

# CN25 - Homework 3

Matteo Mazzetti 0001161552

## 1 Regressione lineare semplice

### 1.1 Problema test

Creiamo un problema test in Python con un vettore arbitrario alphaTrue tale che  $\alpha_0 = 0$ ,  $\alpha_i = i^{\frac{1}{2i}}$  per  $i = 1..d$  e con n valori  $x_i$  equispaziati in [0,1].

```
def f(x, alpha):
    d=alpha.shape[0]-1
    y=0
    for i in range(d+1):
        y=y+alpha[i]*x**i
    return y
d=8
alphaTrue=np.zeros((d+1,))
for i in range(d+1):
    if(i==0): alphaTrue[i]=0
    else: alphaTrue[i]=np.sqrt(i)**(1/i)
n=15
x=np.linspace(0, 1, n)
e=np.random.normal(loc=0,scale=0.1, size=(n,))
y=np.zeros_like(x)
for i in range(n):
    y[i]=f(x[i],alphaTrue)+e[i]
def f(x, alpha):
    d=alpha.shape[0]-1
    y=np.zeros_like(x)
    for i in range(d+1):
        y=y+alpha[i]*x**i
    return y
def vandermonde(x,d):
    n=x.shape[0]
    X=np.zeros((n,d+1))
    for i in range(d+1):
        X[:,i]=x**i
    return X
X=vandermonde(x,d)
def SVD(X,y,d):
    n=y.shape[0]
    U,s,Vt=np.linalg.svd(X)
    Sigma=np.zeros((n,d+1))
    for i in range(d+1):
        Sigma[i,i]=s[i]
    alphaSVD=np.zeros((d+1,))
    for i in range(d+1):
        alphaSVD+=(U[:,i].T @ y)/s[i]*Vt[i,:]
    return alphaSVD
alphaSVD=SVD(X,y,d)
def residue(X,y,alpha):
    return np.linalg.norm(X @ alpha - y)
print(f"Residuo SVD: {residue(X, y, alphaSVD)}")
print(f"Residuo True (e): {residue(X, y, alphaTrue)}")
```

Eseguiamo lo snippet di codice precedente e utilizziamo una tabella per visualizzare gli output:

Test	n	d	alphaTrue	Residuo True (e)	Residuo SVD
1	6	5	(0, 1, 1.18920712, 1.20093696, 1.18920712, 1.17461894)	0.20034206385514466	5.1038709474956296e-14
2	10	2	(0, 1, 1.18920712)	0.2679803924132402	0.1086352499303631
3	10	5	(0, 1, 1.18920712, 1.20093696, 1.18920712, 1.17461894)	0.4252414548545332	0.22701582110711566
4	15	8	(0, 1, 1.18920712, 1.20093696, 1.18920712, 1.17461894, 1.16103667, 1.14911673, 1.13878863)	0.3876815795154283	0.1796314890672186

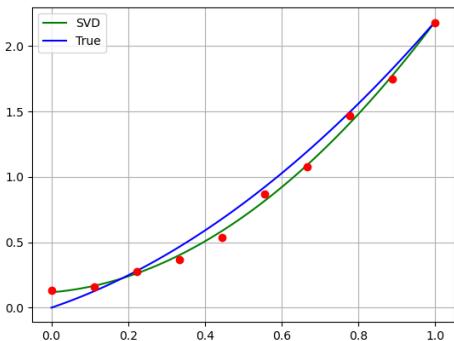


Figure 1: Test 2

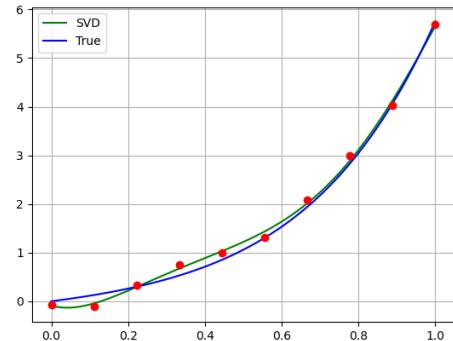


Figure 2: Test 3

## 1.2 Salary dataset

Usiamo ora le funzioni scritte in precedenza (vandermonde, SVD, f) per problema vero: troviamo la funzione che approssima meglio i dati del dataset Salary\_Data.csv che mette in relazione gli anni di lavoro con il salario.

```
dataSet=pd.read_csv("dataSets/Salary_Data.csv")
print(f"Head dataset: \n{dataSet.head()}")
xRaw=dataSet["YearsExperience"].values
yRaw=dataSet["Salary"].values
print(f"Head dataset: \n{dataSet.head()}")
print(f"Descrizione dataset: \n{dataSet.describe()}")
xRaw=dataSet["YearsExperience"].values
yRaw=dataSet["Salary"].values
print(f"Numero di righe: {yRaw.shape[0]}")
d=5
X=vandermonde(xRaw,d)
alphaSVD=SVD(X,yRaw,d)
```

L'output di questo frammento di codice è:

```
Head dataset:
  YearsExperience      Salary
0            1.1    39343.0
1            1.3    46205.0
2            1.5    37731.0
3            2.0    43525.0
4            2.2    39891.0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  -- 
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
```

```
memory usage: 612.0 bytes
```

Descrizione dataset:

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

Numero di righe:30

Vediamo al variare del grado (d) del polinomio approssimante come cambia la funzione.

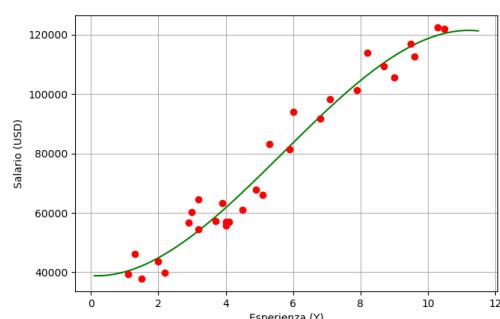


Figure 3: d=3

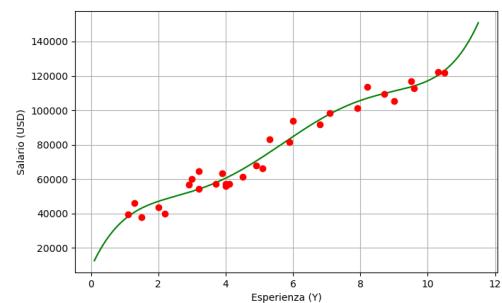


Figure 4: d=5

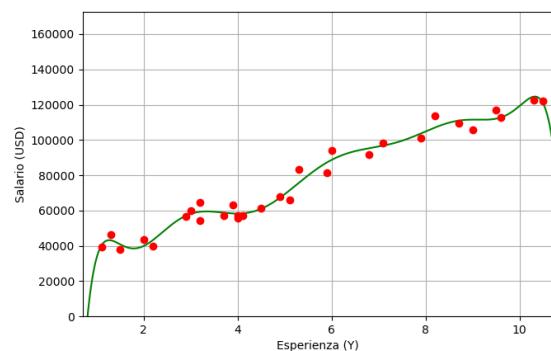


Figure 5: d=10