5 Probabilistic regression and classification

5.1 Probabilistic models

We have discussed a linear model and linear regression in chapter 3, and we will now revise this method to include stochastic data. This will show how modern probabilistic machine learning can be formulated. We will follow a simple stochastic generalization of the linear regression example to introduce the formalism. Later chapters will generalize this idea to non-linear problems in higher dimensions.

We will again consider supervised learning where examples of input-output relations are given and our goal is to make a model that can make predictions of previously unseen data. Let's consider here an example from robotics where we want to model how far a terrestrial robot is moving when the wheels are turning for a given number of seconds after activating the corresponding motors with a certain power. Figure 5.1A shows a Lego Mindstorm robot that has two motorized wheels and an ultrasonic distance motor attached to it. We want to model (or predict) how far this robot moves when both motors are driven for a certain amount of time.

The i-th training data are denoted by the pairs $(x^{(i)}, y^{(i)})$, where the feature inputs $x^{(i)}$ is the time in seconds we let both motors run, and the outputs or labels y is the distance that the robot traveled. This true distance traveled has to be provided by the teacher, likely in form of measurements such as from using a ruler or as sensor that measures distance such as a laser range finder or and ultrasonic sensor. To automate the collection of data we use an ultrasonic sensor to measure the distance to a wall while driving the robot for different amount of time forward and backward. The ultrasonic data are now the teacher feedback, and the teacher's data are considered as the ground truth and will not be questioned here. The point is that we later do not need the teacher (ultrasonic sensor) to predict how far the robots travels.

Figure 5.1B shows several measurements of the distance traveled for different times the motors are activated. 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 trend for most of the data. Let us therefore do again a linear regression as discussed in Chapter 3 with the model,

$$\hat{y}(x; \mathbf{w}) = w_0 + w_1 x. \tag{5.1}$$

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, 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

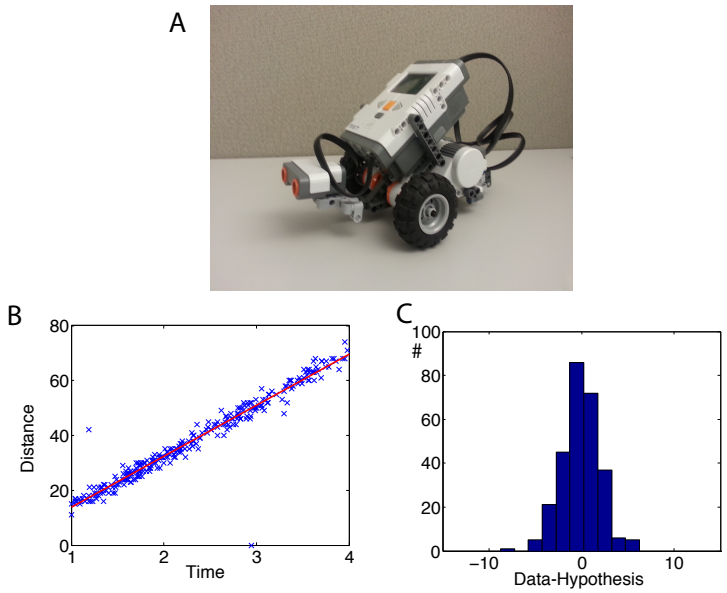

Fig. 5.1 (A) A terrestrial robot build with Lego Mindstorm with an ultrasonic sensor at the front. (B) Measurements of distance travelled by a robot when running the motor for different number of milliseconds with a given power. (C) Corresponding histogram of differences between data and a line that is fitted by minimizing the the mean square error between the data points and the line.

it 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 5.2. However, these data are better described by a bilinear hypothesis,

$$\hat{y}(\mathbf{x};\mathbf{w}) = w_0 + w_1 x_1 x_2. \tag{5.2}$$

The corresponding fit of the data is also shown in Figure **??**A. Sometimes we do not know all the factors. For example, we might not be given the power settings of the motors in this experiment so that the data look as shown in Figure **??**C. These data look more noisy than the previous data. although we know that this is not really noise but rather unknown factors. The point of including this example is basically that it does not really matter if 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 uncertainty has helped to make large progress in the machine learning area.

We have so far basically ignored the fluctuations in the data with the functional regression procedure, and we now investigate more the fluctuations around this trend. Figures 5.1C and 5.2B and D are plots of the histogram of the differences between the actual data and the (linear) functional hypothesis of the above regressions. The

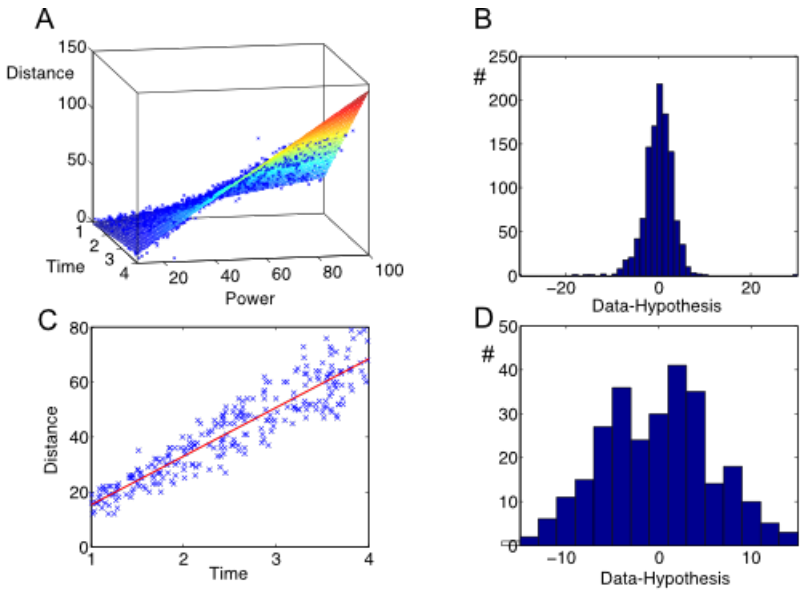

Fig. 5.2 (A) Measurements of distance travelled by the robot when running the motor for different number of milliseconds and various power settings. Fit according to equation 5.2 (B) Histogram of differences between data and hypothesis. (C) The same data as shown in (A) when collapsed across the power setting. The corresponding distribution shown in (D) is much wider than the histogram shown in Figure 5.1.

histograms look a bit like a Gaussian distribution, 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 $\mathbf{w}$,

$$p(\hat{y}, \mathbf{x}|\boldsymbol{w}) = p(\hat{y}|\mathbf{x}; \boldsymbol{w})p(\mathbf{x}). \tag{5.3}$$

We assume that the feature values are equally likely so that $p(\mathbf{x})$ is constant and we can concentrate on the first factor. As in the deterministic case we have to come up with a specific function, a density function with parameters $\mathbf{w}$. Guided by the form of the "noise" in the above robot example, 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 5.1C are Gaussian distributed with a mean $\mu = \hat{y}(x)$ that depends linearly on the value of x,

$$p(\hat{y}|x; \mathbf{w}, \sigma) = N(\mu = \mathbf{w}^T\mathbf{x}, \sigma) \tag{5.4}$$

$$= \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\hat{y} - \mathbf{w}^T\mathbf{x})^2}{2\sigma^2}\right) \tag{5.5}$$

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 at a specific value so

that we only consider the variables $\mathbf{w}$ as free. This just helps to keep the formulas manageable for illustration purposes, though including it is 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. We have just specified a specific probabilistic model for the example robot data.

5.2 Learning in probabilistic models: Maximum likelihood estimate

We will now turn to the important principle that will guide our learning process. Learning means of course to determine the parameters of the model from example data. Here we introduce the important **maximum likelihood principle**. This principle states that we choose the parameters in a probabilistic model in the following way:

> Given a parameterize hypothesis function $p(y|\mathbf{x};\mathbf{w})$, we will chose as parameters the values which make the data y most likely under this assumption.

Let illustrate this on the Gaussian example above with the parameterized hypothesis eq.5.5. We are considering the one dimensional case with one feature value x. Given parameters w_0, w_1, and σ, and the feature value $x^{(1)}$, then the probability

$$p(Y_1 = y^{(1)}|x^{(1)}, w_0, w_1, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(1)} - w_0 - w_1 x^{(1)})^2}{2\sigma^2}\right). \quad (5.6)$$

No we invoke the maximum likelihood principle and ask which parameters we should choose in order to maximize the probability that$y = y^{(1)}$. This can be achieved by choosing values

$$\sigma = \text{arbitrary} \quad (5.7)$$
$$w_1 = \text{arbitrary} \quad (5.8)$$
$$w_0 = y^{(1)} - w_1 x^{(1)} \quad (5.9)$$

This is our first **maximum likelihood estimate (MLE)** of the parameters w_0, w_1, σ. Of course, this is a whole manifold of possible values and it is clear that we need more than one data point to make better predictions. In order to do this we need to know the probability of a combination of data points. For example, lets say we have two data points $(x^{(1)}, y^{(1)})$ and $(x^{(2)}, y^{(2)})$. We then need to know the joined distribution of having the values for the two y's. We have not specified a model for this until now, but we will now make the assumption that the data points are conditionally independent, that is

$$\begin{aligned} p(Y_1 = y^{(1)}, Y_2 = y^{(2)}|x^{(1)}, x^{(2)}, w_0, w_1, \sigma) &= ... \\ .... \; p(Y_1 = y^{(1)}|x^{(1)}, x^{(2)}, w_0, w_1, \sigma) &\quad p(Y_2 = y^{(2)}|x^{(1)}, x^{(2)}) \end{aligned} \quad (5.10)$$

In general, if we have m samples, and if we assume that the observation are all conditionally independent, the joint probability of several observations is the product of the individual probabilities,

$$p(Y_1 = y^{(1)}, Y_2 = y^{(2)}, ..., Y_m = y^{(1)}|\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m; \boldsymbol{w}) = \Pi_i^m p(y_i|\mathbf{x}_i; \boldsymbol{w}). \tag{5.11}$$

We have written this formula again for a general case where all the parameters are collected in the vector $\mathbf{w}$, and y_i is a short form of $Y_i = y^{(i)}$. We can now insert the specific observations (training data) into the resulting function. We end then up with a function that is a function of the parameters. This is equivalent to the function eq.5.6, but now for m observations. This function is called the **Likelihood function**

$$L(\boldsymbol{w}) = \Pi_i^m p(\boldsymbol{w}; y^{(i)}, \mathbf{x}^{(i)}). \tag{5.12}$$

Note that this is now a regular function of the model parameters, not a density function. Also, since this function it is not a probability density function, we correctly replaced the notation of the vertical bar "|" with the semicolon ";". We are nearly done, but 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,

$$l(\boldsymbol{w}) = \log L(\boldsymbol{w}) = \sum_i^m \log(h(\boldsymbol{w}; y^{(i)}, \mathbf{x}^{(i)})). \tag{5.13}$$

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

$$\boldsymbol{w}^{\mathrm{MLE}} = \mathrm{argmax}_{\boldsymbol{w}} l(\boldsymbol{w}). \tag{5.14}$$

Of course we still have to find the maximum in practice. We might be able to calculate this analytically or use one of the search algorithms to find a maximum from this function like a gradient ascent.

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

$$l(\mathbf{w}, \sigma) = \log \Pi_{i=1}^m \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2}{2\sigma^2}\right) \tag{5.15}$$

$$= \sum_{i=1}^m \left(\log \frac{1}{\sqrt{2\pi}\sigma} - \frac{(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2}{2\sigma^2}\right) \tag{5.16}$$

$$= -\frac{m}{2}\log 2\pi\sigma - \sum_{i=1}^m \frac{(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2}{2\sigma^2}. \tag{5.17}$$

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 the parameter σ is given. We can thus concentrate on the estimation of the other parameters w. Since the first term in the expression 5.17, $-\frac{m}{2}\log 2\pi\sigma$, is independent of $\mathbf{w}$, maximizing the log-likelihood function is equivalent to minimizing a quadratic error term

$$E = \frac{1}{2}(y - h(\mathbf{x}; \mathbf{w}))^2 \Longleftrightarrow p(y|\mathbf{x}; \mathbf{w}) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{(y - h(\mathbf{x}; \mathbf{w}))^2}{2\sigma}) \tag{5.18}$$

This is the square-error loss function, or least mean square (LMS) error if we consider a batch algorithm, that has long been the choice for linear regression. It has also been

dominant in some other machine learning approaches such as multilayer perceptrons for a long time. In terms of our probabilistic view, the LMS regression is equivalent to MLE for Gaussian data with linear dependence of the mean and constant variance. However, this is a strong assumption and is not always the case. For example, when the variance is a free parameter, then we need to minimize equation 5.17 instead, and of course, we are also mostly interested in nonlinear extensions later.

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

$$E = \frac{1}{p}||y - h(\mathbf{x}; \mathbf{w})||^p \Longleftrightarrow p(y|\mathbf{x}; \mathbf{w}) = \frac{1}{2\Gamma(1/p)} \exp(-||y - h(\mathbf{x}; \mathbf{w})||^p). \quad (5.19)$$

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

$$E = ||y - h(\mathbf{x}; \mathbf{w}||_\epsilon \Longleftrightarrow p(y|\mathbf{x}; \mathbf{w}) = \frac{p}{2(1-\epsilon)} \exp(-||y - h(\mathbf{x}; \mathbf{w})||_\epsilon). \quad (5.20)$$

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.

5.3 Probabilistic classification

An important special case of learning problems is **classification** in which features are mapped to a finite number of possible categories. We are now going back to **binary classification**, which is the case of two target classes where the target function (y-values) has only two possible values such as 0 and 1. And we will treat this binary classification as a specific probabilistic model.

5.3.1 MLE of Bernoulli model

Let us consider a random number which takes the value of 1 with probability ϕ and the value 0 with probability $1 - \phi$ (the probability of being either of the two choices has to be 1.). That is,

$$p(y) = \phi^y (1 - \phi)^{1-y} \quad (5.21)$$

Such a random variable is called Bernoulli distributed, and all Bernoulli distributions are characterized by one parameter ϕ. Tossing a coin is a good example of a Bernoulli process (a process of generating such random numbers). We can use maximum likelihood estimation (MAP with uniform prior) to estimate the parameter ϕ from such trials. That is, let us consider m tosses in which h heads have been found. The log-likelihood of such m trials is

$$l(\phi) = log \prod_i \phi^{y^{(i)}} (1 - \phi)^{1-y^{(i)}} \quad (5.22)$$

$$= \log(\phi^h(1-\phi)^{m-h}) \tag{5.23}$$
$$= h\log(\phi) + (m-h)\log(1-\phi). \tag{5.24}$$

To find the maximum with respect to ϕ we set the derivative of l to zero,

$$\frac{\mathrm{d}l}{\mathrm{d}\phi} = \frac{h}{\phi} - \frac{m-h}{1-\phi} \tag{5.25}$$
$$= \frac{h}{\phi} - \frac{m-h}{1-\phi} \tag{5.26}$$
$$= 0 \tag{5.27}$$
$$\rightarrow \quad \phi = \frac{h}{m} \tag{5.28}$$

As you might have expected, the maximum likelihood estimate of the parameter ϕ is the fraction of heads in m trials. This also explains why taking this ratio for the parameter estimate of the rain prediction example of Chapter **??** was appropriate. We used the MLE for Bernoulli random variables.

5.3.2 Logistic regression

Of course, we usually consider the case of classification when the parameter ϕ, depends on an attribute x, as usual in a stochastic (noisy) way. More specifically,

$$p(y=1|\mathbf{x};\mathbf{w}) = h(\mathbf{x};\mathbf{w}) \tag{5.29}$$
$$p(y=0|\mathbf{x};\mathbf{w}) = 1 - h(\mathbf{x};\mathbf{w}), \tag{5.30}$$

where $h(\mathbf{x};\mathbf{w}$ is a hypothesis function (not the number of heads as before). We can combine the probabilities into one expression,

$$p(y|\mathbf{x};\mathbf{w}) = (h(\mathbf{x};\mathbf{w}))^y(1-h(\mathbf{x};\mathbf{w}))^{1-y} \tag{5.31}$$

The corresponding log-likelihood function is

$$l(\mathbf{w}) = \sum_{i=1}^{m} y^{(i)}\ log(h(\mathbf{x};\mathbf{w})) + (1-y^{(i)})\ log(1-h(\mathbf{x};\mathbf{w})). \tag{5.32}$$

To find the corresponding maximum we can use the gradient ascent algorithm, which is like the gradient descent algorithm with a changed sign,

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\nabla_{\mathbf{w}} l(\mathbf{w}). \tag{5.33}$$

To calculate the gradient we can calculate the partial derivative of the log-likelihood function with respect to each parameters,

$$\frac{\partial l(\mathbf{w})}{\partial w_j} = \left(y\frac{1}{h} - (1-y)\frac{1}{1-h}\right)\frac{\partial h(\mathbf{w})}{\partial w_j} \tag{5.34}$$

where we dropped indices for better readability.

Let us apply this to logistic regression. An example of 100 sample points of two classes (crosses and stars) are shown in Fig.5.3. The data suggest that it is far more likely that the class is $y = 0$ for small values of x and that the class is $y = 1$ for large values of x, and the probabilities are more similar in between. Thus, we put forward the hypothesis that the transition between the low and high probability region is smooth and qualify this hypothesis as parameterized density function known as a **logistic** or **sigmoid** function

$$h = g(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x})}. \tag{5.35}$$

As before, we can then treat this density function as function of the parameters $\mathbf{w}$ for the given data values (likelihood function), and use maximum likelihood estimation to estimate values for the parameters so that the data are most likely. The density function with sigmoidal offset $w_0 = 2$ and slope $w_1 = 4$ is plotted as solid line in Fig.5.3.

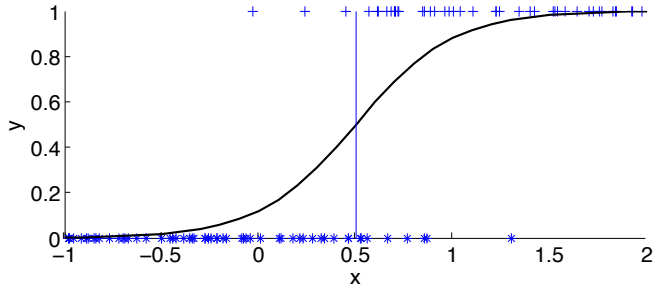

Fig. 5.3 Binary random numbers (stars) drawn from the density $p(y=1) = \frac{1}{1+\exp(-w_1 x - w_0)}$ (solid line).

We can now calculate the derivative of the hypothesis h with respect to the parameters for the specific choice of the logistic functions. This is given by

$$\frac{\partial h}{\partial w} = \frac{\partial}{\partial w}\frac{1}{1+e^{-wx}} \tag{5.36}$$

$$= \frac{1}{(1+e^{-wx})^2}e^{-wx}(-x) \tag{5.37}$$

$$= \frac{1}{(1+e^{-wx})}(1-\frac{1}{(1+e^{-wx})})(-x) \tag{5.38}$$

$$= -h(1-h)x \tag{5.39}$$

Using this in equation 5.34 and inserting it into equation 5.33 with the identity

$$\left(y\frac{1}{h} - (1-y)\frac{1}{1-h}\right)h(1-h) = y(1-h) - (1-y)h \tag{5.40}$$

$$= y - yh - h + yh \tag{5.41}$$

$$= y - h \tag{5.42}$$

gives the learning rule

$$w_j \leftarrow w_j + \alpha(y^{(i)} - h(\mathbf{x}^{(i)}, \mathbf{w})x_j^{(i)} \tag{5.43}$$

Hence, the learning rule for logistic regression is a similar to the learning rule for linear regression on Gaussian data. We will later see that logistic regression is equivalent to a simple neural network called a perceptron for which this learning rule has been proposed on heuristic grounds and is usually called the perceptron learning rule. Our derivation shows how such a learning rule relates to assumptions of a probabilistic model.

How can we use the knowledge (estimate) of the density function to do classification? The obvious choice is to predict the class with the higher probability, given the input attribute. This **bayesian decision point**, x^t, or **dividing hyperplane** in higher dimensions, is give by

$$p(y = 1|x^t) = p(y = 0|x^t) = 0.5 \rightarrow x^t \mathbf{w}^T \mathbf{x^t} = 0. \tag{5.44}$$

We have here considered binary classification with linear decision boundaries as logistic regression, and we can also generalize this method to problems with non-linear decision boundaries by considering hypothesis with different functional forms. This will be the subject when discussing neural networks later in this course.

5.4 MAP and Regularization with priors

Maximum likelihood estimation is the workhorse of probabilistic supervised learning, but it is worth while to put this into the context of the wider probabilistic framework. In the probabilistic sense, choosing parameters values given data should be based on the posterior

$$p(\mathbf{w}|\mathbf{x}, y). \tag{5.45}$$

Let us assume we would know this conditional distribution like the 1-dimensional example shown in Fig. 5.4. If we know this distribution we can pick a parameter value that we like. For example, we could pick the value w_1 which is the most probable value given the specific data. This is called the **maximum a posteriori (MAP)**

$$\mathbf{w}^{\mathrm{MAP}} = \mathrm{argmax}_{\mathbf{w}} p(\mathbf{w}|\mathbf{x}, y). \tag{5.46}$$

Before we go on to discuss how to calculate this, let is point out that both learning methods, MLE and MAP, 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. For example, looking again at the example in Fig. 5.4, another strategy might be to pick a weight value in a range where variations in this value would not change the probability considerably in some range, such as values around w_2.

However, the main difficulty with MAP or using another procedure based on the distribution $p(\boldsymbol{w}|\mathbf{x}, y)$ is that we usually do not know this distribution a priori. Instead,

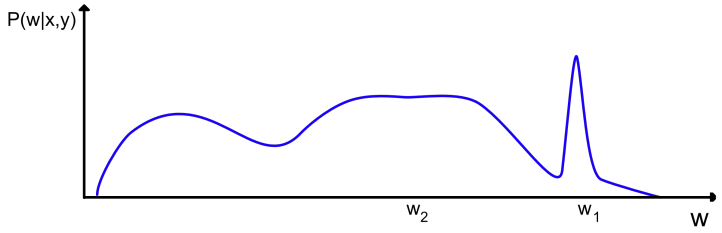

Fig. 5.4 Example of a possible probability distribution of a parameter w given some data.

our approach has been to formulate a probabilistic model in form of a parameterized density function like

$$p(y|\mathbf{x};\mathbf{w}). \tag{5.47}$$

We will no discuss the relation of these density functions. To start with, in the probabilistic model of eq.5.47 we assumed that the feature data are magically given, but we could also consider how we select data values in a randomized fashion so that we could also consider as model

$$p(y,\mathbf{x}|\mathbf{w}) = p(y|\mathbf{x};\mathbf{w})p(\mathbf{x}|\mathbf{w}) \tag{5.48}$$

We factorized thereby the joined density with the conditional density $p(y|\mathbf{x};\mathbf{w})$ and the marginal density $p(\mathbf{x}|\mathbf{w})$. Now we will choose to pick the training data uniformly, so that the marginal distribution over the features in just a constant, $p(\mathbf{x}|\mathbf{w}) = \text{const}$, and hence

$$p(y,\mathbf{x}|\mathbf{w}) \propto p(y|\mathbf{x};\mathbf{w}). \tag{5.49}$$

Next, we can use Bayes theorem to posterior $p(\mathbf{w}|\mathbf{x},y)$ to the hypothesis model, namely

$$p(\mathbf{w}|\mathbf{x},y) = \frac{p(y,\mathbf{x}|\mathbf{w})p(\mathbf{w})}{\int_{\mathbf{w}'\in W} p(\mathbf{x},y|\mathbf{w}')p(\mathbf{w}')\mathrm{d}\mathbf{w}'}, \tag{5.50}$$

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 $\mathbf{w}$ as we are integrating (summing) over all possible values. The most likely values for the parameters can thus be calculated without this term and is given by the maximum a posteriori (MAP) estimate,

$$\boldsymbol{w}^{\mathrm{MAP}} = \mathrm{argmax}_{\mathbf{w}} p(\mathbf{x},y|\boldsymbol{w})p(\boldsymbol{w}). \tag{5.51}$$

The name of the method comes from the fact that we modify our prior knowledge of parameters, which is summarized as prior distribution $p(\boldsymbol{w})$ by combining this to measurements $(\mathbf{x},y)$ from specific realizations of the parameters, which is given by the likelihood function $p(\mathbf{x},y|\boldsymbol{w})$. The resulting posterior distribution should then be a better estimate of the probability of values for the parameters. The function $\mathrm{argmax}_x(f(x))$ pics the argument x for which the function $f(x)$ is maximal. The argument of the function is the set of parameters that 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).

We are now ready to discuss again regularization in the probabilistic framework. In section **??** we introduced L^p regularization by introducing a bias in the parameters. In the probabilistic models we can do this very elegantly by providing a prior for the parameters that encapsulate the bias in the choice of parameter as argued previously. For example, let us assume that the values of the parameters should be more likely be small than large. More specifically, lets assume that we think they should be Normal distributed around zero. Then we can write the MAP estimate

$$\boldsymbol{w}^{\mathrm{MAP}} = \mathrm{argmax}_{\mathbf{w}} p(\mathbf{x}, y|\boldsymbol{w}) p(\boldsymbol{w}) \tag{5.52}$$

as

$$\boldsymbol{w}^{\mathrm{MAP}} = \mathrm{argmax}_{\mathbf{w}} p(\mathbf{x}, y|\boldsymbol{w}) \mathcal{N}(0,\, \sigma^2)\,. \tag{5.53}$$

As usual we should maximize the logarithm of the corrsponding likelihood functions,

$$\boldsymbol{w}^{\mathrm{MAP}} = \mathrm{argmax}_{\mathbf{w}} \log(p(\mathbf{x}, y|\boldsymbol{w})) + \alpha||w||^2, \tag{5.54}$$

with

$$\alpha = \log \frac{1}{2\sigma^2}. \tag{5.55}$$

Thus, our previously discussed L^2 regularization corresponds to a Gaussian prior on the weights. Similarly, L^1 regularization correspond to a prior of a symmetric Laplace distribution

$$p(\mathbf{w}) = \frac{1}{2\,b} \exp\left(-\frac{|\mathbf{w}|}{b}\right), \tag{5.56}$$

which is more peaked towards zero. Hence this norm forces weights more towards the zero compared to the L^2 regularization.