MML book Note part II

October 1, 2025

Contents

8 When Models Meet Data **5**
8.1 The three major components of a ML system: Data, Models, and Learning 5
8.1.1 Data as Vectors 5
8.1.2 Models as Functions 6
8.1.3 Models as Probability Distributions 6
8.1.4 Learning is Finding Parameters 6
8.2 Empirical Risk Minimization 7
8.2.1 Hypothesis Class of Functions 7
8.2.2 Loss Function for Training 7
8.2.3 Regularization to Reduce Overfitting 8
8.2.4 Cross-Validation to Assess the Generalization Performance 8
8.2.5 Further Reading 9
8.3 Parameter Estimation 9
8.3.1 Maximum Likelihood Estimation (MLE) 9
8.3.2 Maximum A Posteriori Estimation 9
8.3.3 Model Fitting 10
8.3.4 Further reading 10
8.4 Probabilistic Modeling and Inference 11
8.4.1 Probabilistic Models 11
8.4.2 Bayesian Inference 11
8.4.3 Latent-Variable Models 11
8.4.4 Further Reading 12
8.5 Directed Graphical Models 12
8.5.1 Graph Semantics 13
8.5.2 Conditional Independence and d-Separation 14
8.5.3 Furhter Reading 14
8.6 Model Selection 14
8.6.1 Nested Cross-Validation 14
8.6.2 Bayesian Model Selection 15
8.6.3 Bayes Factors for Model Comparison 16
8.6.4 Further Reading 16
8.7 Problem Formulation 17
8.8 Parameter Estimation 18
8.8.1 Maximum Likelihood Estimation 18

Chapter 8

When Models Meet Data

8.1 The three major components of a ML system: Data, Models, and Learning

Question: What do we mean by good models?

Good Model : should perform well on unseen data. $\Rightarrow$ performance metrics (accuracy or distance frmo ground truth)

Machine learning algorithm : training and prediction

8.1.1 Data as Vectors

Data is assumed to be tabular. Rows are intances from population, and columns are features. If we have some categorical value, we can convert them into numerical by assining a number to each distinct value. But even for this kind of data, we should consider carefully their units, scaling and constraints. We may shit and scale all columns of dataset such that they have an empirical mean of and an empirical variance of 1.

Notation in this course :

$\boldsymbol{N}$: denote the number of examples in a dataset

$n = \mathbf{1,\ldots,}\boldsymbol{N}$: index of each example

x_n : row in a dataset as numerical table. nth example out of a total of N examples in the dataset.

$d = 1, \ldots, \boldsymbol{D}$: Index of each column (feature)

Supervised learning : we have a label or a target or a response variable or an annotation y_n (the salary) associated with each example x_n (the age).

In this book, it is assumed that : each input x_n is a D-dimensional vector of real numbers, which are called **features, attributes, or covariates.**. We asume that the vector is a column vector in this case.

A dataset is written as a set of example–label pairs

$$\{(x_1, y_1), \ldots, (x_n, y_n), \ldots, (x_N, y_N)\}.$$

The table of examples

$$\{x_1, \ldots, x_N\}$$

is often concatenated, and written as

$$X \in \mathbb{R}^{N \times D}.$$

Feature map $\Phi()$: allows us to represent inputs x_n using a higher-dimensional representation $\Phi(x_n)$. This leads to a *kernel*

8.1.2 Models as Functions

Predicator : a predictive function Two approches in the book: a predictor as a function, and a predictor as a probabilistic model

Predictor is a function that, when given a particular input example, produces an output.

$$f : \mathbb{R}^D \to \mathbb{R} \tag{8.1}$$

Where the input vector x is D-dimensional (has D features), and the function f return a real number.

$$f(x) = \theta^T x + \theta_0 \tag{8.2}$$

for unkown θ and θ_0.

8.1.3 Models as Probability Distributions

Data as a noisy observations where some truth is hidden. We need then to expect that our predictor express some sort of uncertainty. $\Rightarrow$ predictor as probabilistic models,i.e, models describing the distribution of possible functions. $\Rightarrow$ multivariate probability distributions

8.1.4 Learning is Finding Parameters

Goal: find a model and its corresponding parameters to perform well on unseen data. Phases:

1. Prediction or inference (for probabilistic model)

2. Training or parameter estimation

3. Hyperparameter tuning or model selection

Empirical risk minimization: For the non-probabilistic model, which provides an optimization problem for finding good parameters.

Maximum likelihood: For a statistical model, which is used to find a good set of parameters

We use numerical methods to find good parameters that "fit" the data.

Cross-validation : Simulate the behavior of our predictor on future unseen data

Abduction: balance between fitting well on training data and finding "simple" explanations of the phenomenon. abduction is the process of inference to the best explanation

Hyperparameter : The choice of the number of components **Model selection** : The problem of choosing different models. **Nested cross-validation** for non-probabilistic models.

Distinction between parameters and hyperparameters: consider parameters as the explicit parameters of a probabilistic model, and to consider hyperparameters (higher-level parameters) as parameters that control the distribution of these explicit parameters.

8.2 Empirical Risk Minimization

(ERM $\Rightarrow$ for the case of predictor as a function).
Learning means estimating parameters based on training data.

ERM was originally popularize by the proposal of the support vector machine.
Design choices we will cover in subsections.

8.2.1 Hypothesis Class of Functions

What is the set of functions we allow the predictor to take? Consider N examples $x_n \in \mathbb{R}^R$, $y_n \in \mathbb{R}$. The dataset : $(x_1, y_1), \dots, (x_N, y_N)$.
Predictor to be estimated: $f(\cdot, \theta) : \mathbb{R}^R \to \mathbb{R}$ parametrized by θ.
Find a good parameter θ^* such that :

$$f(x_n, \theta^*) \approx y_n \text{ for all } n = 1, \dots, N \tag{8.3}$$

The ouput of predictor is anoted as $\hat{y}_n = f(x_n, \theta^*)$
Exemple 8.1 (see p.259 of the book)

8.2.2 Loss Function for Training

How do we measure how well the predictor performs on the training data?
y_n: current label and $\hat{y}_n$ the prediction based on x_n.
loss function $\ell(y_n, \hat{y}_n)$: define what it means to fit the data sell. Produces a non-negative number (the loss) representing the error that was made.

Finding a good parameter vector $\theta^* \leftrightarrow$ minimize the average loss on the set of N training examples. *Assumption in ML:* dataset is *independent (two data points are not statistically dependent on each other) and indentically distributed.* This leads us the following formula. Matrix of training data $X := [x_1, \dots, x_N]^T \in \mathbb{R}^{N \times D}$. A label vector $y := [y_1, \dots, x_N]^T \in \mathbb{R}^N$.
Empirical risk :

$$R_{\text{emp}}(f, X, y) = \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, \hat{y}_n), \tag{8.6}$$

where $\hat{y}_n = f(x_n, \theta)$. f : predictor X : data y: label For unseen test data $\to$ minimizes the *expected risk*

$$R_{\text{true}}(f) = \mathbb{E}_{x,y}\big[\ell(y, f(x))\big], \tag{8.10}$$

Where y is the label and $f(x)$ is the prediction based on the example x. The notation $R_{\text{true}}(f)$ means that this is the true risk if we had access to an infinite amount of data.

Two pratical questions that arises when minimizing expected risk :

- How should we change our training procedure to generalize well?
- How do we estimate expected risk from (finite) data?

8.2.3 Regularization to Reduce Overfitting

How do we construct predictors from only training data that performs well on unseen test data?
Test set: a holded proportion of the whole dataset. The performance on test set is unknown because Even knowing only the performance of the predictor on the test set leaks information (Blumand Hardt, 2015).
The subscripts $_{train}$ and $_{test}$ to denote the training and test sets.
Overfitting : the predictor fits too closely to the training data and does not generalize well to new data. If the test risk is much larger than training risk $R_{emp}(f, X_{train}, y_{train}) << R_{true}(f)$
$\Rightarrow$ penalty term or *regularization*: a way to compromise between accurate solution of empirical risk minimization and the size or complexity of the solution. biases the vector θ to be closer to the origin. **Example 8.3 (Regularized Least Squares)** see p.262

8.2.4 Cross-Validation to Assess the Generalization Performance

What is the procedure for searching over the space of models?
Validation set $\mathcal{V}$: test data. a subset of the available training data that we keep aside.
Cross-validation

- K-fold $\rightarrow$ partition the data into K chunks
- $K-1$ for training set $\mathcal{R}$ and the last chunk as the *validation set* $\mathcal{V} \rightarrow \mathcal{D} = \mathcal{R} \cup \mathcal{V}$ such that $\mathcal{R} \cap \mathcal{V} = \emptyset$
- train through $\mathcal{R}$ and test through $\mathcal{V}$
- repeat for K different choices of $\mathcal{V}$

$\Rightarrow$ for each partition k the training data $\mathcal{R}^{(k)}$ produces a predictor $f^{(k)}$, which is then applied to validation set $\mathcal{V}^{(k)}$ to compute the empirical risk $\mathcal{R}(f^{(k)}, V^{(k)})$. We cycle through all possible partitionings of validation and training sets and compute the average generalization error of the predictor.
Cross-validation approximates the expected generalization error

$$\mathbb{E}[R(f, \mathcal{V})] \approx \frac{1}{K} \sum_{k=1}^{K} \mathcal{R}(f^{(k)}, V^{(k)})$$

Where $\mathcal{R}(f^{(k)}, V^{(k)})$. is the risk on the validation set $\mathcal{V}^{(k)}$ for predictor $f^{(k)}$.
Embarrassingly parallel : little effort is needed to separate the problem into a number of parallel tasks.

8.2.5 Further Reading

see p.265

8.3 Parameter Estimation

how to use probability distributions to model our uncertainty due to the observation process and our uncertainty in the parameters of our predictors.

8.3.1 Maximum Likelihood Estimation (MLE)

To define a function of the parameters that enables us to find a model that fits the data well. $\Rightarrow$ *Likelihood function* or its negative logarithm.
Negative log-likelihood: for data represented by a random variable x and for a family of probability densities $p(x|`\theta$ parametrized by θ

$$\mathcal{L}_x(\theta) = -\log p(x|\theta) \tag{8.14}$$

The parameter θ is varying while the data x is fixed. $\Rightarrow \mathcal{L}(\theta)$ when we consider it as a function of θ.
We can interpret the probability the predictor constructed here as : given a vector x_n we want the probability distribution of the label y_n.

$(x_1, y_1), \dots, (x_N, y_N)$ as *independent and identically distributed* $\Rightarrow p(\mathcal{Y}|\mathcal{X}, \theta) = \prod_{n=1}^{N} p(y_n|x_n, \theta)$(8.16)
where $p(y_n|x_n, \theta)$ is a particular distribution, $\mathcal{Y} = \{y_1, \dots, y_N\}$ and $\mathcal{X} = \{x_1, \dots, x_N\}$.
The expression "identically distributed" means that each term in the product (8.16) is of the same distribution, and all of them share the same parameters.
In ML, we often consider the negative log-likelihood:

$$\mathcal{L}(\theta) = -\log p(\mathcal{Y}|\mathcal{X}, \theta) = -\sum_{n=1}^{N} \log p(y_x|x_n, \theta) \tag{8.17}$$

We ought to minimize $\mathcal{L}(\theta)$ with respect to θ.
Remark. The negative sign in (8.17) is a historical artifact that is due to the convention that we want to maximize likelihood, but numerical optimization literature tends to study minimization of functions.

Example 8.5 see p.267

8.3.2 Maximum A Posteriori Estimation

x as data and θ as the parameter, we got the *posterior* distribution $p(x|\theta)$ from Bayes' theorem:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} \tag{8.19}$$

We are interested in finding the parameter θ that maximizes the posterior. Since $p(x)$ does not depend on θ, we ought to optimize this instead:

$$p(\theta|x) \alpha p(x|\theta)p(\theta) \tag{8.20}$$

Instead of estimating the minimum of the negative log-likelihood, we now estimate the minimum of the negative log-posterior, which is referred to as *maximum a posteriori estimation (MAP estimation).*
Example 8.6 (see p.269)
Remark

- In ML, the idea of including prior knowledge about where good parameters lie is widespread.

- The maximum likelihood estimate θ_{ML} possesses the following properties

  Asymptotic consistency The MLE converges to the true value in the limit of infinitely many observations, plus a random error that is approximately normal

  The size of the samples necessary to achieve these properties can be quite large.

  The error's variance decays in $1/N$, where N is the number of data points. Especially, in the "small" data regime, maximum likelihood estimation can lead to overfitting.

MLE and MPE uses probabilistic modeling to reason about the uncertainty in the data and model parameters.

8.3.3 Model Fitting

Fitting : mean optimizing/learning model parameters so that they minimize some loss function (e.g, the negative log-likelihood).
M_θ : Model class
M^* : unknown model
Training: For a given training dataset, we optimize θ so that M_θ is as close as possible to M^* , where the "closeness" is defined by the objective function we optimize.
After we obtain the best possible parameters θ^* (optimizations), we distinguish three different cases:

Overfitting : refers to the situation where the parametrized model class is too rich to model the dataset generated by M^* , i.e., M_θ could model much more complicated datasets. One way to detect overfitting in practice is to observe that the model has low training risk but high test risk during cross validation.

Underfitting : we encounter the opposite problem where the model class M_θ is not rich enough. Models that underfit typically have few parameters.

Fitting well : when the parametrized model class is about right, i.e., it neither overfits nor underfits.

8.3.4 Further reading

see p.272

8.4 Probabilistic Modeling and Inference

ML: interpretation and analysis of data $\Rightarrow$ models that describe the *generative process* that generates the observed data.

8.4.1 Probabilistic Models

From tje observed variables x and the hidden parameters θ, a probabilistic model is specified by the joint distribution $p(x, \theta)$ of all random variables. It represent the uncertain aspects of an experiment as probability distributions.
$\Rightarrow$ set of tools from probability theory (Chapter 6) for modeling, inference, prediction, and model selection.
Encapsulate information from :

- The prior and the likelihood
- The marginal likelihood $p(x)$
- The posterior

8.4.2 Bayesian Inference

The predictive distribution will be $p(x|\theta^*)$ where we use θ_* in the likelihood function.
Bayesian inference is about learning the distribution of random variables, thus it is about finding the posterior distribution.
For a dataset $\mathcal{X}$, a parameter prior $p(\theta)$, and a likelihood function, the posterior

$$p(\theta|\mathcal{X}) = \frac{p(\mathcal{X}|\theta)p(\theta)}{p(\mathcal{X})}, p(\mathcal{X}) = \int p(\mathcal{X}|\theta)p(\theta)d\theta, \tag{8.22}$$

is obtained by applying Bayes' theorem. Bayesian inference inverts the relationship between parameters θ and the data $\mathcal{X}$ to obtain the posterior distribution $p(\theta|\mathcal{X})$.
With a distribution $p(\theta)$ on the parameters, our predictions will be

$$p(x) = \int p(x|\theta)p(\theta)d\theta = \mathbb{E}[p(x|\theta)], \tag{8.23}$$

$\Rightarrow$ The prediction is an average over all plausible parameter value θ, where the plausibility is encapsulated by the parameter distribution $p(\theta)$
See p.274 for comparison between parameter estimation(optimization problem) and Bayesian inference(integral problem).
Remark: In the machine learning literature, there can be a somewhat arbitrary separation between (random) "variables" and "parameters". While parameters are estimated (e.g., via maximum likelihood), variables are usually marginalized out.

8.4.3 Latent-Variable Models

z: *Latent variables* (besides the model parameters θ). It may describe the data-generating process, thereby contributing to the interpretability of the model. They also often simplify the structure of the model and allow us to define simpler and richer model structure.
$\Rightarrow$ We often use Expectation maximization algorithm.

x as data, θ as the model parameters, z as the latent variables, we obtain the conditional distribution

$$p(x|z,\theta) \tag{8.24}$$

that allows us to generate data for any model parameters and latent variables.
Two-step procedure to facilitate learning:

1. Compute the likelihood $p(x|\theta)$ of the model
2. Use this likelihood for parameter estimation or Bayesian inference

To marginalize out the latent variables:

$$p(x|\theta) = \int p(x|z,\theta)p(z)dz, \tag{8.25}$$

where $p(x|z,\theta)$ is given in (8.24) and $p(z)$ is the prior on the latent variables.
The posterior distribution

$$p(\theta|\mathcal{X}) = \frac{p(\mathcal{X}|\theta)p(\theta)}{p(\mathcal{X})}) \tag{8.26}$$

over the model parameters given a dataset $\mathcal{X}$.
Posterior on the latent variables according to

$$p(z|\mathcal{X}) = \frac{p(\mathcal{X}|z)p(z)}{p(\mathcal{X})}, \quad p(\mathcal{X}|z) = \int p(\mathcal{X}|z,\theta)p(\theta)d\theta \tag{8.27}$$

where $p(z)$ is the prior on the latent variables and $p(\mathcal{X}|z)$ requires us to integrate out the model parameters θ .
The posterior distribution on the latent variables, but conditioned on the model parameters, i.e.,

$$p(z|\mathcal{X},\theta) = \frac{p(\mathcal{X}|z,\theta)p(z)}{p(\mathcal{X}|\theta)}, \tag{8.28}$$

where $p(z)$ is the prior on the latent variables and $p(\mathcal{X}|z,\theta)$ is given in (8.24).
Remark (see p.277)

8.4.4 Further Reading

see p.277
read about *probabilistic programming*

8.5 Directed Graphical Models

A graphical language for specifying a probabilistic model, in order to visually parse dependencies between random variables

Directed graphical models are also known as Bayesian networks.
In *graphical model*, nodes are random variables.
Remark Not every distribution can be represented in a particular choice of graphical model.
Properties

- They are a simple way to visualize the structure of a probabilistic model.
- They can be used to design or motivate new kinds of statistical models.
- Inspection of the graph alone gives us insight into properties, e.g., conditional independence.
- Complex computations for inference and learning in statistical models can be expressed in terms of graphical manipulations.

8.5.1 Graph Semantics

Directed graphical models/Bayesian networks: are a method for representing conditional dependencies in a probabilistic model.
Directed **links** (arrows) between two **nodes** (random variables) indicate conditional probabilities.
Directed graphical models can be derived from joint distributions if we know something about their factorization.
Example 8.7 (see p279)
Step for creating a directed graphical model from a factorized joint distribution

1. Create a node for all random variables.
2. For each conditional distribution, we add a directed link (arrow) to the graph from the nodes corresponding to the variables on which the distribution is conditioned.

The graph layout depends on the choice of factorization of the joint distribution. **Example 8.8** (see p 280)

In , the joint distribution $p(x) = p(x_1, \dots, x_K)$ is given as

$$p(x) = \prod_{k=1}^{K} p(x_k|\mathrm{Pa}_k), \tag{8.31}$$

Where Pa_k means "the parent nodes of x_k"

Application with a Bernouilli experiment
probability that the outcome x pf this experiment is "head" is

$$p(x|\mu) = \mathrm{Ber}(\mu) \tag{8.32}$$

After N repetition, we get $x_1, \dots, x_N$ outcomes $\Rightarrow$

$$p(x_1, \dots, x_N|\mu) = \prod_{n=1}^{N} p(x_n|\mu) \tag{8.33}$$

To write the graphical model down for this setting, we make the distinction between unobserved/latent variables and observed variables.
See p. 281 For the graphical model with *Plate* notation.
Hyperprior: a second layer of prior distributions on the parameters of the first layer of priors.

8.5.2 Conditional Independence and d-Separation

Consider a general directed graph in which $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ are arbitrary nonintersecting sets of nodes.
We want to verify that the following statement is implied by a given directed acyclic graph:

$$\text{" } \mathcal{A} \text{ is conditionally independent of } \mathcal{B} \text{ given } \mathcal{C} \text{ "} \Leftrightarrow \mathcal{A} \perp\!\!\!\perp \mathcal{B} | \mathcal{C} \tag{8.34}$$

To do so, we consider all possible trails (paths that ignore the direction of the arrows) from any node in $\mathcal{A}$ to any nodes in $\mathcal{B}$. Any such path is said to be blocked if it includes any node such that either of the following are true:

- The arrows on the path meet either head to tail or tail to tail at the node, and the node is in the set C .
- The arrows meet head to head at the node, and neither the node nor any of its descendants is in the set C .

If all paths are blocked, then $\mathcal{A}$ is said to be *d-separated* from $\mathcal{B}$ by $\mathcal{C}$,and the joint distribution over all of the variables in the graph will satisfy $\mathcal{A} \perp\!\!\!\perp \mathcal{B} | \mathcal{C}$.
Example 8.9 (Conditional Independence) (see p.282)

8.5.3 Furhter Reading

see p.283

- Bishop, Christopher M. 2006. Pattern Recognition and Machine Learning. Springer.
- Koller, Daphne, and Friedman, Nir. 2009. Probabilistic Graphical Models. MIT Press.
- Herbrich, Ralf, Minka, Tom, and Graepel, Thore. 2007. TrueSkill(TM): A Bayesian Skill Rating System. In: Advances in Neural Information Processing Systems.

8.6 Model Selection

Model performance is very important from the point of view of testing set. We should consider if we need a complex model to describe the relationship in data or something else, many things influence the decision when we are choosing the right approach of training and when we are choosing the model to train.

8.6.1 Nested Cross-Validation

For each split, we can perform another round of cross-validation.
The are two level : the inner level and outer level.
Test set : the set used to estimate the generalization performance.
Validation set : the set used for choosing the best model.

The inner loop estimates the expected value of the generalization error for a given model (8.39), by approximating it using the empirical error on the validation set, i.e.,

$$\mathbb{E}_{\mathcal{V}}[R(\mathcal{V}|M)] \approx \frac{1}{K}\sum_{k=1}^{K} R(\mathcal{V}^{(k)}|M), \tag{8.36}$$

where $R(\mathcal{V}|M)$ is the empirical risk (e.g., root mean square error) on the validation set $\mathcal{V}$ for model M. We repeat this procedure for all models and choose the model that performs best. Once the model is chosen, we can evaluate the final performance on the test set.
The standard error is defined as $\frac{\sigma}{\sqrt{K}}$ where K is the number of experiments and σ is the standard deviation of the risk of each experiment.

8.6.2 Bayesian Model Selection

To trade off model complexity and data fit.
Occam's razor: find the simplest model that explains the data reasonably well.
An "automatic Occam's Razor" is quantitatively embodied in the application of Bayesian probability, thus, we can use this for model selection instead of doing the process manually.
Evidence for M_i: $p(\mathcal{D}|M_i)$
$\Rightarrow$ Hierarchical inference problem.
Consideration :

- A finite number of models $M = \{M_1, \ldots, M_K\}$
- M_k : model that possesses parameters θ_k
- In Bayesian model selection, we place a prior $p(M)$ on the set of models.

The **corresponding generative** process that allows us to generate data from this model is

$$M_k \sim p(M) \tag{8.40}$$
$$\theta_k \sim p(\theta|M_k) \tag{8.41}$$
$$\mathcal{D} \sim p(\mathcal{D}|\theta_k) \tag{8.42}$$

The posterior distribution over models

$$p(M_k|\mathcal{D}) \; \alpha \; p(M_k)p(\mathcal{D}|M_k) \tag{8.43}$$

Model evidence or marginal likelihood:

$$p(\mathcal{D}) = \int p(\mathcal{D}|\theta_k)p(\theta_k|M_k)d\theta_k \tag{8.44}$$

where $p(\theta_k|M_k)$ is the prior distribution of the model parameters θ_k of the model M_k.
The MAP estimation:

$$M^* = arg\ \max_{M_k} p(M_k|\mathcal{D}) \tag{8.45}$$

Remark (Difference between Likelihood and Marginal Likelihood): While the likelihood is prone to overfitting, the marginal likelihood is typically not as the model parameters have been marginalized out (i.e., we no longer have to fit the parameters). Furthermore, the marginal likelihood automatically embodies a trade-off between model complexity and data fit (Occam's razor).

8.6.3 Bayes Factors for Model Comparison

Comparing two probabilistic models M_1 , M_2 ,given a dataset $\mathcal{D}$. **Posterior odds**: see (8.46) on page 287. You can see there the definition to **Prior odds** and **Bayes factor**. **Remark**: *Jeffreys-Lindley paradox*. (see P287 for detail about choosing a model according to Bayes factors)

8.6.4 Further Reading

see p288

chapterLinear Regression Find a function that maps inputs $x \in \mathbb{R}^D$ to corresopnding function values $f(x) \in \mathbb{R}$.
We assume zero-mean Gaussian noise.
Requiered problem to be solved when finding a regression function:

- Choice of the model (type) and the parametrization of the regression function
- Finding good parameters
- Overfitting and model selection
- Relationship between loss functions and parameter priors
- Uncertainty modeling

8.7 Problem Formulation

Model the noise using a likelihood function:

$$p(y|x) = \mathcal{N}(y|f(x), \sigma^2) \tag{9.1}$$

Here, $x \in \mathbb{R}^D$ are inputs and $y \in \mathbb{R}$ are noisy function values targets).
The relationship between x and y :

$$y = f(x) + \upsilon \tag{9.2}$$

Where $\upsilon \sim \mathcal{N}(0, \sigma^2)$ is independent, identically distributed (i.i.d) Gaussian measurement noise with mean 0 and variance σ^2.
Objective : to find a function that is close (similar) to the unknown function f that generated the data and that generalizes well.
Parametric models (parameters θ): σ^2 as the noise variance, we will focus on learning the model parameters θ.
θ appear linearly in our model like:

$$p(y|x, \theta) = \mathcal{N}(y|x^T\theta, \sigma^2) \tag{9.3}$$

$$\Leftrightarrow y = x^T\theta + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2) \tag{9.4}$$

Where $\theta \in \mathbb{R}^D$ the parameters we seek.
The likelihood in (9.3) is the probability density function of y evaluated at $x^T\theta$.

A Dirac delta (delta function) is zero everywhere except at a single point, and its integral is 1. It can be considered a Gaussian in the limit of $\sigma^2 \to 0$ likelihood **Example 9.1** see p.291

Linear regression refers to models that are linear in the parameters.
Linear regression model $\Rightarrow$ linear input x.
For non linear input $\Rightarrow y = \Phi^T(x)\theta$ where Φ is also linear regression model.
From here, we assume that the noise variance σ^2 is known.

8.8 Parameter Estimation

(we want to find θ^* for the function)
Training set $\mathcal{D} := \{(x_1, y_1), \ldots, (x_N, y_N)\}$
N inputs $x_n \in \mathbb{R}^D$
Observations/targets $y_n \in \mathbb{R},\ n = 1, \ldots, N$.
As y_i and y_j are conditionally independent, we can factorize the likelihood

$$p(\mathcal{Y}|\mathcal{X}, \theta) = p(y_1, \ldots, y_N | x_1, \ldots, x_N, \theta) \tag{9.5a}$$

$$= \prod_{n=1}^{N} p(y_n|x_n, \theta) = \prod_{n=1}^{N} \mathcal{N}(y_n|x_n^T\theta, \sigma^2) \tag{9.5b}$$

Training set: $\mathcal{X} := \{x_1, \ldots, x_n\}$, corresponding targets $\mathcal{Y} := \{y_1, y_N\}$

8.8.1 Maximum Likelihood Estimation

Maximizing the likelihood means maximizing the predictive distribution of the (training) data given the parameters. The likelihood is not a probability distribution in the parameters.
Design matrix

Maximum Likelihood Estimation with Features: Since "linear regression" only refers to "linear in the parameters", we can perform an arbitrary nonlinear transformation $\Phi(x)$ of the inputs x and then linearly combine the components of this transformation.
Φ : Feature vector

Estimating the Noise Variance σ^2 : We follow the standard procedure: We write down the log-likelihood, compute its derivative with respect to $\sigma^2 > 0$, set it to 0,
The maximum likelihood estimate of the noise variance is the empirical mean of the squared distances between the noise-free function values $\Phi^T(x_n)\theta$ and the corresponding noisy observations y_n at input locations x_n .