# Book

## Part I

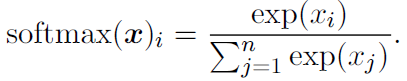
### Chapter 4 – Numerical Computation – p.95

#### 4.1 Overflow and Underflow – p.95

Rounding error is problematic, when it compounds across many operations and can cause algorithms that work in theory to fail in practice if they are not designed to minimize the accumulation of rounding error.

* *Underflow* occurs when numbers near zero are rounded to zero. Many functions behave qualitatively differently when their argument is zero rather than a small positive number. For example, we usually want to avoid division by zero or taking the log of zero (both can return not-a-number value).
* *Overflow* occurs when numbers with large magnitude are approximated as *∞* or *−∞* (which results in not-a-number value).

One a function that must be stabilized against underflow and overflow is the softmax function.



…

Usually, we don’t mind about this much because developers of low-level libraries should keep numerical issues in mind when implementing deep learning algorithms.

#### 4.2 Poor Conditioning – p.97

Conditioning refers to how rapidly a function changes with respect to small changes in its inputs. Functions that change rapidly when their inputs are perturbed slightly can be problematic for scientific computation because rounding errors in the inputs can result in large changes in the output.

…

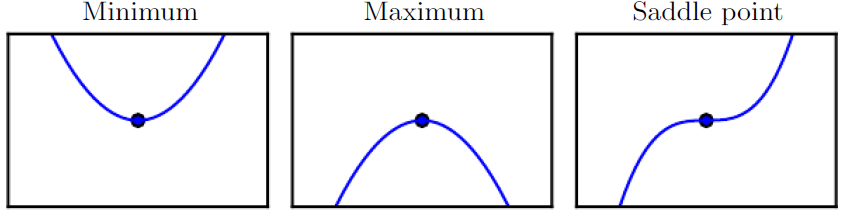
#### 4.3 Gradient-Based Optimization – p.97

The function we want to minimize or maximize is called the **objective function** or **criterion**. When we are minimizing it, we may also call it the **cost function**, **loss function**, or **error function**.

We denote the value that minimizes or maximizes a function with a superscript *∗*. For example, *x∗* = argmin *f*(*x*).

…

Points where *f’*(*x*) = 0 are known as **critical points** or **stationary points**.



Some critical points are neither maxima nor minima. These are known as saddle points.

*4.3.1 Beyond the Gradient: Jacobian and Hessian Matrices – p.101*

…

#### 4.4 Constrained Optimization – p.108

Sometimes we wish not only to maximize or minimize a function *f*(*x*) over all possible values of *x*. Instead we may wish to find the maximal or minimal value of *f* (*x*) for values of *x* in some set S. This is known as **constrained** **optimization**. Points *x* that lie within the set S are called **feasible** points in constrained optimization terminology.

### Chapter 5 – Machine Learning Basics – p.113

#### 5.1 Learning Algorithms – p.114

*5.1.1 The Task, T*

* *Classification:*
* *Classification with missing inputs:* … When some of the inputs may be missing, rather than providing a single classification function, the learning algorithm must learn a *set* of functions. Each function corresponds to classifying *x* with a different subset of its inputs missing. This kind of situation arises frequently in medical diagnosis, because many kinds of medical tests are expensive or invasive.
* *Regression:*
* *Transcription:* In this type of task, the machine learning system is asked to observe a relatively unstructured representation of some kind of data and transcribe it into discrete, textual form.
* *Machine translation:*
* *Structured output:*
* *Anomaly detection:* the computer program sifts through a set of events or objects, and flags some of them as being unusual or atypical. An example of an anomaly detection task is credit card fraud detection. By modeling your purchasing habits, a credit card company can detect misuse of your cards.
* *Synthesis and sampling:* In this type of task, the machine learning algorithm is asked to generate new examples that are similar to those in the training data.
* *Imputation of missing values:* d the machine learning algorithm is given a new example but with some entries missing. The algorithm must provide a prediction of the values of the missing entries.

…

#### 5.2 Capacity, Overfitting and Underfitting – p.125

We can control whether a model is more likely to overfit or underfit by altering its **capacity** (it’s another way of saying complexity). Informally, a model’s capacity is its ability to fit a wide variety of functions. Models with low capacity may struggle to fit the training set. Models with high capacity can overfit by memorizing properties of the training set that do not serve them well on the test set (See?).

One way to control the capacity of a learning algorithm is by choosing its **hypothesis space**, the set of functions that the learning algorithm is allowed to select as being the solution….

*5.2.1 The No Free Lunch Theorem – p.131*

…

*5.2.2 Regularization – p.133*

…

#### 5.3 Hyperparameters and Validation Sets – p.135

…

#### 5.7 Supervised Learning Algorithms – p.155

…

*5.7.2 Support Vector Machines – p.156*

Similar to logistic regression in that it is driven by a linear function *x* + *b*. Unlike logistic regression, the support vector machine does not provide probabilities, but only outputs a class identity. The SVM predicts that the positive class is present when *x + b* is positive. Likewise, it predicts that the negative class is present when *x + b* is negative.

…

#### 5.11 Challenges Motivating Deep Learning – p.170

The simple machine learning algorithms described work very well on a wide variety of important problems. However, they have not succeeded in solving the central problems in AI, such as recognizing speech or recognizing objects.

…in high-dimensional spaces. Such spaces also often impose high computational costs. Deep learning was designed to overcome these and other obstacles.

*5.11.1 The Curse of Dimensionality – p.170*

Many machine learning problems become exceedingly difficult when the number of dimensions in the data is high. This phenomenon is known as the **curse of** **dimensionality**. Of particular concern is that the number of possible distinct configurations of a set of variables increases exponentially as the number of variables increases.

…

*5.11.2 Local Constancy and Smoothness Regularization – p.172*

*5.11.3 Manifold Learning – p.176*

An important concept underlying many ideas in machine learning is that of a manifold.

## Part II Deep Networks: Modern Practices – p.181

### Chapter 6 – Deep Feedforward Networks – p.183

**Deep feedforward networks, feedforward neural networks,** or **multilayer perceptrons** (MLPs). are the quintessential deep learning models. The goal of a feedforward network is to approximate some function *f ∗*. For example, for a classifier, *y* = *f ∗*(*x*) maps an input *x* to a category *y*. A feedforward network defines a mapping *y* = *f* (*x*; *θ*) and learns the value of the parameters *θ* that result in the best function approximation.

…

One way to understand feedforward networks is to begin with linear models and consider how to overcome their limitations. The obvious defect that the model capacity is limited to linear functions, so the model cannot understand the interaction between any two input variables.

To extend linear models to represent nonlinear functions of *x*, we can apply the linear model not to *x* itself but to a transformed input *φ*(*x*), where *φ* is a nonlinear transformation. Equivalently, we can apply the kernel trick described in section 5.7.2, to obtain a nonlinear learning algorithm based on implicitly applying the *φ* mapping. We can think of *φ* as providing a set of features describing *x*, or as providing a new representation for *x*.

The question is then how to choose the mapping *φ*:

1. One option is to use a very generic *φ*, such as the infinite-dimensional *φ* that is implicitly used by kernel machines based on the RBF kernel. …
2. Another option is to manually engineer *φ*. …
3. The strategy of deep learning is to learn *φ*. In this approach, we have a model *y* = *f*(*x*; *θ,w*) = . We now have parameters *θ* that we use to learn *φ* from a broad class of functions, and parameters *w* that map from *φ*(*x*) to the desired output. … This approach is the only one of the three that gives up on the convexity of the training problem, but the benefits outweigh the harms.

**6.1 Example: Learning XOR – p.186**