Regression_ML_professor

September 28, 2020

1 Parametric Model-Based regression

```
Notebook version: 1.3 (Sep 20, 2019)

Author: Jesús Cid-Sueiro (jesus.cid@uc3m.es)
        Jerónimo Arenas García (jarenas@tsc.uc3m.es)
```

```
Changes: v.1.0 - First version, expanding some cells from the Bayesian Regression
                 notebook
         v.1.1 - Python 3 version.
         v.1.2 - Revised presentation.
         v.1.3 - Updated index notation
```

```
Pending changes: * Include regression on the stock data
```

[3]:
```
# Import some libraries that will be necessary for working with data and␣
↪displaying plots

# To visualize plots in the notebook
%matplotlib inline

import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import scipy.io       # To read matlab files
import pylab
```

1.1 A quick note on the mathematical notation

In this notebook we will make extensive use of probability distributions. In general, we will use capital leters $\mathbf{X}$, S, E ..., to denote random variables, and lower-case letters $\mathbf{x}$, s, ϵ ..., to denote the values they can take.

In general, we will use letter p for probability density functions (pdf). When necessary, we will use, capital subindices to make the random variable explicit. For instance, $p_{\mathbf{X},S}(\mathbf{x}, s)$ would be the joint pdf of random variables $\mathbf{X}$ and S at values $\mathbf{x}$ and s, respectively.

However, to avoid a notation overload, we will omit subindices when they are clear from the context. For instance, we will use $p(\mathbf{x}, s)$ instead of $p_{\mathbf{X},S}(\mathbf{x}, s)$.

1.2 1. Model-based parametric regression

1.2.1 1.1. The regression problem

Given an observation vector $\mathbf{x}$, the goal of the regression problem is to find a function $f(\mathbf{x})$ providing *good* predictions about some unknown variable s. To do so, we assume that a set of *labelled* training examples, $\{\mathbf{x}_k, s_k\}_{k=0}^{K-1}$ is available.

The predictor function should make good predictions for new observations $\mathbf{x}$ not used during training. In practice, this is tested using a second set (the *test set*) of labelled samples.

1.2.2 1.2. The underlying model assumption

Many regression algorithms are grounded on the idea that all samples from the training set have been generated independently by some common stochastic process.

If $p(\mathbf{x}, s)$ were known, we could apply estimation theory to estimate s for a given $\mathbf{x}$ using p. For instance, we could apply any of the following classical estimates:

- Maximum A Posterior (MAP):

$$\hat{s}_{\text{MAP}} = \arg\max_s p(s|\mathbf{x})$$

- Minimum Mean Square Error (MSE):

$$\hat{s}_{\text{MSE}} = \mathbb{E}\{S|\mathbf{x}\} = \int s\, p(s|\mathbf{x})\, ds$$

Note that, since these estimators depend on $p(s|\mathbf{x})$, knowing the posterior distribution of the target variable is enough, and we do not need to know the joint distribution $p(\mathbf{x}, s)$.

More importantly, note that **if we knew the underlying model, we would not need the data** in $\mathcal{D}$ to make predictions on new data.

Exercise 1: Assume the target variable s is a scaled noisy version of the input variable x:

$$s = 2x + \epsilon$$

where ϵ is Gaussian a noise variable with zero mean and unit variance, which does not depend on x.

1. Compute the target model $p(s|x)$
2. Compute prediction $\hat{s}_{\text{MAP}}$ for an arbitrary input x
3. Compute prediction $\hat{s}_{\text{MSE}}$ for an arbitrary input x
4. Compute prediction $\hat{s}_{\text{MSE}}$ for input $x = 4$

Solution:

1. Since ϵ is Gaussian, so it is s for a given x, with mean

$$\mathbb{E}\{S \mid x\} = \mathbb{E}\{2x \mid x\} + \mathbb{E}\{\epsilon \mid x\} = 2x + 0 = 2x$$

   and variance

$$\text{Var}\{S \mid x\} = \text{Var}\{2x \mid x\} + \text{Var}\{\epsilon \mid x\} = 0 + 1 = 1$$

   therefore

$$p(s|x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(s - 2x)^2\right)$$

2. The MAP estimate is

$$\hat{s}_{\text{MAP}} = \arg\max_s p(s|x) = 2x$$

3. Since the MSE estimate is the conditional mean, which has been already computed, have $\hat{s}_{\text{MSE}} = 2x$

4. The prediction is $\hat{s}_{\text{MSE}} = 2 \cdot 4 = 8$

1.2.3 1.3. Model-based regression

In practice, the underlying model is usually unknown.

Model based-regression methods exploit the idea of using the training data to estimate the posterior distribution $p(s|\mathbf{x})$ and then apply estimation theory to make predictions.

1.4. Parametric model-based regression

In some cases, we may have a partial knowledge about the underlying mode. In this notebook we will assume that p belongs to a parametric family of distributions $p(s|\mathbf{x}, \mathbf{w})$, where $\mathbf{w}$ is some unknown parameter.

Exercise 2: Assume the target variable s is a scaled noisy version of the input variable x:

$$s = wx + \epsilon$$

where ϵ is Gaussian a noise variable with zero mean and unit variance, which does not depend on x. Assume that w is known. 1. Compute the target model $p(s|x, w)$ 2. Compute prediction $\hat{s}_{\text{MAP}}$ for an arbitrary input x 3. Compute prediction $\hat{s}_{\text{MSE}}$ for an arbitrary input x

Solution:

1. As in Exercise 1 ϵ is Gaussian, so it is s for a given x, with mean

$$\mathbb{E}\{S|x, w\} = wx$$

   and variance

$$\text{Var}\{S|x, w\} = 1$$

   therefore

$$p(s|x, w) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(s - wx)^2\right)$$

2. The MAP estimate is
$$\hat{s}_{\text{MAP}} = \arg\max_{s} p(s|x, w) = wx$$
3. Since the MSE estimate is the conditional mean, which has been already computed, have $\hat{s}_{\text{MSE}} = wx$

We will use the training data to estimate $\mathbf{w}$

The estimation of $\mathbf{w}$ from a given dataset $\mathcal{D}$ is the goal of the following sections

1.3 2. Maximum Likelihood parameter estimation.

The ML (Maximum Likelihood) principle is well-known in statistics and can be stated as follows: take the value of the parameter to be estimated (in our case, $\mathbf{w}$) that best explains the given observations (in our case, the training dataset $\mathcal{D}$). Mathematically, this can be expressed as follows:

$$\hat{\mathbf{w}}_{\text{ML}} = \arg\max_{\mathbf{w}} p(\mathcal{D}|\mathbf{w})$$

Exercise 3: All samples in dataset $\mathcal{D} = \{(x_k, s_k), k = 0, \ldots, K-1\}$

$$s_k = w \cdot x_k + \epsilon_k$$

where ϵ_k are i.i.d. (independent and identically distributed) Gaussian noise random variables with zero mean and unit variance, which do not depend on x_k.

Compute the ML estimate, $\hat{w}_{\text{ML}}$, of w.

Solution: From Exercise 2,

$$p\left(s_k \mid x_k, w\right) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(s_k - wx_k\right)^2\right)$$

Since the noise variables are i.i.d, we have

$$p(\mathcal{D}|w) = p\left(s_0, \ldots, s_{K-1} \mid x_0, \ldots, x_{K-1}, w\right) p\left(x_0, \ldots, x_{K-1} \mid w\right) \qquad = \prod_{k=0}^{K-1} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(s_k - wx_k\right)^2\right) p\left(x_0, \right.$$

Therefore:

$$\hat{\mathbf{w}}_{\text{ML}} = \arg\max_{\mathbf{w}} p(\mathcal{D}|\mathbf{w}) = \arg\min_{\mathbf{w}} \sum_{k=0}^{K-1} \left(s_k - wx_k\right)^2$$

Differentiating with respect to w:

$$-2 \sum_{k=0}^{K-1} \left(s_k - \hat{w}_{\text{ML}} x_k\right) x_k = 0$$

that is

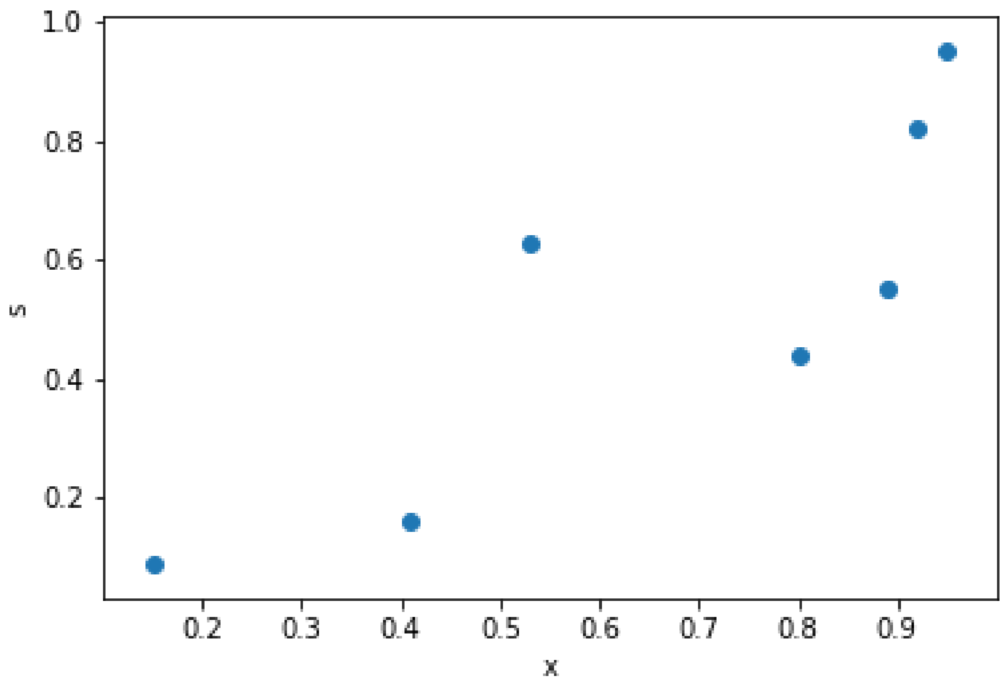
$$\hat{w}_{\text{ML}} = \frac{\sum_{k=0}^{K-1} s_k x_k}{\sum_{k=0}^{K-1} \left(x_k\right)^2}$$

Exercise 4: The inputs and the targets from a dataset $\mathcal{D}$ have been stored in the following Python arrays:

```
[4]: X = np.array([0.15, 0.41, 0.53, 0.80, 0.89, 0.92, 0.95])
     s = np.array([0.09, 0.16, 0.63, 0.44, 0.55, 0.82, 0.95])
```

- **4.1.** Represent a scatter plot of the data points

```
[5]: # <SOL>
     plt.figure()
     plt.scatter(X, s)
     plt.xlabel('x')
     plt.ylabel('s')
     plt.show()
     # </SOL>
```



- **4.2.** Compute the ML estimate

```
[7]: # wML = <FILL IN>
     wML = np.sum(X*s) / np.sum(X*X)

     print("The ML estimate is {}".format(wML))
```

```
The ML estimate is 0.79709787816564
```

- **4.3.** Plot the likelihood as a function of parameter w along the interval $-0.5 \leq w \leq 2$, verifying that the ML estimate takes the maximum value.

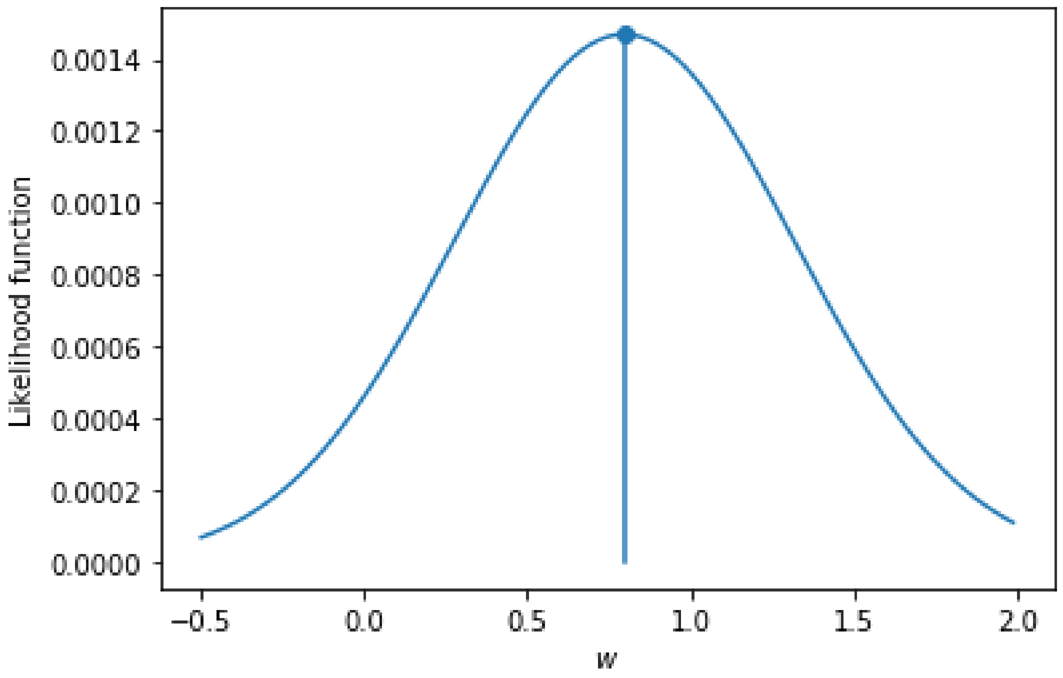
```
[8]: sigma_eps = 1
K = len(s)
wGrid = np.arange(-0.5, 2, 0.01)

p = []
for w in wGrid:
    d = s - X*w
    # p.append(<FILL IN>)
    p.append((1.0/(np.sqrt(2*np.pi)*sigma_eps))**K * np.exp(-np.dot(d, d) /␣
 ↪(2*sigma_eps**2)))

# Compute the likelihood for the ML parameter wML
# d = <FILL IN>
d = s-X*wML
# pML = [<FILL IN>]
pML = [(1.0/(np.sqrt(2*np.pi)*sigma_eps))**K * np.exp(-np.dot(d, d) /␣
 ↪(2*sigma_eps**2))]

# Plot the likelihood function and the optimal value
plt.figure()
plt.plot(wGrid, p)
plt.stem([wML], pML)
plt.xlabel('$w$')
plt.ylabel('Likelihood function')
plt.show()
```

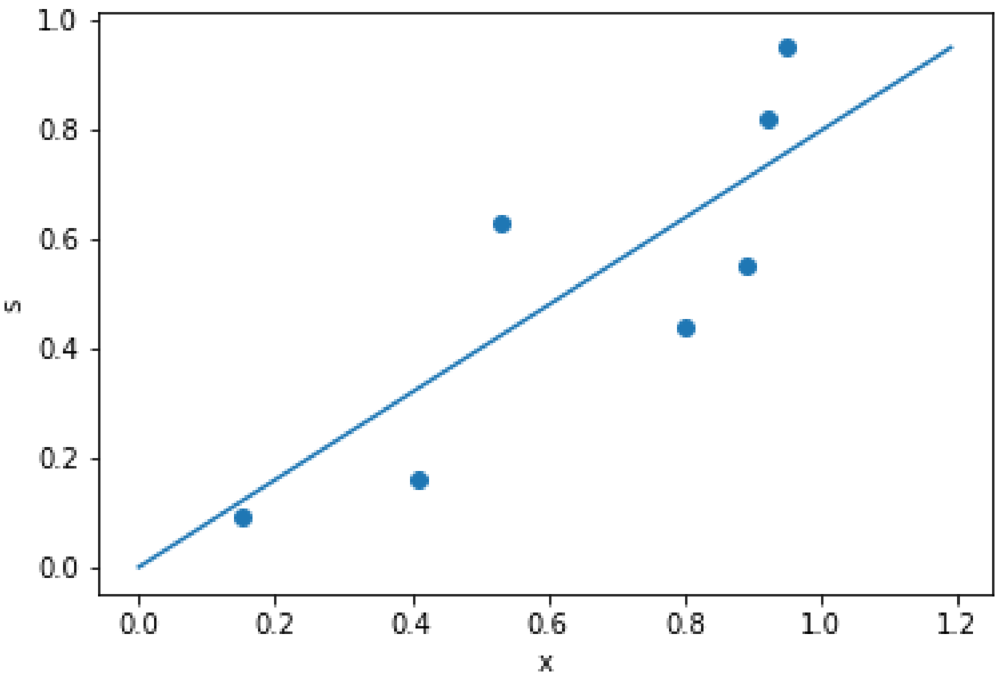
```
/Users/jcid/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:20:
UserWarning: In Matplotlib 3.3 individual lines on a stem plot will be added as
a LineCollection instead of individual lines. This significantly improves the
performance of a stem plot. To remove this warning and switch to the new
behaviour, set the "use_line_collection" keyword argument to True.
```



- **4.4.** Plot the prediction function on top of the data scatter plot

[9]:
```
xgrid = np.arange(0, 1.2, 0.01)
# sML = <FILL IN>
sML = wML * xgrid

plt.figure()
plt.scatter(X, s)
# plt.plot(<FILL IN>)
plt.plot(xgrid, sML)
plt.xlabel('x')
plt.ylabel('s')
plt.axis('tight')
plt.show()
```



1.3.1 2.1. Model assumptions

In order to solve exercise 4 we have taken advantage of the statistical independence of the noise components. Some independence assumptions are required in general to compute the ML estimate in other scenarios.

In order to estimate $\mathbf{w}$ from the training data in a mathematicaly rigorous and compact form let us group the target variables into a vector

$$\mathbf{s} = (s_0, \ldots, s_{K-1})^\top$$

and the input vectors into a matrix

$$\mathbf{X} = (\mathbf{x}_0, \ldots, \mathbf{x}_{K-1})^\top$$

We will make the following assumptions:

- A1. All samples in $\mathcal{D}$ have been generated by the same distribution, $p(\mathbf{x}, s \mid \mathbf{w})$
- A2. Input variables $\mathbf{x}$ do not depend on $\mathbf{w}$. This implies that

$$p(\mathbf{X} \mid \mathbf{w}) = p(\mathbf{X})$$

- A3. Targets $s_0, \ldots, s_{K-1}$ are statistically independent, given $\mathbf{w}$ and the inputs $\mathbf{x}_0, \ldots, \mathbf{x}_{K-1}$, that is:

$$p(\mathbf{s} \mid \mathbf{X}, \mathbf{w}) = \prod_{k=0}^{K-1} p(s_k \mid \mathbf{x}_k, \mathbf{w})$$

Since $\mathcal{D} = (\mathbf{X}, \mathbf{s})$, we can write

$$p(\mathcal{D}|\mathbf{w}) = p(\mathbf{s}, \mathbf{X}|\mathbf{w}) = p(\mathbf{s}|\mathbf{X}, \mathbf{w})p(\mathbf{X}|\mathbf{w})$$

Using assumption A2,

$$p(\mathcal{D}|\mathbf{w}) = p(\mathbf{s}|\mathbf{X}, \mathbf{w})p(\mathbf{X})$$

and, finally, using assumption A3, we can express the estimation problem as the computation of

$$\begin{aligned}
\hat{\mathbf{w}}_{\text{ML}} &= \arg\max_{\mathbf{w}} p(\mathbf{s}|\mathbf{X}, \mathbf{w}) && (1)\\
&= \arg\max_{\mathbf{w}} \prod_{k=0}^{K-1} p(s_k \mid \mathbf{x}_k, \mathbf{w}) && (2)\\
&= \arg\max_{\mathbf{w}} \sum_{k=0}^{K-1} \log p(s_k \mid \mathbf{x}_k, \mathbf{w}) && (3)
\end{aligned}$$

Any of the last three terms can be used to optimize $\mathbf{w}$. The sum in the last term is usually called the **log-likelihood** function, $L(\mathbf{w})$, whereas the product in the previous line is simply referred as the **likelihood** function.

1.3.2 2.2. Summary.

Let's summarize what we need to do in order to design a regression algorithm based on ML estimation:

1. Assume a parametric data model $p(s|\mathbf{x}, \mathbf{w})$
2. Using the data model and the i.i.d. assumption, compute $p(\mathbf{s}|\mathbf{X}, \mathbf{w})$.
3. Find an expression for $\mathbf{w}_{\text{ML}}$
4. Assuming $\mathbf{w} = \mathbf{w}_{\text{ML}}$, compute the MAP or the minimum MSE estimate of s given $\mathbf{x}$.

1.4 3. ML estimation for a Gaussian model.

1.4.1 3.1. Step 1: The Gaussian generative model

Let us assume that the target variables s_k in dataset $\mathcal{D}$ are given by

$$s_k = \mathbf{w}^\top \mathbf{z}_k + \varepsilon_k$$

where $\mathbf{z}_k$ is the result of some transformation of the inputs, $\mathbf{z}_k = T(\mathbf{x}_k)$, and ε_k are i.i.d. instances of a Gaussian random variable with mean zero and varianze σ_ε^2, i.e.,

$$p_E(\varepsilon) = \frac{1}{\sqrt{2\pi}\sigma_\varepsilon} \exp\left(-\frac{\varepsilon^2}{2\sigma_\varepsilon^2}\right)$$

Assuming that the noise variables are independent of $\mathbf{x}$ and $\mathbf{w}$, then, for a given $\mathbf{x}$ and $\mathbf{w}$, the target variable is gaussian with mean $\mathbf{w}^\top \mathbf{z}_k$ and variance σ_ε^2

$$p(s|\mathbf{x}, \mathbf{w}) = p_E(s - \mathbf{w}^\top \mathbf{z}) = \frac{1}{\sqrt{2\pi}\sigma_\varepsilon} \exp\left(-\frac{(s - \mathbf{w}^\top \mathbf{z})^2}{2\sigma_\varepsilon^2}\right)$$

1.4.2 3.2. Step 2: Likelihood function

Now we need to compute the likelihood function $p(\mathbf{s}, \mathbf{X}|\mathbf{w})$. If the samples are i.i.d. we can write

$$p(\mathbf{s}|\mathbf{X}, \mathbf{w}) = \prod_{k=0}^{K-1} p(s_k|\mathbf{x}_k, \mathbf{w}) = \prod_{k=0}^{K-1} \frac{1}{\sqrt{2\pi}\sigma_\varepsilon} \exp\left(-\frac{\left(s_k - \mathbf{w}^\top \mathbf{z}_k\right)^2}{2\sigma_\varepsilon^2}\right) = \left(\frac{1}{\sqrt{2\pi}\sigma_\varepsilon}\right)^K \exp\left(-\sum_{k=1}^{K} \frac{\left(s_k - \mathbf{w}^\top \mathbf{z}_k\right)^2}{2\sigma_\varepsilon^2}\right)$$

Finally, grouping variables $\mathbf{z}_k$ in

$$\mathbf{Z} = (\mathbf{z}_0, \dots, \mathbf{z}_{K-1})^\top$$

we get

$$p(\mathbf{s}|\mathbf{X}, \mathbf{w}) = \left(\frac{1}{\sqrt{2\pi}\sigma_\varepsilon}\right)^K \exp\left(-\frac{1}{2\sigma_\varepsilon^2}\|\mathbf{s} - \mathbf{Z}\mathbf{w}\|^2\right)$$

1.4.3 3.3. Step 3: ML estimation.

The maximum likelihood solution is then given by:

$$\mathbf{w}_{ML} = \arg\max_{\mathbf{w}} p(\mathbf{s}|\mathbf{w}) = \arg\min_{\mathbf{w}} \|\mathbf{s} - \mathbf{Z}\mathbf{w}\|^2$$

Note that $\|\mathbf{s} - \mathbf{Z}\mathbf{w}\|^2$ is the sum or the squared prediction errors (Sum of Squared Errors, SSE) for all samples in the dataset. This is also called the * **Least Squares** * (LS) solution.

The LS solution can be easily computed by differentiation,

$$\nabla_{\mathbf{w}} \|\mathbf{s} - \mathbf{Z}\mathbf{w}\|^2 \Big|_{\mathbf{w}=\mathbf{w}_{\text{ML}}} = -2\mathbf{Z}^\top \mathbf{s} + 2\mathbf{Z}^\top \mathbf{Z}\mathbf{w}_{\text{ML}} = \mathbf{0}$$

and it is equal to

$$\mathbf{w}_{\text{ML}} = (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \mathbf{s}$$

1.4.4 3.4. Step 4: Prediction function.

The last step consists on computing an estimate of s by assuming that the true value of the weight parameters is $\mathbf{w}_{\text{ML}}$. In particular, the minimum MSE estimate is

$$\hat{s}_{\text{MSE}} = \mathbb{E}\{s|\mathbf{x}, \mathbf{w}_{\text{ML}}\}$$

Knowing that, given $\mathbf{x}$ and $\mathbf{w}$, s is normally distributed with mean $\mathbf{w}^\top \mathbf{z}$ we can write

$$\hat{s}_{\text{MSE}} = \mathbf{w}_{\text{ML}}^\top \mathbf{z}$$

Exercise 5: Assume that the targets in the one-dimensional dataset given by

```
[16]: X = np.array([0.15, 0.41, 0.53, 0.80, 0.89, 0.92, 0.95])
      s = np.array([0.09, 0.16, 0.63, 0.44, 0.55, 0.82, 0.95])
```

have been generated by the polynomial Gaussian model

$$s = w_0 + w_1 x + w_2 x^2 + \epsilon$$

(i.e., with $\mathbf{z} = T(x) = (1, x, x^2)^\intercal$) with noise variance

```
[17]: sigma_eps = 0.3
```

- **5.1.** Compute the ML estimate.

```
[18]: # Compute the extended input matrix Z
      nx = len(X)
      # Z = <FILL IN>
      Z = np.hstack((np.ones((nx, 1)), X[:,np.newaxis], X[:,np.newaxis]**2))

      # Compute the ML estimate using linalg.lstsq from Numpy.
      # wML = <FILL IN>
      wML = np.linalg.lstsq(Z, s)[0]
      print(wML)
```

```
[0.01643973 0.50472594 0.32250412]
```

```
/Users/jcid/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8:
FutureWarning: `rcond` parameter will change to the default of machine precision
times ``max(M, N)`` where M and N are the input matrix dimensions.
To use the future default and silence this warning we advise to pass
`rcond=None`, to keep using the old, explicitly pass `rcond=-1`.
```

- **5.2.** Compute the value of the log-likelihood function for $\mathbf{w} = \mathbf{w}_{\mathrm{ML}}$.

```
[19]: K = len(s)

      # Compute the likelihood for the ML parameter wML
      # d = <FILL IN>
      d = s - np.dot(Z, wML)

      # LwML = [<FILL IN>]
      LwML = - K/2*np.log(2*np.pi*sigma_eps**2) - np.dot(d, d) / (2*sigma_eps**2)

      print(LwML)
```

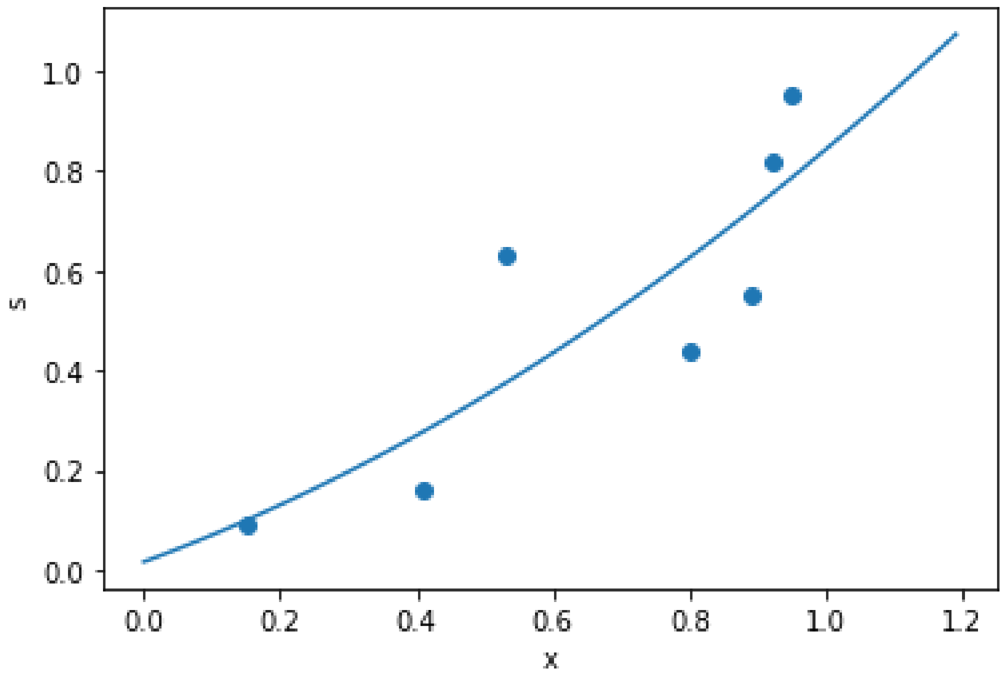
```
1.0272256462076963
```

- **5.3.** Plot the prediction function over the data scatter plot

```
[20]: xgrid = np.arange(0, 1.2, 0.01)
      nx = len(xgrid)
      # Compute the input matrix for the grid data in x
      # Z = <FILL IN>
      Z = np.hstack((np.ones((nx, 1)), xgrid[:,np.newaxis], xgrid[:,np.newaxis]**2))

      # sML = <FILL IN>
      sML = np.dot(Z, wML)

      plt.figure()
      plt.scatter(X, s)
      # plt.plot(<FILL IN>)
      plt.plot(xgrid, sML)
      plt.xlabel('x')
      plt.ylabel('s')
      plt.axis('tight')
      plt.show()
```



Exercise 6: Assume the dataset $\mathcal{D} = \{(x_k, s_k, k = 0, \ldots, K-1\}$ contains i.i.d. samples from a distribution with posterior density given by

$$p(s \mid x, w) = wx \exp(-wxs), \qquad s \geq 0,\ x \geq 0,\ w \geq 0$$

- **6.1.** Determine an expression for the likelihood function

Solution: The likelihood function is

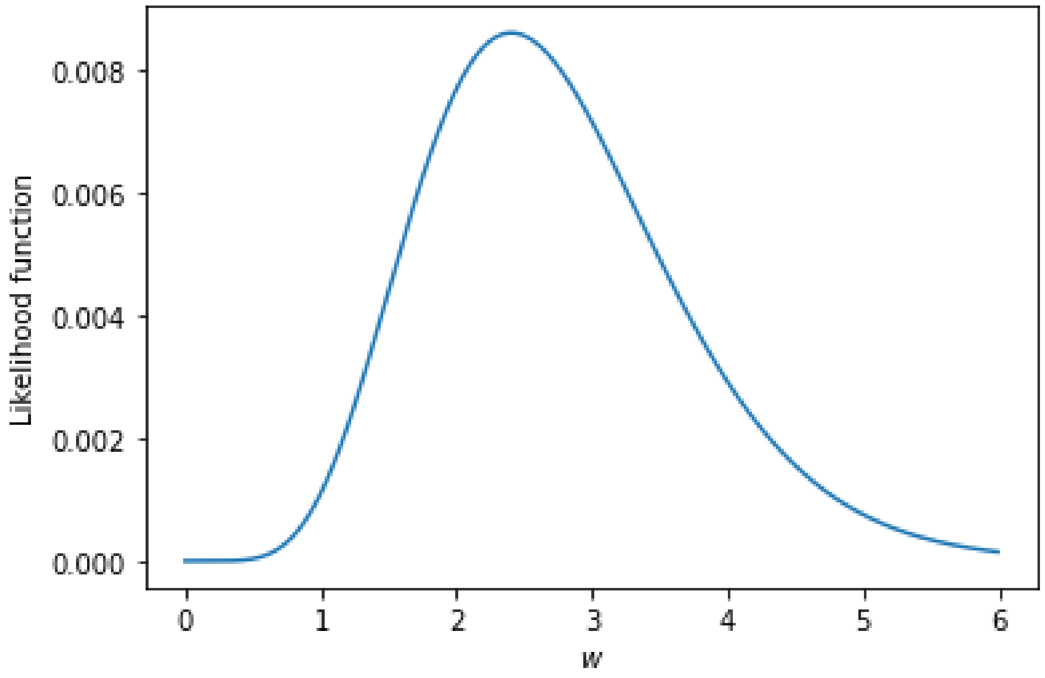
$$p(\mathbf{s}|w, \mathbf{X}) = \prod_{k=0}^{K-1} w x_k \exp(-w x_k s_k) = w^K \left(\prod_{k=0}^{K-1} x_k\right) \exp\left(-w \sum_{k=0}^{K-1} x_k s_k\right)$$

- **6.2.** Draw the likelihood function for the dataset in **Exercise 4** in the range $0 \leq w \leq 6$.

[22]:
```python
K = len(s)
wGrid = np.arange(0, 6, 0.01)

p = []
Px = np.prod(X)
xs = np.dot(X,s)
for w in wGrid:
    # p.append(<FILL IN>)
    p.append((w**K)*Px*np.exp(-w*xs))

plt.figure()
# plt.plot(<FILL IN>)
plt.plot(wGrid, p)
plt.xlabel('$w$')
plt.ylabel('Likelihood function')
plt.show()
```



- **6.3.** Determine the maximum likelihood coefficient, w_{ML}.

(*Hint: you can maximize the log-likelihood function instead of the likelihood function in order to simplify the differentiation*)

Solution:

Applyng the logarithm to the likelihood function we get

$$\log p(\mathbf{s}|\mathbf{w}, \mathbf{X}) = K \log w + \sum_{k=0}^{K-1} \log(x_k) - w\mathbf{X}^\top \mathbf{s}$$

which is minimum for

$$w_{\mathrm{ML}} = \frac{K}{\mathbf{X}^\top \mathbf{s}}$$

- **6.4.** Compute w_{ML} for the dataset in **Exercise 4**

```
[24]: # wML = <FILL IN>
wML = np.float(K) /xs
print(wML)
```

```
2.4043415538915984
```

- **6.5.** Assuming $w = w_{\mathrm{ML}}$, compute the prediction function based on the estimate s_{MSE}

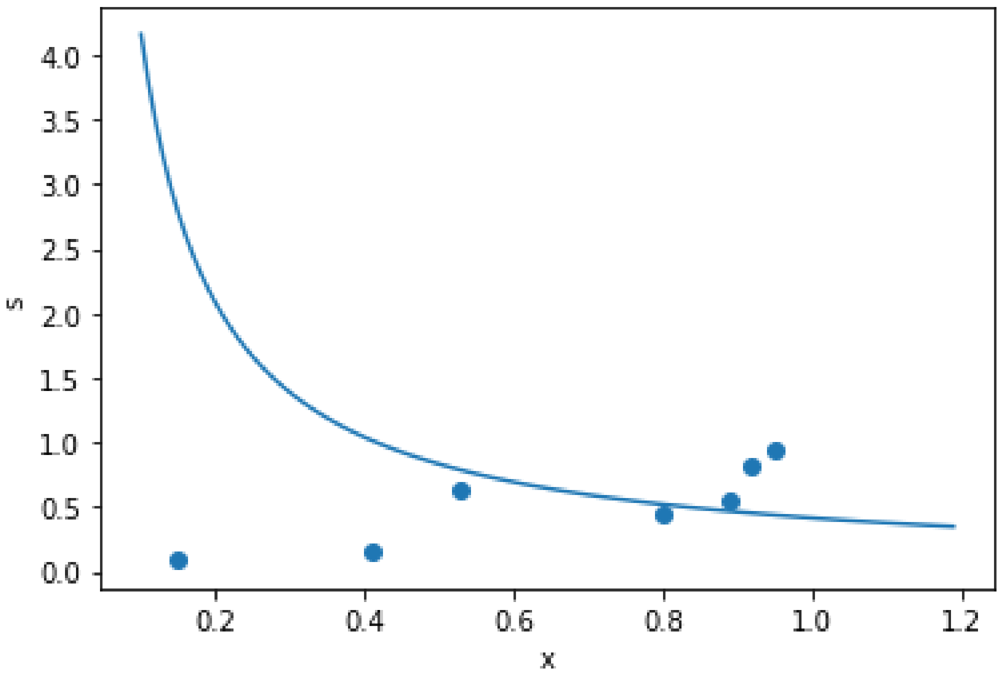
Solution:

$$\hat{s}_{\mathrm{MSE}} = \mathbb{E}\{s|x, w\} = \int swx \exp(w_{\mathrm{ML}} xs) ds = \frac{1}{w_{\mathrm{ML}} x}$$

- **6.6.** Plot the prediction function obtained in ap. 6.5, and compare it with the linear predictor in exercise 4

```
[25]: xgrid = np.arange(0.1, 1.2, 0.01)
# sML = <FILL IN>
sML = 1 / (wML * xgrid)

plt.figure()
plt.scatter(X, s)
# plt.plot(<FILL IN>)
plt.plot(xgrid, sML)
plt.xlabel('x')
plt.ylabel('s')
plt.axis('tight')
plt.show()
```



Subjectively, we can see that the predictor computed in exercise 6 does not fit the given data very well. This could be a false perception. If the data have been truly generated by the parametric model assumed in exercise 6 (i.e. $p(s \mid x, w) = wx\exp(-wxs)$, the apparent missbehavior of the estimator could be caused by the natural randomness of the data, and a greater amount of data would show a better adjustement.

Alternative, it may be the case the model assumed in sec. 6 is incorrect. Again, more data would be useful to asses that.

This shows that the choice of the data model is important. In many applications, no parametric data model is available, and the data scientist must make a choice based on the nature of the data or any previous knowledge about the statistical behavior of the data.

If no previous information is available, the data scientist can try different models, and compare using validation data and some cross validation technique.