1 Regression and maximum likelihood

We have previously introduced **supervised learning** where we propose a model in form of a parameterized function and where we then learned the model parameters from example data. We now through the same process again with the addition that we take uncertainty into account.

A note of caution: I have recently added some parts to this manuscript and also changed notations. Thus, let me know if you spot inconstistencies or if things are not clear.

1.1 Trends in stochastic data

In supervised learning, examples of input-output relations are given and our goal is to discover the relations between these data so that we can make **predictions** of previously unseen data. Let's consider here an example from robotics where we want to model how a far a terrestrial robot is moving when the robot moves for a given number of seconds when activating the motors with certain power.

More formally, the **training data** that are denoted by the inputs $\mathbf{x}$, such as motor power or the number of seconds we let it run, and the outputs or labels y, such as the distance that the robot traveled. This distance we (the teacher) has to provide such as from using a ruler to measure the actual distance or use a distance measure to estimate in a more automated way. In the following we consider m training examples, the pairs of values

$$\{(\mathbf{x}^{(i)}, y^{(i)}); i = 1...m\}. \tag{1.1}$$

The above notation describes mapping examples from an n-dimensional feature space to a 1-dimensional output space, hence the vector notation for $\mathbf{x}$ and the scalar notation for y. We consider here a 1-dimensional output space for our discussions mainly for convenience. Generalization to higher-dimensional are straight forward. Note that an index with brackets, (i), is used to label different training examples.

As an example, let us consider again how the tribot moves when both motors are driven for a certain amount of time. To automate the collection of data we can use the ultrasonic sensor to measure the distance to a wall while driving the tribot for different amount of time forward and backward. In Figure 1.2A we show several measurements of the distance traveled.

The data clearly reveal some systematic relation between the time of running the motor and the distance traveled, the general trend being that the traveled distance increases with increasing running time of the motors. While there seems to be some noise in the data, the outliers and the noise can not hide a linear trent for most of the data. This hypothesis can be quantified as a parameterized function,

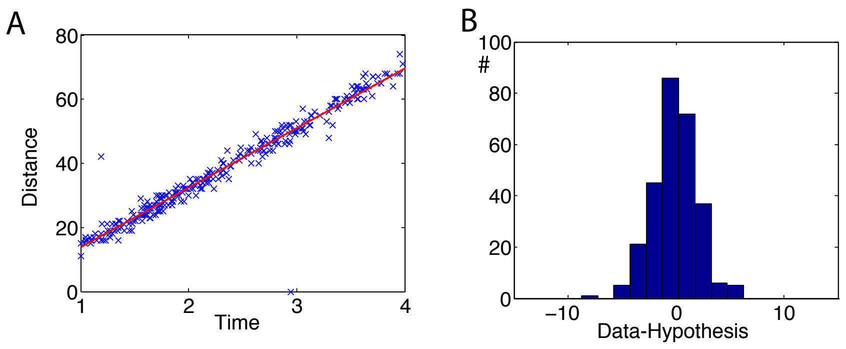

Fig. 1.1 (A) Measurements of distance travelled by a robot when running the motor for different number of milliseconds with a given power. (B) Corresponding histogram of differences between data and hypothesis.

$$\hat{y}(x; \mathbf{w}) = w_0 + w_1 x. \tag{1.2}$$

This notation means that the hypothesis $\hat{y}$ is a function of the quantity x, and the hypothesis includes all possible straight lines, where each line can have a different offset w_0 (intercept with the y-axis), and slope w_1.

We typically collect parameters in a **parameter vector** $\mathbf{w}$. We only considered one input variable x above, but we can easily generalize this to higher dimensional problems where more input **attributes** are given. For example, we could not only vary the time the motor is running, let us label this attribute now with x_1, but also push the robot forward by hand for a certain time, labeled with x_2. As the effects of these manipulations are independent on the results, we can independently add the effects of higher dimensions to our hypothesis. To compress our notations further we also introduce here the convention that we consider a constant input, $x_0 = 1$ as the first component of the input vector, so that the corresponding parameter encodes the offset of the function. The state vector can then be written as,

$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}. \tag{1.3}$$

With this convention we can write the hypothesis as

$$\hat{y}(\mathbf{x}; \mathbf{w}) = w_0 x_0 + w_1 x_1 + w_2 x_2. \tag{1.4}$$

It is then even easy to write a linear hypothesis with n attributes as

$$\hat{y}(\mathbf{x}; \mathbf{w}) = w_0 x_0 + ... + w_n x_n = \sum_i w_i x_i = \mathbf{w}^T \mathbf{x}, \tag{1.5}$$

where the superscript T indicates the transpose of a vector.

Another factor that influences the distance traveled is the power setting of the motor. Of course, the distance traveled within a certain time does depend on the power and is not just an independent additive effect on the travelled distance. Results of the experiment for different power settings and different travel times are show in Figure **??**. Figure **??**A also includes a fit to equation 1.4. However, these data are better described by a bilinear hypothesis,

$$\hat{y}(\mathbf{x}; \mathbf{w}) = w_0 x_0 + w_1 x_1 x_2. \tag{1.6}$$

The corresponding fit of the same data is shown in Figure **??**B.How to perform these fits is discussed further below.

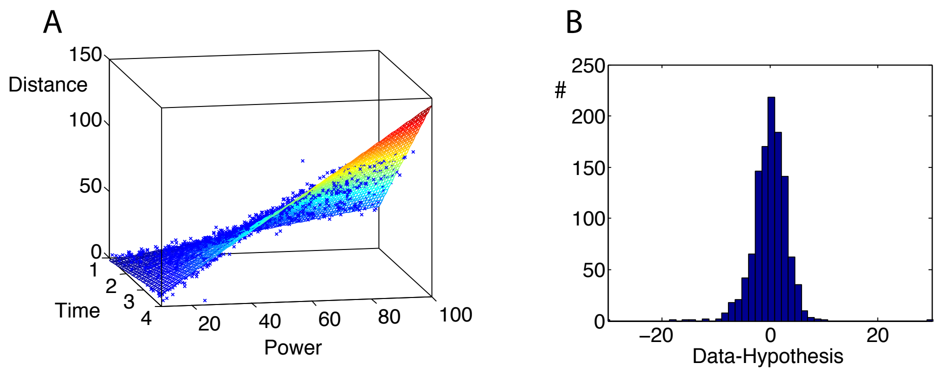

Fig. 1.2 (A) Measurements of distance travelled by the tribot when running the motor for different number of milliseconds and various power settings. Fit according to equation 1.4 (B) Same as (A) with fit according to equation 1.6. (C,D) Corresponding histogram of differences between data and hypothesis.

While we had to make a good guess for the functional form of the trend in the data, the actual parameters have so far not been specified. Thus, we made a **hypothesis** in the form of a parameterized function, $h(\mathbf{x}; \mathbf{w})$, and the learning part boils down to determining appropriate values for the parameters from the sample data. After learning these parameters we can then use this function to predict specific reactions of a plant even for motor commands for which no training examples were given. The remaining question is how we find appropriate values for the parameters. However, before we do this we need to be more faithful to the data and acknowledge fluctuation around our initial hypothesis.

1.2 Probabilistic models and maximum likelihood

So far, we have only modelled the trend of the data, and we should investigate more the fluctuations around this trend. Of course, we expect several sources of noise such as the accuracy of the ultrasonic sensor and the tendency of the tribot to sometimes turn due to wheel slippage. Indeed, while gathering these data we have fixed the follower wheel to minimize turning when moving forward and backward. We also started new trials when the robot went too much off track. Thus, we already try to minimize error due to a careful setup of the experiment to gather the data. However, what we did not tell you is that we run the experiment in Figure 1.2A with different powers of the motor between 40 and 60. Such hidden information can also contribute to uncertainties in the environment. Data for doing the same experiment as before but with a fixed motor power of 50 is shown in Figure 1.2B. These data vary less but are still noisy due other sources of uncertainty.

Figures 1.2B and **??**B are plots of the histogram of the differences between the actual data and the hypothesis regression line. The histograms look a bit Gaussian, which according to the central limit theorem is a likely finding for additive and independent noise sources. In any case, we should revise our hypothesis by acknowledging the stochastic nature of the data and writing a down a specific functional form of a conditional density function for the quantity y given some input values $\mathbf{x}$. Similar to before, we also allow this probabilistic hypothesis to depend on some parameters,

$$p(\hat{y}|\mathbf{x};\mathbf{w}). \tag{1.7}$$

For our specific example of the tribot we assume here that the data follow our previous deterministic hypothesis $\hat{y}(\mathbf{x};\mathbf{w})$ with **additive Gaussian noise**, or with other words, that the data in Figure 1.2B are Gaussian distributed with a mean $\mu = \hat{y}(x)$ depends linearly on the value of x,

$$\begin{aligned} p(\hat{y}|x;\mathbf{w},\sigma) &= N(\mu = \mathbf{w}^T\mathbf{x}, \sigma) && (1.8) \\ &= \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\hat{y}-\mathbf{w}^T\mathbf{x})^2}{2\sigma^2}\right) && (1.9) \end{aligned}$$

This functions specifies the probability of values for $\hat{y}$, given an input x and the parameters $\mathbf{w}$ and σ. In the following we keep the parameter σ so that we only consider the variables $\mathbf{w}$ as free. This just helps to keep the formulas manageable, though including it should be straight forward.

Specifying a model with a density function is an important step in modern modelling and machine learning. In this type of thinking, we treat data from the outset as fundamentally stochastic, that is, data can be different even in situations that we deem identical. This randomness may come from an **irreducible indeterminacy**, that is, true randomness in the world that can not be penetrated by further knowledge, or this noise might represent **epistemological limitations** such as the lack of knowledge of hidden processes or limitations in observing states directly. The only important fact for us is that we have to live with these limitations. This acknowledgement together with the corresponding language of probability theory has helped to make large progress in the machine learning area.

We will now turn to the important principle that will guide our learning process which corresponds here to estimating the parameters of the model. While the parameterized hypothesis so far describes the form of the data, we need to estimate values for the parameters to make real predictions. We therefore consider now the examples for the input-output pairs, our training set $\{(x^{(i)}, y^{(i)}); i = 1...m\}$. The important principle that we will now follow is to choose the parameters so that the examples we have are most likely. This is called **maximum likelihood estimation**. To formalize this principle, we need to think about how to combine probabilities for several observations. If the observations are independent, then the joint probability of several observations is the product of the individual probabilities,

$$p(y_1, y_2, ..., y_m|x_1, x_2, ..., x_m; \mathbf{w}) = \Pi_i^m p(y_i|x_i; \mathbf{w}). \tag{1.10}$$

Note that y_i are still random variables at this point. But we now use our training examples as specific observations for each of these random variables, and introduce the **Likelihood function**

$$L(\mathbf{w}) = \Pi_i^m h(\mathbf{w}; y^{(i)}, x^{(i)}). \tag{1.11}$$

The h on the right hand side is now not a density function, but it is a regular function (with the same form as our parameterized hypothesis) of the parameters $\mathbf{w}$ for the given values $y^{(i)}$ and $x^{(i)}$. Instead of evaluating this large product, it is common to use the logarithm of the likelihood function, so that we can use the sum over the training examples,

$$l(\mathbf{w}) = \log L(\mathbf{w}) = \sum_i^m \log(h(\mathbf{w}; y^{(i)}, x^{(i)})). \tag{1.12}$$

Since the log function increases monotonically, the maximum of L is also the maximum of l. The maximum (log-)likelihood can thus be calculated from the examples as

$$\mathbf{w}^{\text{MLE}} = arg \max_{\mathbf{w}} l(\mathbf{w}). \tag{1.13}$$

We might be able to calculate this analytically or use one of the search algorithms to find a maximum from this function.

Let us apply this to the linear regression discussed above. The log-likelihood function for this example is

$$l(\mathbf{w}) = \log \Pi_{i=1}^m \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(\mathbf{i})})^2}{2\sigma^2}\right) \tag{1.14}$$

$$= \sum_{i=1}^m \left(\log \frac{1}{\sqrt{2\pi}\sigma} - \frac{(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(\mathbf{i})})^2}{2\sigma^2}\right) \tag{1.15}$$

$$= -\frac{m}{2}\log 2\pi\sigma - \sum_{i=1}^m \frac{(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2}{2\sigma^2}. \tag{1.16}$$

Thus, the log was chosen so that we can use the sum in the estimate instead of dealing with big numbers based on the product of the examples.

Let us now consider the special case in which we assume that the constant σ, the variance of the data, is the same for all x and thus has a fixed value given to us. We can thus concentrate on the estimation of the other parameters w. Since the first term in the expression 1.16, $-\frac{m}{2}\log 2\pi\sigma$, is independent of $\mathbf{w}$, maximizing the log-likelihood function is equivalent to minimizing a quadratic error term

$$E = \frac{1}{2}(y - h((x; (w))^2 \Longleftrightarrow p(y|\mathbf{x}; \mathbf{w}) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{((y - h((x; \mathbf{w})^2}{2}) \tag{1.17}$$

This **error function** or **cost function** was a frequently used criteria called **Least Mean Square (LSM)** regression for parameters estimation when considering deterministic hypothesis. I terms of our probabilistic view, the LSM regression is equivalent to MLE for gaussian data with constant variance. When the variance is a free parameter, then we need to minimize equation 1.16. instead.

We have discussed Gaussian distributed data in most of this section, but one can similarly find corresponding error functions for other distributions. For example, a **polynomial error function** correspond more generally to a density model of the form

$$E = \frac{1}{p}||y - h((x;(w)||^p \Longleftrightarrow p(y|\mathbf{x};\mathbf{w}) = \frac{1}{2\Gamma(1/p)} \exp(-||y - h((x;\mathbf{w}||^p). \quad (1.18)$$

Later we will specifically discuss and use the **ϵ-insensitive error function**, where errors less than a constant ϵ do not contribute to the error measure, only errors above this value,

$$E = ||y - h((x;(w)||_\epsilon \Longleftrightarrow p(y|\mathbf{x};\mathbf{w}) = \frac{p}{2(1-\epsilon)} \exp(-||y - h((x;\mathbf{w}||_\epsilon). \quad (1.19)$$

Since we already acknowledged that we do expect that data are noisy, it is somewhat logical to not count some deviations form the expectation as errors. It also turns out that this error function is much more robust than other error functions.

1.3 Maximum a posteriori estimates

In the maximum likelihood estimation we assumed that we have no prior knowledge of the parameters $\mathbf{w}$. However, we sometimes might know which values of the parameters are impossible or less likely. This prior knowledge can be summarized in the prior distribution $p(\mathbf{w})$, and the next question is how to combine this prior knowledge in the maximum likelihood scheme. Combining prior knowledge with some evidence is described by Bayes' theorem. Thus, let us consider again that we have some observations (x, y) from specific realizations of the parameters, which is given by $(p(x, y|\mathbf{w})$, and the prior about the possible values of the parameters, given by $p(\mathbf{w})$. The prior is in this situation sometimes called the **regularizer**, restricting possible values in a specific domain. We want to know the distribution of parameters given the observation, $p(\mathbf{w}|x, y)$, which can be calculated from Bayes's theorem,

$$p(\mathbf{w}|x, y) = \frac{p(x, y|\mathbf{w})p(\mathbf{w})}{\int_{w' \in W} p(x, y|\mathbf{w}')p(\mathbf{w}')\mathrm{d}w'}, \quad (1.20)$$

where w is the domain of the possible parameter values. We can now use this expression to estimate the most likely values for the parameters. For this we should notice that the denominator, which is called the **partition function**, does not depend on the parameters $\mathbf{w}$. The most likely values for the parameters can thus be calculated without this term and is given by the **maximum a posteriori (MAP)** estimate,

$$\mathbf{w}^{\mathrm{MAP}} = arg \max_{\mathbf{w}} p(x, y|\mathbf{w})p(\mathbf{w}). \quad (1.21)$$

This is, in a Bayesian sense, the most likely value for the parameters, where, of course, we now treat the probability function as a function of the parameters (e.g., a likelihood function).

A final caution: ML and MAP estimates give us a **point estimate**, a single answer of the most likely values of the parameters. This is often useful as a first guess and is commonly used to **make decisions** about which actions to take. However, it is possible that other sets of parameters values might have only a little smaller likelihood value, and should therefore also be considered. Thus, one limit of the estimation methods discussed here is that they do not take distribution of answers into account, which is more common in more advanced Bayesian methods.

1.4 Cross-Entropy Loss for sigmoidal classification

In this chapter we have embraced the stochastic nature of the problem by building specific parameterized probabilistic models. But what if we don't know the functional form of the underlying probabilistic nature. Bayesian people would say that this must be suboptimal, but we should relate this to our previous approach of general learning machine which somewhat tries to compensate for the unknown by building large and encompassing functional models. In particular we have studied deep networks with a large number of parameters.

For a given set of parameters, such a model gives us a specific response to each possible input. In order to make such a model stochastic we need to introduce some stochastic component into the model. There are different types of generalizations of such models. For example, we could include some noise into the response of each node of the network or include stochasticities in the parameters (weight values) of the model. In the past, different versions of stochastic gain functions have been used, and drop-out, which sets the response of a node to zero in random trials is another more recent example. With such stochastic modifications we get a model that represents a density function $p(\hat{y}|x)$.

Now we need a measure how good this model is in comparison to the true nature of data that are described by the unknown density function $q(y|x)$. One specific measure of the distance or divergence between two probability distributions is given by the **Cross-entropy**

$$H(p,q) = -\sum_x p(x)\log q(x) \tag{1.22}$$

In other words, this is the negative log probability of the given labels under the current model. Since we want to maximize the probability of the data under the model, we want to minimize the cross entropy. The cross entropy is related to the **KL-divergence** by

$$H(p,q) = H(p) + KL(p||q), \tag{1.23}$$

and since changing model parameters do not effect the true data, minimizing the cross entropy is equivalent to minimizing the KL-divergence.

Let us apply this to binary classification model which is described by Bernoulli variables that take the value 0 or 1. For this density function, the cross entropy is given by

$$H(p,q) = -p(x=0)\log q(x=0) + -p(x=1)\log q(x=1)$$

$$= -p(x)\log q(x) - (1-p(x))\log(1-q(x))$$

where $p(x)$ is shorthand for $p(x=1)$. The natural way for a neural networks to represent a Bernoulli variable is with a sigmoid output.

$$p(\hat{y}|x;w) = \frac{1}{1+e^{-\mathbf{x}\mathbf{w}}} \tag{1.24}$$

$$1-p(\hat{y}|x;w) = \frac{1+e^{-\mathbf{x}\mathbf{w}}-1}{1+e^{-\mathbf{x}\mathbf{w}}} = \frac{1}{1+e^{\mathbf{x}\mathbf{w}}} \tag{1.25}$$

$$\log p(\hat{y}|x;w) = -\log(1+e^{-\mathbf{x}\mathbf{w}}) \tag{1.26}$$

$$\tag{1.27}$$

Note that the output of the network is the probability of the label being y=1 and not directly the label.

Maximizing the log-probability of the observed data is given by minimizing the cross-entropy between the network output, $p(\hat{y})$, and the given labels, y, or mathematically minimizing the Loss function

$$L = -y\log p(\hat{y}) - (1-y)\log(1-p(\hat{y})) \tag{1.28}$$

$$= y\log(1+e^{-\mathbf{x}\mathbf{w}}) + (1-y)\log(1+e^{\mathbf{x}\mathbf{w}}) \tag{1.29}$$

$$\frac{dL}{d(\mathbf{x}\mathbf{W})} = y\frac{-e^{-\mathbf{x}\mathbf{w}}}{1+e^{-\mathbf{x}\mathbf{w}}} + (1-y)\frac{e^{\mathbf{x}\mathbf{w}}}{1+e^{\mathbf{x}\mathbf{w}}} \tag{1.30}$$

$$= -y\frac{1}{1+e^{\mathbf{x}\mathbf{w}}} + (1-y)\frac{1}{1+e^{-\mathbf{x}\mathbf{w}}} \tag{1.31}$$

$$= -y(1-p(\hat{y})) + (1-y)p(\hat{y}) \tag{1.32}$$

$$= -y + yp(\hat{y}) + p(\hat{y}) - yp(\hat{y}) \tag{1.33}$$

$$= p(\hat{y}) - y \tag{1.34}$$

For multi-class problems, the equivalent of the sigmoid is the **softmax function**

$$p(\hat{y}=i|x) = \frac{e^{\mathbf{x}\mathbf{W}_i}}{\sum_{j=1}^{N} e^{\mathbf{x}\mathbf{W}_j}}, \tag{1.35}$$

where N is the number of classes. You can easily see the equivalence to the sigmoid in the case of having two classes where one of them has input 0,

$$p(\hat{y}=0) = \frac{e^0}{e^0+e^{\mathbf{x}\mathbf{w}}} = \frac{1}{1+e^{\mathbf{x}\mathbf{w}}} = 1 - \frac{1}{1+e^{-\mathbf{x}\mathbf{w}}}$$

$$p(\hat{y}=1) = \frac{e^{\mathbf{x}\mathbf{w}}}{e^0+e^{\mathbf{x}\mathbf{w}}} = \frac{e^{\mathbf{x}\mathbf{w}}}{1+e^{\mathbf{x}\mathbf{w}}} = \frac{1}{1+e^{-\mathbf{x}\mathbf{w}}}$$

The derivation of the gradient for this multi class case works out to the same as the binary classification,

$$\frac{dL}{d(\mathbf{x}\mathbf{W})} = p(\hat{\mathbf{y}}) - \mathbf{y} \tag{1.36}$$

$$\frac{dL}{d\mathbf{x}} = (p(\hat{\mathbf{y}}) - \mathbf{y})\mathbf{W}^T \tag{1.37}$$

$$\frac{dL}{d\mathbf{W}} = \mathbf{x}^T(p(\hat{\mathbf{y}}) - \mathbf{y}) \tag{1.38}$$