Pb 1

In [1]:
```python
import numpy as np
```

In [2]:
```python
n = 10
d = 100
w = np.random.standard_normal(size=(1, d))
```

In [3]:
```python
X = np.random.standard_normal(size=(d, n))
y= np.matmul(w, X)
```

(a)

In [4]:
```python
w_est = np.matmul(y, np.linalg.pinv(X))
```

(b)

In [5]:
```python
mse = ((w - w_est)**2).mean()
```

In [6]:
```python
mse
```

Out[6]:
```
0.9771778540388765
```

(c)

In [7]:
```python
# abstraction
def compute_mse(n):
    d = 100
    X = np.random.standard_normal(size=(d, n))
    y= np.matmul(w, X)
    w_est = np.matmul(y, np.linalg.pinv(X))
    return ((w - w_est)**2).mean()
```

In [8]:
```python
N = [1, 10, 20, 30, 40, 50, 60, 70, 80, 90]
```

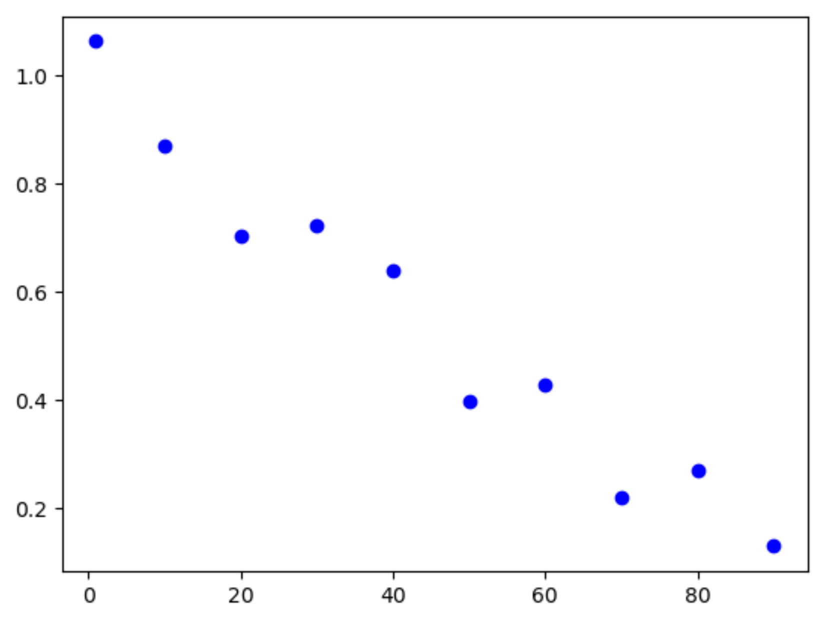
In [9]:
```python
MSE = []
for n in N:
    MSE.append(compute_mse(n))
```

In [10]:
```python
MSE
```

Out[10]:
```
[1.063279804213499,
 0.8710203803937349,
 0.7039548100483457,
 0.7221590533458447,
 0.6388734162969246,
 0.3968917929248041,
 0.42853459888666223,
 0.21950206086448973,
 0.2681749526652219,
 0.12860264974031302]
```

In [11]:
```python
import matplotlib.pyplot as plt
plt.scatter(N, MSE, c ="blue")
```

Out[11]:
```
<matplotlib.collections.PathCollection at 0x1235ef0d0>
```

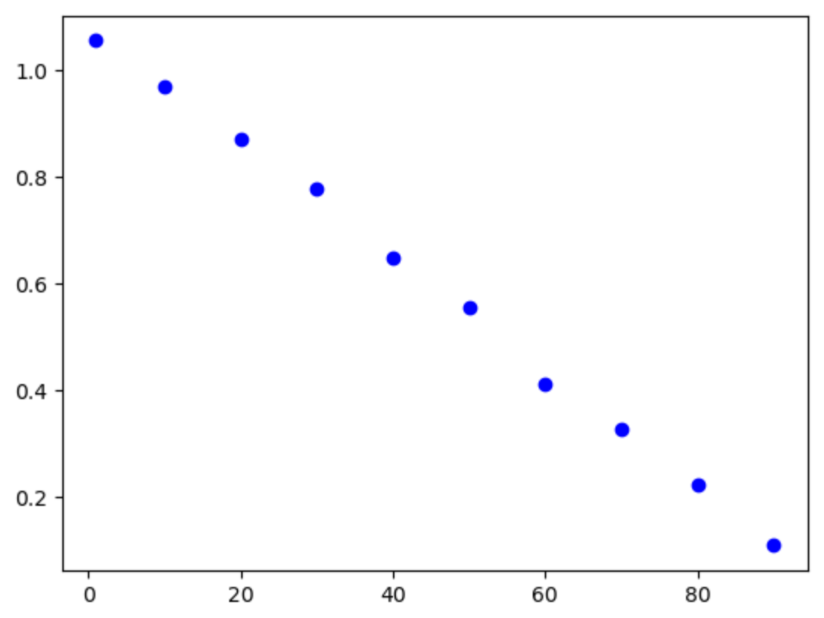


(d)

In [12]:
```python
# repeat 30 times
seeds = [i for i in range(30)]
repeat = 30

MSE_AVG = []
for n in N:
    MSE = []
    for seed in seeds:
        np.random.seed(seed)
        MSE.append(compute_mse(n))
    MSE_AVG.append(np.mean(MSE))
```

In [13]:
```python
import matplotlib.pyplot as plt
plt.scatter(N, MSE_AVG, c ="blue")
```

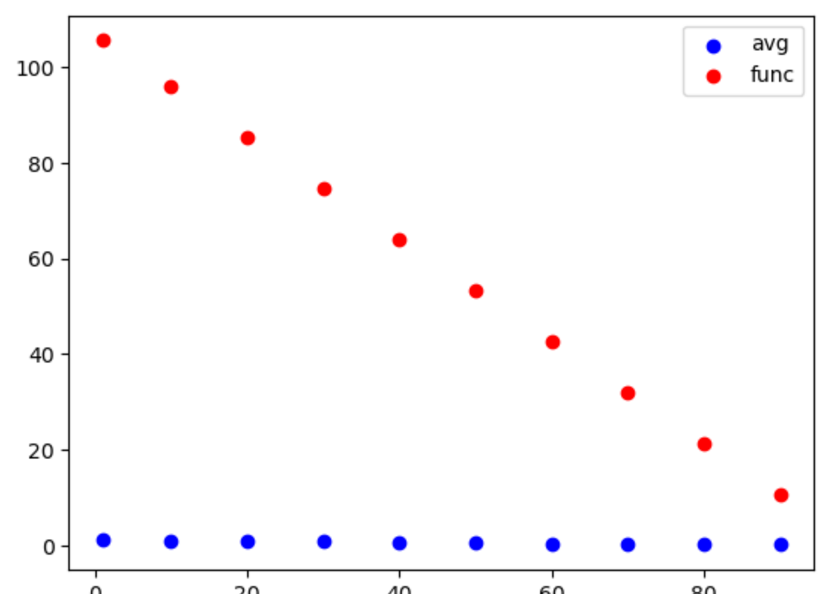
Out[13]:
```
<matplotlib.collections.PathCollection at 0x1277f88e0>
```



(e)

In [14]:
```python
F = []
for n in N:
    F.append(np.inner(w, w)*(1-n/100))
plt.scatter(N, MSE_AVG, c ="blue", label="avg")
plt.scatter(N, F, c ="red", label="func")
plt.legend()
```

Out[14]:
```
<matplotlib.legend.Legend at 0x127860be0>
```



Pb 5

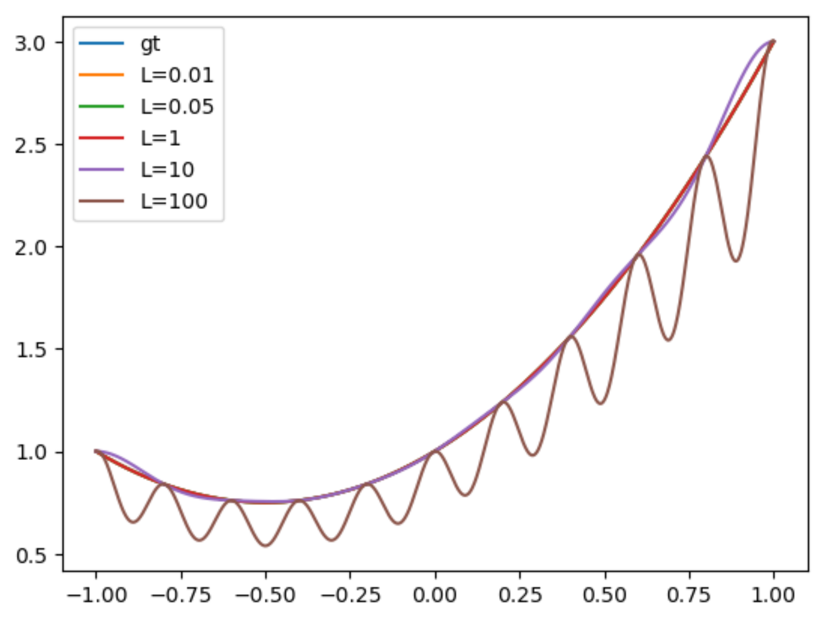
In [15]:
```python
f = lambda x : x**2+x+1

n_train = 11
X_train = np.linspace(-1, 1, num=n_train)
Y_train = [f(x) for x in X_train]
```

(a)

In [16]:
```python
from numpy.linalg import norm, solve

n_test = 1000
X_test = np.linspace(-1, 1, num=n_test)
Y_test = [f(x) for x in X_test]

def gaussian_kernel(x1, x2, L=10):
    return np.exp(-L*norm([x1 - x2], ord=2)**2)

L = [0.01, 0.05, 1, 10, 100]
pred = {}
for l in L:
    # step 1: construct K using X_train
    K = np.zeros((n_train, n_train))
    for i in range(n_train):
        for j in range(n_train):
            K[i, j] = gaussian_kernel(X_train[i], X_train[j], L=l)
    # step 2: solve for getting alpha
    alpha = solve(K, Y_train)

    # step 3: compute K for pred
    k = np.zeros((n_test, n_train))
    for i in range(n_test):
        for j in range(n_train):
            k[i, j] = gaussian_kernel(X_test[i], X_train[j], L=l)

    # step 4: Compute f(x) = alpha * k
    pred[l] = k @ alpha
```

In [17]:
```python
plt.plot(X_test, Y_test, label="gt")
for l in L:
    plt.plot(X_test, pred[l], label=f"L={l}")
plt.legend()
```

Out[17]:
```
<matplotlib.legend.Legend at 0x1278f2310>
```



(b)

In [18]:
```python
def gaussian_kernel_no_square(x1, x2, L=10):
    return np.exp(-L*norm([x1 - x2], ord=2))

L = [0.01, 0.05, 1, 10, 100]
pred = {}
for l in L:
    # step 1: construct K using X_train
    K = np.zeros((n_train, n_train))
    for i in range(n_train):
        for j in range(n_train):
            K[i, j] = gaussian_kernel_no_square(X_train[i], X_train[j], L=l)
    # step 2: solve for getting alpha
    alpha = solve(K, Y_train)

    # step 3: compute K for pred
    k = np.zeros((n_test, n_train))
    for i in range(n_test):
        for j in range(n_train):
            k[i, j] = gaussian_kernel_no_square(X_test[i], X_train[j], L=l)

    # step 4: Compute f(x) = alpha * k
    pred[l] = k @ alpha
```

In [19]:
```python
plt.plot(X_test, Y_test, label="gt")
for l in L:
    plt.plot(X_test, pred[l], label=f"L={l}")
plt.legend()
```

Out[19]:
```
<matplotlib.legend.Legend at 0x12797bcd0>
```