Tom Nguyen
Dr. Xiaobai Liu
CS 596 MW
February 16, 2018

Homework Assignment 1

Question 1 – Straight Line y= mx + c

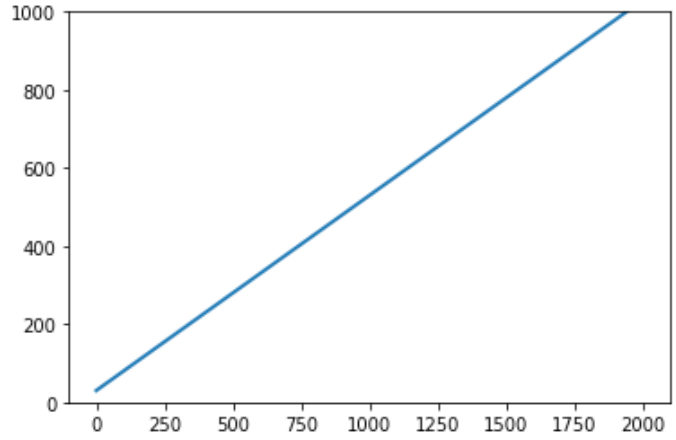
In [27]:
```
import numpy as np
import matplotlib.pyplot as plt

ax = plt.subplot(111)

t = np.arange(0.0, 2000, 0.01)#x-axis
#y = np.cos(2*np.pi*x)
#x=30
s=30+.5*t
line, = plt.plot(t, s, lw=2)#(x,y)

#xytext=how curvy the arrow should be.
#plt.annotate('line', xy=(1000, 500), xytext=(1100,600 ),#xy=hard coding local x
#             arrowprops=dict(facecolor='black', shrink=.00),#shrink = for arrow
#             )

plt.ylim(0,1000)#range y-axis
plt.show()
```



Quadratic function: $y = a(x-h)^2 + k$

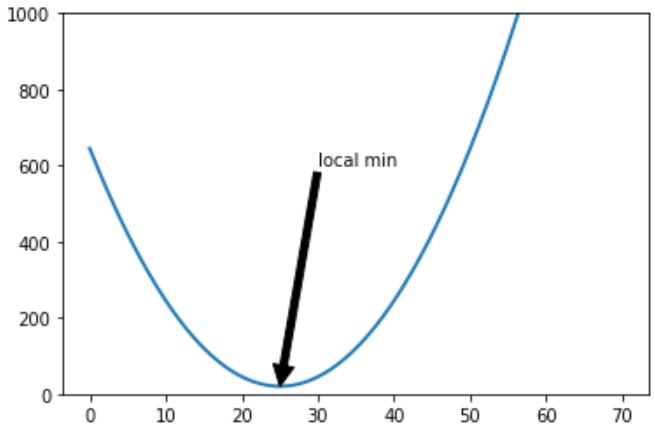
In [34]:

```
import numpy as np
import matplotlib.pyplot as plt

ax = plt.subplot(111)
x=70
t = np.arange(0.0, x,.01)#x-axis input
#y = np.cos(2*np.pi*x)
#variable zero
v0=20#theta_0
#variable one
v1=25#theta_1
s=(t-v1)**2 +v0 #output
line, = plt.plot(t, s, lw=2)#(x,y)

#xytext=how curvy the arrow should be.
plt.annotate('local min', xy=(25, 20), xytext=(30,600 ),#xy=hard coding local m
             arrowprops=dict(facecolor='black', shrink=.00),#shrink = for arrow
             )

plt.ylim(0,1000)#range y-axis
plt.show()
```

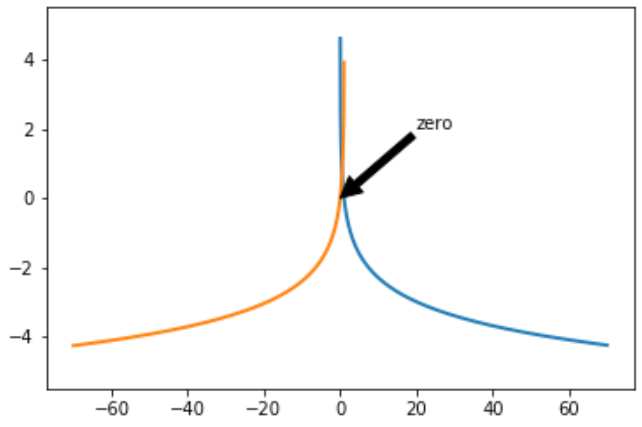


Log Function: log(b) and log(1-a)

In [19]:
```
import numpy as np
import matplotlib.pyplot as plt
import math

ax = plt.subplot(111)
x=70
t = np.arange(0.01, x,.01)#x-axis input
#y = np.cos(2*np.pi*x)
#variable zero
v0=20#theta_0
#variable one
v1=25#theta_1
s=-np.log(t) #output
q=np.arange(-x,.99,.01)
w=-np.log(1-q)
line, = plt.plot(t, s, lw=2)#(x,y)
line, = plt.plot(q,w,lw=2)
#xytext=how curvy the arrow should be.
plt.annotate('zero', xy=(0, 0), xytext=(20,2 ),#xy=hard coding local max/min
             arrowprops=dict(facecolor='black', shrink=.00),#shrink = for arrow
             )

plt.ylim(-5.5,5.5)#range y-axis
plt.show()
```

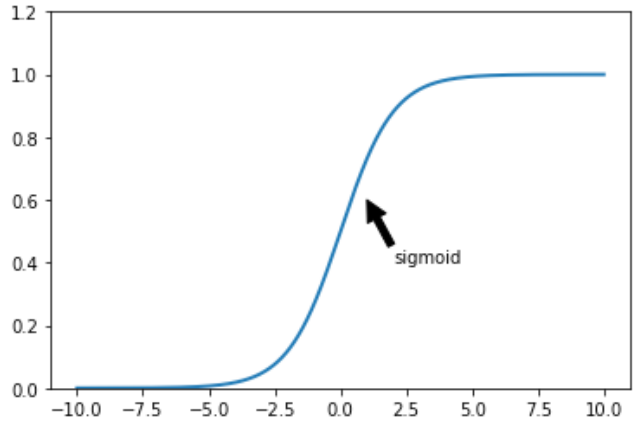


Sigmoid function: $1 / (1+e^{-a})$

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import math

ax = plt.subplot(111)
x=10
t = np.arange(-10, x,.01)#x-axis input
#y = np.cos(2*np.pi*x)
#variable zero
v0=20#theta_0
#variable one
v1=25#theta_1
s=1/(1+np.e**(-t)) #output
line, = plt.plot(t, s, lw=2)#(x,y)
#xytext=how curvy the arrow should be.
plt.annotate('sigmoid', xy=(1,0.6), xytext=(2,0.4 ),#xy=hard coding local max/m
             arrowprops=dict(facecolor='black', shrink=.00),#shrink = for arrow
             )

plt.ylim(0,1.2)#range y-axis
plt.show()
```



Question 2: B

B) i) supervised learning with discrete predictions; ii) supervised learning with discrete predictions; iii) unsupervised learning with discrete results;

Question 3: A & D

A)This problem can be solved by supervised learning that uses a single feature as predictor;

D) To solve this problem, we should compare different models, being complex or not, over both training data and testing data before making a selection.

Question 4:
My machine learning task will be a classification on rather the price of bitcoin will go up or down. I will be using supervised learning with discrete predictions.

Input:
The input data for this model will be the prices of bitcoin and the dates associated with it.

Output:
prompts rather it goes up or down.(should I worry about if the price is the same?)

Goal:
To predict if the future price of bitcoin will go up or down.

Data preparation:
Training data:
the history of bitcoin from Coinbase.

Validation data:
Using the algorithm from machine learning to predict if the price will go up or down at the end of the day from the model.

Testing data:
At the end of each day,same time every day, we will check the cost of bitcoin to validate our predictions.

Ground-truth:
The ground-truth will be taken from Coinbase history. And continuous updated at the end of each day.