vec{\theta} := \vec{\theta} - \alpha \vec{\Delta} \quad , \quad \Delta = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \vec{x}^{(i)}$$

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \end{bmatrix}$$

$$\Delta = \begin{bmatrix} \Delta_0 \\ \Delta_1 \\ \Delta_2 \\ \vdots \end{bmatrix}$$