Chapter 11

Stochastic Grammars and Linguistics

11.1 Introduction to Formal Grammars

In this chapter we explore one final class of probabilistic models for sequences, stochastic grammars. The basic idea behind stochastic grammars is a direct extension of the simple dice model of chapter 3 and of HMMs.

As briefly mentioned in chapter 1, formal grammars were originally developed to model natural languages, around the same time that the double-helical structure of DNA was elucidated by Watson and Crick. Since then, grammars have been used extensively in the analysis and design of computer languages and compilers [3]. Grammars are natural tools for modeling strings of letters and, more recently, they have been applied to biological sequences. In fact, many problems in computational molecular biology can be cast in terms of formal languages [91, 479]. Here, language!formalthe basic goal again is to produce, by machine learning, the corresponding grammars from the data.

Next, we review the rudiments of the theory of formal grammars, including different classes of grammars, their properties, the Chomsky hierarchy, and the connection to HMMs. In section 11.3, we demonstrate how stochastic grammars can be applied to biological sequences, and especially the application of context-free grammars to RNA molecules. In the subsequent three sections we consider priors, likelihoods, and learning algorithms. Finally, in the last two sections we cover the main applications.

11.2 Formal Grammars and the Chomsky Hierarchy

11.2.1 Formal Languages

We begin with an alphabet A of letters. The set of all finite strings over A is denoted by A^*. $\varnothing$ denotes the empty string. A *language* is a subset of A^*. In this trivial sense, we can say that promoters or acceptor sites in intervening sequences form a language over the DNA alphabet. Such a definition by itself is not very useful unless we define simple ways of generating, recognizing, and classifying languages. A grammar can be seen as a compact set of rules for generating a language. language!formal

11.2.2 Formal Grammars

A formal grammar is a set of rules for producing all the strings that are syntactically correct, and only those strings. A *formal grammar* G consists of an alphabet A of letters, called the terminal symbols; a second alphabet V of variables, also called nonterminal symbols; and a set R of production rules. Among the nonterminal symbols, there is a special s = *start* variable. Each production rule in R consists of a pair (α, β), more commonly denoted $\alpha \to \beta$, where α and β are elements of $(A \cup V)^*$. The arrow in $\alpha \to \beta$ can be read as "produces" or "expands into." We use Greek letters to denote strings that could be combinations of nonterminal and terminal symbols. Thus, in the most general case, α and β are strings made up of letters and variables. In addition, we will assume that α contains at least one nonterminal symbol. Given G and two strings γ and δ over $(A \cup V)$, we say that δ can be *derived* from γ if there is a finite sequence of strings $\pi = \alpha_1, \ldots, \alpha_n$ such that $\gamma \to \alpha_1 \to \ldots \to \alpha_n \to \delta$ (also denoted $\gamma \to_\pi \delta$), each step corresponding to an application of a production rule in R. The language $L = L(G)$ *generated* by the grammar G is the set of all terminal strings that can be derived from the start state s.

As an example, let us consider the grammar defined by $A = \{\mathsf{X}, \mathsf{Y}\}$, $V = \{s\}$, and $R = \{s \to \mathsf{X}s\mathsf{X}, s \to \mathsf{Y}s\mathsf{Y}, s \to \mathsf{X}, s \to \mathsf{Y}, s \to \varnothing\}$. The string XYYX can be derived from the string s: $s \to \mathsf{X}s\mathsf{X} \to \mathsf{XY}s\mathsf{YX} \to \mathsf{XYYX}$, by applying the first, second, and fourth production rules in succession. More generally, it is easy to show that G generates the set of all palindromes over A. Palindromes are strings that can be read identically in the forward and backward directions. We can now define several different types of grammars and the Chomsky hierarchy. The Chomsky hierarchy is a classification of grammars by increasing degrees of complexity and expressive power.

11.2.3 The Chomsky Hierarchy

The Chomsky hierarchy and its properties are summarized in table 11.1.

Regular Grammars

One of the simplest classes of grammars is the regular grammars (RGs). In a regular grammar, the left-hand side of a production rule is a single variable, and the right-hand side is typically a single letter of the alphabet followed by at most a single variable. Thus strings can grow in only one direction. More precisely, a grammar G is *regular* (or right-linear) if all the production rules are of the form $u \to \mathsf{X}v$, or $u \to \mathsf{X}$, or $u \to \emptyset$, where u and v are single nonterminal symbols. A language is regular if it can be generated by a regular grammar. language!regularRegular languages can also be described by other means—for instance, in terms of regular expressions. Regular languages can be recognized very efficiently, although their expressive power is limited.

Context-free Grammars

Regular grammars are special cases of context-free grammars (CFGs), that is, grammars where the replacement of a variable by an expression does not depend on the context surrounding the variable being replaced. More precisely, a grammar G is *context-free* if all the production rules in R are of the form $u \to \beta$, where u is a single nonterminal symbol. A language is language!context-freecontext-free if it can be generated by a context-free grammar. Context free-grammars can be expressed in canonical forms, also called normal forms, such as the "Chomsky normal form" or the "Greibach normal form." A context-free grammar is said to be in Chomsky normal form if each production rule has one of the following three forms: (1) $s \to \emptyset$; (2) $u \to vw$, where u, v, and w are nonterminal symbols; (3) $u \to \mathsf{X}$. In addition, if $s \to \emptyset$ is in R, then v and w in (2) must be different from s.

The palindrome grammar above is context-free but not regular. Context-free grammars are often used to specify the syntax of computer languages and to build compilers. As can be expected, not all languages are context-free. For example, copy languages are not context-free. A *copy* language consists of all the strings where the second half is a copy of the first half. XXYXXY belongs to a copy language (corresponding to direct repeats in DNA). Although copy languages may appear similar to palindromes, they really require a more complex class of grammars. Context-free grammars have also been used to model natural languages, but with limited success because natural languages are not context-free.

	Regular	Context-free	Context-sensitive	Recursively enumerable
Production rules	$u \to \mathsf{X}v$ $u \to \mathsf{X}$	$u \to vw$ $u \to \mathsf{X}$	$\alpha\mathsf{X}\gamma \to \alpha\beta\gamma$	All
Closure properties	$\cup, ., \star$ $\cap, ^-$	$\cup, ., \star$ no $\cap$, no $^-$	$\cup, ., \star$	All
Automata equivalence	Finite-state automata	Pushdown automata	Bounded tape Turing machine	Turing machine
Characteristic language		Palindromes	Copy language	All
Characteristic dependencies	No long-range	Nested	Crossing	All

Table 11.1: The Zoo of Grammars and Their Associated Production Rules and Equivalence Relations.

Context-sensitive Grammars

Within the grammars that are not context-free, we can define the subclass of context-sensitive grammars (CSGs). A grammar G is *context-sensitive* if all the production rules are of the form $\alpha\mathsf{X}\gamma \to \alpha\beta\gamma$ for X in A, $\beta \neq \emptyset$. (X can be replaced by β in the context $\alpha - \gamma$.) In addition, the single rule $s \to \emptyset$ is allowed, provided s does not appear on the right-hand side of any other production rule. A language is context-sensitive if it can be generated by a context-sensitive grammar. It can be shown that copy languages are context-sensitive but not context-free. Context-sensitive languages are characterized by grammars in which the right-hand side of the production rules is at least as long as the left-hand side.

Recursively Enumerable Grammars

These are the most general grammars, without any of the restrictions above. Recursively enumerable refers to the fact that if a word is in the corresponding language, its derivation can always be obtained on a Turing machine in finite time, simply by listing all possible (countable) derivations. Recursively enumerable is weaker than recursive: in general membership of a word in a language cannot be established in finite time, as in the classical halting problem. The term "Chomsky hierarchy" refers to the theorem that the main classes of grammars we have seen so far form a strictly increasing sequence. That is,

$$\text{RGs} \subset \text{CFGs} \subset \text{CSGs} \subset \text{REGs}, \tag{11.1}$$

where all the inclusions are strict and RGs = regular grammars, CFGs = context-free grammars, CSGs = context-sensitive grammars and REGs = recursively enumerable grammars. Going up the Chomsky hierarchy allows one to have more general rules, but also more restrictions on the language by excluding more strings.

11.2.4 Ambiguity and Parsing

A derivation can be arranged in a tree structure, called a *parse tree*, that reflects the syntactic structure of a sequence. Parsing can be done top down or bottom up. A sequence is *ambiguous* if it admits more than one parse tree. The notion of ambiguity is important for compilers. Ambiguity introduces complexity in parsing, both in the parsing algorithm and because the number of parse trees may grow exponentially with the length of the string being parsed. There are algorithms and complexity results for parsing specific grammars. A grammar is said to be *linear* if the right-hand sides of all the production rules contain at most one nonterminal symbol. Fast parsing algorithms exist for linear context-free grammars. In general, language recognition and sequence parsing become more computationally demanding as one ascends the Chomsky hierarchy.

11.2.5 Closure Properties

Each of the grammar classes in the Chomsky hierarchy is closed or stable under a number of language operations, such as union ($L_1 \cup L_2$), concatenation ($L_1.L_2$), and iteration (L_1^*). Regular languages are also closed under complement ($\bar{L}_1$) and intersection ($L_1 \cap L_2$). Context-free languages are not closed under complement or intersection.

11.2.6 Dependencies

Two additional ways of looking at grammars are in terms of the patterns they can generate and in terms of automata. Regular grammars can generate overall patterns, such as alternating strings like XYXYXYXY. Like HMMs, regular grammars cannot handle long-range dependencies in a string. Context-free grammars can model certain simple long-range dependencies, that is, *nested* dependencies. A pattern of dependencies is *nested* if it can be drawn without having two lines cross each other. Nested dependencies are characteristic of context-free languages such as palindromes, where the first letter must match the last one, the second must match the second to last, and so on. When dependencies cross, as in a copy language, a context-sensitive language is necessary because crossing dependencies can be implemented only with the freedom of movement that nonterminals enjoy during context-sensitive derivation.

11.2.7 Automata

A final way of understanding the Chomsky hierarchy is to look at the automata associated with each language. Without going into details, regular languages

correspond to finite state automata (FSA), with typically one state per nonterminal symbol in the grammar, as in HMMs. In such automata, there is no storage facility apart from the states themselves: everything must be hardwired. Context-free languages correspond to pushdown automata (PDA), which are like finite-state automata but with a memory stack. Only the top of the stack is accessible at any one time. This one-place memory holder is used for palindromes by pushing in, and popping off, one symbol at a time. Such automata cannot handle crossing dependencies because they can access only the top of the stack at any one time. Context-sensitive languages are associated with Turing machines with a linearly bounded tape, that is, with tape length proportional to the I/O strings. Left and right movements along the tape are needed for copying and for handling crossing dependencies. Finally, general languages, that is recursively enumerable languages, correspond to Turing machines (TMs) with unbounded tape, that is, the standard model of universal computers.

11.2.8 Stochastic Grammars and HMMs

So far we have considered deterministic grammars. Stochastic grammars are obtained by superimposing a probability structure on the production rules. Specifically, each production rule $\alpha \to \beta$ is assigned a probability $\mathbf{P}(\alpha \to \beta)$, so that $\sum_\beta \mathbf{P}(\alpha \to \beta) = 1$. A stochastic grammar is therefore characterized by a set of parameters w and can be viewed as a probabilistic generative model for the corresponding language (i.e., the language associated with the underlying deterministic grammar).

By now, it should be clear to the reader that HMMs can be viewed exactly as stochastic regular grammars (SRGs). To see this, it suffices to replace the transition from a state s_j to a state s_i in an HMM, together with the emission of the alphabet letter X, with the SRG production rule $s_j \to \mathsf{X}s_i$ with associated probability $t_{ij}e_{i\mathsf{X}}$. Stochastic context-free grammars (SCFGs) then form a more general class of models. They are used in the following sections to model the structure of RNA sequences, and can also be viewed as further generalizations of the dice models of chapter 3. SCFGs include a type of die that has two letters on each face. In the simplest RNA models, the two letters reflect base complementarity. Thus some of the RNA dice have four faces, just like a simple DNA die, but the letters on the faces are AU, UA, CG, and GC (excluding GU, UG pairs) (see figure 11.1).

Figure 11.1: Illustration of the Complementarity in the Watson–Crick Base Paring in DNA. In RNA uracil (U) replaces thymine (T).

11.2.9 Graph Grammars

So far we have considered grammars over alphabets of letters. It is possible, however, to consider more general alphabets where the "letters" are graphs or pixel configurations in image processing. In the case of graph grammars (see [165, 158] and other papers in the same volume), one must carefully specify how graphs are to be joined to each other during the derivation process. Graph grammars have considerable expressive power, and could be a natural candidate for modeling secondary and tertiary structures of biological macromolecules. Little work, however, has been done in this direction so far; a key problem is the lack of efficient learning algorithms for general (or even restricted) graph grammars.

11.3 Applications of Grammars to Biological Sequences

Ultimately, one would like to derive grammar models all the way up to the scale of genes, chromosomes, and even genomes. After all, genomes represent only a very small fraction of all the possible DNA sequences of comparable length. But to begin with, one must consider simpler examples, associated with smaller grammars, such as RNA secondary structure and palindromes.

11.3.1 RNA Secondary Structure and Biological Palindromes

RNA Secondary Structure

Many components in biological macromolecules consist of RNA. Important RNA families include transfer RNA (tRNA), ribosomal RNA (rRNA), small nuclear RNA in the spliceosome (snRNA), messenger RNA (mRNA), and various classes of introns. New phylogenies of small RNA molecules can also be selected in vitro for particular functions, such as protein binding or catalysis [109, 356, 469, 55].

Although RNA normally is single-stranded, helices formed by complementary base pairing strongly control how the RNA folds to form a distinctive 3D structure. The folding of an RNA chain into a functional molecule is largely determined by the Watson–Crick pairs A–U and G–C, but also to some extent by G–U and, more rarely, G–A pairs. RNA nucleotides interact to form secondary structure motifs such as stems, loops, bulges, and pseudoknots where otherwise unpaired nucleotides far from each other in the sequence interact [573]. These pairings often have a nested structure and cannot be modeled efficiently using a regular language or HMMs. We first consider the case of biological palindromes in RNA and other molecules.

Biological Palindromes

There are many examples of RNA/DNA palindromes associated, for example, with protein binding sites. Biological palindromes are slightly different from the ones described above because the letters are not identical when they are matched pairwise, starting from both ends, but complementary. For example, AGAUUUCGAAAUCU is an RNA palindrome. In DNA such palindromes are called inverted repeats.

Because of the complementary double-helix structure of DNA, each half of the palindrome on one side of the helix has a mirror image on the other strand. Thus, if a palindrome string is read from left to right on one strand, the same string can be read from right to left on the opposite strand. RNA palindromes can have arbitrary lengths, and therefore it is likely that they

need to be modeled by context-free or more complex grammars (technically, palindromes with a fixed upper length can be modeled by a regular grammar). RNA palindromes are typically folded so that they create hairpin (stem-loop) structures.

A grammar for RNA palindromes is given by

$$s \to \mathsf{A}s\mathsf{U} \mid \mathsf{U}s\mathsf{A} \mid \mathsf{C}s\mathsf{G} \mid \mathsf{G}s\mathsf{C} \mid \varnothing, \tag{11.2}$$

where we have listed all the alternative production rules in one line separated by a "|". A palindrome can be generated by: $s \to \mathsf{A}s\mathsf{U} \to \mathsf{AG}s\mathsf{CU} \to \mathsf{AGU}s\mathsf{ACU}$, etc. The parse tree produced can be drawn to reflect the base pairing (see also figure 11.2). Real RNA palindromes are not as perfect, but an occasional mismatched pair does not destroy the secondary structure. Some alternative pairings, such as UG, are more tolerated than others; hence also the need to introduce probabilities. It is also common for the stem of a hairpin to have bulges of unpaired bases. RNA is usually not flexible enough to make a 180° turn at the tip of the hairpin. There is often a loop of at least three to four unpaired bases, and sometimes the loop is much longer. Likewise, in DNA palindromes the two halves of the relevant palindrome can be separated by significant distances. All such features can be incorporated into a grammar but complicate the rules.

The previous grammar can generate strings corresponding to single palindromes. Both DNA and RNA are rich in compound palindromes, that is, sequential and recursive ones. Sequential palindromes occur when two or more palindromes follow each other side by side. Recursive palindromes occur when one palindrome is nested within another. The secondary RNA structure associated with a recursive palindrome is a stem with another stem budding from its side. Obtaining simple recursive palindromes is surprisingly easy: one needs only to add the production rule of the form $s \to ss$. Duplicating the variable s allows for the start of a new palindrome anywhere within an existing one. The corresponding grammar generates structures of branched stems, known as orthodox secondary structures. The best-known example is perhaps the cloverleaf structure of transfer RNA. There are many other examples, especially in ribosomal RNA, of structures consisting of combinations of loops and nested stems. The grammar of recursive palindromes is context-free but, unlike the grammar of simple palindromes, it is ambiguous. The double inverted repeat UGAUCA–UGAUCA can be parsed as a single hairpin, but also as two or more side-by-side stems, not necessarily of equal length. The alternative parse tree corresponds to alternative secondary structures. There are known cases where structural ambiguity seems to be used to confer different roles on the same RNA element. Other examples of ambiguity in DNA linguistics include overlapping genes—in HIV viruses, where some segments of the

genome can encode more than one gene, using ambiguous starting points and reading frames.

11.3.2 Context-free Grammars for RNA

More generally, the types of rules needed for an SCFG for RNA are the following:

1. Pair emission rules, for Watson–Crick pairs

$$u \to \mathsf{A}v\mathsf{U} \mid \mathsf{U}v\mathsf{A} \mid \mathsf{C}v\mathsf{G} \mid \mathsf{G}v\mathsf{C}, \tag{11.3}$$

   but also for rarer pairs (in order of rarity)

$$u \to \mathsf{G}v\mathsf{U} \mid \mathsf{G}v\mathsf{A}. \tag{11.4}$$

2. Single-letter left emissions (unpaired bases)

$$u \to \mathsf{A}v \mid \mathsf{C}v \mid \mathsf{G}v \mid \mathsf{U}v. \tag{11.5}$$

3. Single-letter right emissions (unpaired bases)

$$u \to v\mathsf{A} \mid v\mathsf{C} \mid v\mathsf{G} \mid v\mathsf{U}. \tag{11.6}$$

4. Single-letter emissions (unpaired bases)

$$u \to \mathsf{A} \mid \mathsf{C} \mid \mathsf{G} \mid \mathsf{U}. \tag{11.7}$$

5. Branching (or bifurcation)

$$u \to vw. \tag{11.8}$$

6. Deletions (or skips)

$$u \to v. \tag{11.9}$$

The nonterminal variables on the left-hand side of the production rules, such as u, play the role of HMM states and must be numbered $u_1, u_2, \ldots$. As with HMMs, these nonterminal variables can be partitioned into three classes: match, insert, and delete or skip, each with different distributions. *Match* corresponds to important columns in RNA multiple alignments. The main difference from HMMs is the possibility for some states to emit two paired symbols. For a nonterminal u associated with an insert state, a production rule of the form $u \to \mathsf{X}u$ allows multiple insertions. These are needed in loop regions to adjust the loop length. An example of CFG RNA grammar adapted from [460]

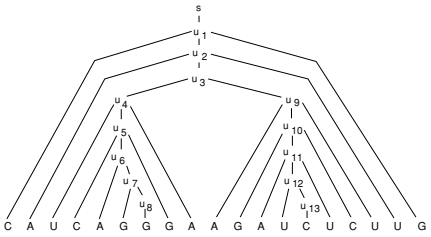
a. Productions

$$
\begin{array}{llllll}
P = \{ & s & \rightarrow & u_1, & u_7 \rightarrow \mathsf{G}\ u_8, & \\
 & u_1 & \rightarrow & \mathsf{C}\ u_2\ \mathsf{G}, & u_8 \rightarrow \mathsf{G}, & \\
 & u_1 & \rightarrow & \mathsf{A}\ u_2\ \mathsf{U}, & u_8 \rightarrow \mathsf{U}, & \\
 & u_2 & \rightarrow & \mathsf{A}\ u_3\ \mathsf{U}, & u_9 \rightarrow \mathsf{A}\ u_{10}\ \mathsf{U}, & \\
 & u_3 & \rightarrow & u_4\ u_9, & u_{10} \rightarrow \mathsf{C}\ u_{10}\ \mathsf{G}, & \\
 & u_4 & \rightarrow & \mathsf{U}\ u_5\ \mathsf{A}, & u_{10} \rightarrow \mathsf{G}\ u_{11}\ \mathsf{C}, & \\
 & u_5 & \rightarrow & \mathsf{C}\ u_6\ \mathsf{G}, & u_{11} \rightarrow \mathsf{A}\ u_{12}\ \mathsf{U}, & \\
 & u_6 & \rightarrow & \mathsf{A}\ u_7, & u_{12} \rightarrow \mathsf{U}\ u_{13}, & \\
 & u_7 & \rightarrow & \mathsf{U}\ u_7, & u_{13} \rightarrow \mathsf{C} & \}
\end{array}
$$

b. Derivation

$$
\begin{aligned}
s &\Rightarrow u_1 \Rightarrow \mathsf{C}u_2\mathsf{G} \Rightarrow \mathsf{CA}u_3\mathsf{UG} \Rightarrow \mathsf{CA}u_4u_9\mathsf{UG} \\
&\Rightarrow \mathsf{CAU}u_5\mathsf{A}u_9\mathsf{UG} \Rightarrow \mathsf{CAUC}u_6\mathsf{GA}u_9\mathsf{UG} \\
&\Rightarrow \mathsf{CAUCA}u_7\mathsf{GA}u_9\mathsf{UG} \Rightarrow \mathsf{CAUCAG}u_8\mathsf{GA}u_9\mathsf{UG} \\
&\Rightarrow \mathsf{CAUCAGGGA}u_9\mathsf{UG} \\
&\Rightarrow \mathsf{CAUCAGGGAA}u_{10}\mathsf{UUG} \\
&\Rightarrow \mathsf{CAUCAGGGAAG}u_{11}\mathsf{CUUG} \\
&\Rightarrow \mathsf{CAUCAGGGAAGA}u_{12}\mathsf{UCUUG} \\
&\Rightarrow \mathsf{CAUCAGGGAAGAU}u_{13}\mathsf{UCUUG} \\
&\Rightarrow \mathsf{CAUCAGGGAAGAUCUCUUG}
\end{aligned}
$$

c. Parse tree



d. Secondary Structure

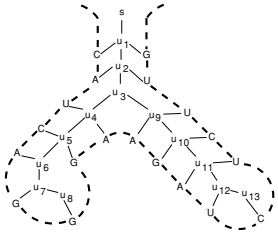


Figure 11.2: The Simple Context-free Grammar and the Derivation of the Particular Sequence CAUCAGGGAAGAUCUCUUG. A. Set of production rules of the grammar, where s is the start symbol and u_1 to u_{13} are nonterminals. B. Derivation. C. Parse tree associated with the derivation. D. Secondary structure reflecting the parse tree. Adapted from [460].

is given in figure 11.2, with the derivation of a sequence, its parse tree, and secondary structure.

The list of rule types given above is of course redundant, and not all combinations of rule types and nonterminal classes are needed to model RNA. In spite of their different name, the RNA covariance models of [156] are essentially equivalent to the SCFG models. The models in [156] use only the following:

- Match states with pair, single left, and single right emissions
- Insert states with single left and single right emissions
- Delete and branching states.

There are of course trade-offs among the richness of a grammar, the time it takes to train it, and whether it underfits or overfits the data.

11.3.3 Beyond Context-free Grammars

So far we have remained in the realm of context-free grammars, pushdown automata, and nested dependencies. Many of the simple evolutionary oper-

ations, such as insertions, deletions, and substitutions, can be expressed in isolation by context-free production rules. There are, however, other genetic operations on blocks of nucleic acid strings—such as duplications, inversions, translocations, and transpositions—that lead to the crossing of dependencies and therefore cannot be accounted for properly in a context-free style. Direct repeats in DNA are fairly common, and essentially form a copy language. As such, they can be modeled by context-sensitive grammars. Crossing of dependencies is also seen in the secondary and tertiary structures of biological molecules. One example here is the pseudoknots occurring in RNA structures.

As already mentioned, pseudoknots occur when a single-stranded loop region forms Watson–Crick base pairs with a complementary sequence outside the loop. Pseudoknots can be viewed as palindromes that are interleaved rather than nested. For instance, AACCGGUU can be regarded as the nesting of two palindromes: AAUU and CCGG. On the other hand, AACCUUGG is a pseudoknot because the complementary pairings must cross each other. Features such as pseudoknots are referred to as non-orthodox secondary structures. The previous context-free grammars are not sufficient to model pseudoknots. Pseudoknots, like direct repeats, can be described using context-sensitive grammars. Finally, it must be noted that the language of DNA may be viewed as the superposition or intersection of several other languages—for instance, for transcription, splicing, and translation. Even if each individual language were context-free, we have seen that there is no reason for the intersection to be context-free.

11.4 Prior Information and Initialization

11.4.1 Learning the Grammar Rules and Initialization from Multiple Alignments

All the rules in an SCFG, as well as their probabilities, can easily be derived from a multiple alignment when one is available, as in the case of HMMs, and with the same caveats. In [156], an algorithm is reported by which the production rules themselves are derived from a set of unaligned sequences. For large RNA molecules, the process of constructing the grammar can also be hierarchically decomposed, whereby a high-level grammar (called metagrammar in [460]) is first constructed on the basis of secondary-structure large-scale motifs [502], such as helices and loops. Each motif is then separately represented by a set of SCFG rules.

		3' A	3' C	3' G	3' U
5'	A	0.160097	0.135167	0.192695	1.590683
5'	C	0.176532	0.134879	3.403940	0.162931
5'	G	0.219045	1.718997	0.246768	0.533199
5'	U	2.615720	0.152039	0.249152	0.249152

Table 11.2: Helix Pseudocounts Are Added to Actual Observed Frequencies to Reflect Prior Information. The 16 parameters in the Dirichlet prior were computed from distributions of basepaired positions in a large alignment of 16S rRNA sequences [346]. From the alignment a four-parameter Dirichlet prior for nucleotide distributions in loop regions was made as well: A (0.26); C (0.21); G (0.18); U (0.20).

11.4.2 Dirichlet Priors

Dirichlet priors are the natural choice for the production rules of stochastic grammars. In the list in section 11.3.2, there are two main different types of rules that need to be considered: the pair emission rules $u \to \mathtt{X}v\mathtt{Y}$ and the singlet emission rules of the form $u \to \mathtt{X}v$ for loop regions. In the case of RNA, there are 16 (resp. 4) possible versions of the first (resp. second) type of rule. Because of Watson–Crick base pairing, the corresponding Dirichlet vectors are not uniform. They can easily be derived from a database of aligned RNA structures, such as [346] (table 11.2).

The other rules, such as branch production, can also be Dirichlet-regularized if necessary.

11.5 Likelihood

First, consider the problem of computing the likelihood $\mathbf{P}(O|w)$ of a sequence $O = \mathtt{X}^1 \ldots \mathtt{X}^t \ldots \mathtt{X}^T$ according to a grammar $M = M(w)$ with parameter w. Recalling that SCFGs are ambiguous, let $\pi = \alpha_1, \ldots, \alpha_n$ be a derivation of O from the start state s. Then

$$\mathbf{P}(s \to_\pi O|w) = \mathbf{P}(s \to \alpha_1|w)\mathbf{P}(\alpha_1 \to \alpha_2|w) \ldots \mathbf{P}(\alpha_n \to O|w), \tag{11.10}$$

$$\mathbf{P}(O|w) = \sum_\pi \mathbf{P}(s \to_\pi O|w). \tag{11.11}$$

These expressions are of course very similar to the ones obtained in the case of HMMs, where HMM paths are replaced by grammar derivations. Again, this expression for the likelihood is not directly practical because the number of possible parse trees is exponential in the length of the sequence. Again, this problem can be circumvented by using dynamic programming. In the case of

nonstochastic context-free grammars in Chomsky normal form, this algorithm is known as the Cocke–Kasami–Younger algorithm [393]. The derivation of a slightly more general version for stochastic context-free grammars is similar to the forward propagation algorithm for HMMs, and is left as an exercise (it is also called the "inside" algorithm). The most probable parse tree of a sequence according to an SCFG can be found by a similar version of dynamic programming that generalizes the Viterbi algorithm of HMMs. By analogy, we will use the term *Viterbi parse tree* or *Viterbi derivation*. One important point to consider is that the additional complexity of SCFGs with respect to HMMs results in three-dimensional forms of dynamic programming that scale as $O(N^3)$ rather than $O(N^2)$ (see [460, 156] for additional details).

11.6 Learning Algorithms

Learning algorithms for SCFGs of one sort or the other are described in [25, 345, 459, 460, 156]. As in the case of HMMs, the basic idea at the first level of Bayesian inference is to estimate model parameters by maximizing the likelihood or the posterior through some iterative algorithm. In most of the examples cited above, this is done by some form of the EM algorithm, although one could also use other methods, such as gradient descent. The derivation of each learning rule is only sketched because it closely parallels what has been described in detail in the case of HMMs. For the sake of simplicity, we begin with ML estimation with a single training sequence O and an SCFG with known rules and parameters w. Extensions to MAP estimation and/or multiple training sequences are straightforward. Let us consider a generic production rule of the form $u \to \beta$. For any derivation π of O, we can define $n(\beta, u, \pi, O)$ to be the number of times the rule $u \to \beta$ is used in π. Similarly, let $n(u, \pi, O) = \sum_\beta n(\beta, u, \pi, O)$.

11.6.1 The EM Algorithm

For the E step of the algorithm, we let $Q(\pi) = \mathbf{P}(\pi|O, w)$. If $P_{u\to\beta}$ denotes the probability parameter associated with the rule, then the EM reestimation equations are given by

$$P^+_{u\to\beta} = \frac{\sum_\pi Q(\pi) n(\beta, u, \pi, O)}{\sum_\pi Q(\pi) n(u, \pi, O)} = \frac{\sum_\pi \mathbf{P}(\pi|O, w) n(\beta, u, \pi, O)}{\sum_\pi \mathbf{P}(\pi|O, w) n(u, \pi, O)} = \frac{n_{u\to\beta}}{n_u}. \tag{11.12}$$

This reestimation formula is simple: all the complexity is hidden in the calculation of the numerator and the denominator. These can be calculated by a dynamic programming procedure similar to the one discussed in section 11.5,

```
          [     ]  <    D-domain    > <   Anticodon    >< Extra ><    T-domain   >[     ]
          (((((((  ((((          )))) (((((   ===  )))))         (((((       )))))))))))
1 DC0380 -GCCAAGGTGGCAGAGTTCGGCCTAACGCGGCGGCCTGCAGAGCCGCTC----ATCGCCGGTTCAAATCCGGCCCTTGGCT---
2 DA6281 -GGGCGTGTGGCGTAGTC-GGT--AGCGCGCTCCCTTAGCATGGGAGAG----GTCTCCGGTTCGATTCCGGACTCGTCCA---
3 DE2180 --GCCCCATCGTCTAGA--GGCCTAGGACACCTCCCTTTCACGGAGGCG----A-CGGGGATTCGAATTCCCCTGGGGGTA---
4 DC2440 -GGCGGCATAGCCAAGC--GGT--AAGGCCGTGGATTGCAAATCCTCTA----TTCCCCAGTTCAAATCTGGGTGCCGCCT---
5 DK1141 -GTCTGATTAGCGCAACT-GGC--AGAGCAACTGACTCTTAATCAGTGG----GTTGTGGGTTCGATTCCCACATCAGGCACCA
6 DA0260 -GGGCGAATAGTGTCAGC-GGG--AGCACACCAGACTTGCAATCTGGTA----G-GGAGGGTTCGAGTCCCTCTTTGTCCACCA
7 DA3880 -GGGGCTATAGTTTAACT-GGT--AAAACGGCGATTTTGCATATCGTTA----T-TTCAGGATCGAGTCCTGATAACTCCA---
8 DH4640 -AGCTTTGTAGTTTATGTG-----AAAATGCTTGTTTGTGATATGAGTGAAAT----------TGGAGCTT---
          (((((((  ((((          )))) (((((   ===  )))))         (((((       ))))))))))))
1 DC0380 -GCCAAGGUGGCAG.AGUUcGGccUAACGCGGCGGCCUGCAGAGCCGCUC---AUCGCCGGUUCAAAUCCGGCCCUUGGCU---
2 DA6281 -GGGCGUGUGGCGU.AGUC.GG..UAGCGCGCUCCCUUAGCAUGGGAGAGG---UCUCCGGUUCGAUUCCGGACUCGUCCA---
3 DE2180 -GCCCC-AUCGUCU.AGAG.GCc.UAGGACACCUCCCUUUCACGGAGGCG----ACGGGGAUUCGAAUUCCCCU-GGGGGU--A
4 DC2440 -GGCGGCAUAGCCA.AGC-.GG..UAAGGCCGUGGAUUGCAAAUCCUCUA---UUCCCCAGUUCAAAUCUGGGUGCCGCCU---
5 DK1141 -GUCUGAUUAGCGC.AACU.GG..CAGAGCAACUGACUCUUAAUCAGUGGG---UUGUGGGUUCGAUUCCCACAUCAGGCACCA
6 DA0260 -GGGCGAAUAGUGUcAGCG.GG..-AGCACACCAGACUUGCAAUCUGGUA----GGGAGGGUUCGAGUCCCUCUUUGUCCACCA
7 DA3880 -GGGGCUAUAGUUU.AACU.GG..UAAAACGGCGAUUUUGCAUAUCGUUA----UUUCAGGAUCGAGUCCUGAUAACUCCA---
8 DH4640 -AGCUUUGUAGUUU.A--U.GU..GAAAAUGCUUGUUUGUGAUAUGAGUGA--AAU----------UGGAGCU
```

Figure 11.3: Comparison of Multiple Alignments of Several Representative tRNAs in the Data Set (top) [502] with That Produced by the Trained Grammar **RandomTRNA618** (bottom). Parentheses indicate base paired positions; = = =, the anticodon; '[]', the 5' and 3' sides of the acceptor helix. For **RandomTRNA618**, capital letters correspond to nucleotides aligned to the match nonterminals of the grammar; lowercase letters to insertions; -, to deletions by skip productions; and ., to fill characters required for insertions. The sequences are taken from the seven groups above and are denoted by their database section codes: 1. ARCHAE (*Halobacterium cutirubrum*), 2. CY (*Saccharomyces cerevisiae*), 3. CYANELCHLORO (*Cyanophora paradoxa*), 4. CYANELCHLORO (*Chlamydomonas reinhardtii*), 5. EUBACT (*Mycoplasma capricolum*), 6. VIRUS (*Phage T5*), 7. MT (*Aspergillus nidulans*), 8. PART III (*Ascaris suum*).

and to the forward–backward algorithm of HMMs, which scales as $O(N^3)$ instead of $O(N^2)$ for HMMs, where N is the average sequence length. When the grammar is in Chomsky normal form, this is known also as the inside–outside algorithm [345]. In the case of K training sequences $O_1, \ldots, O_K$, the EM reestimation formula becomes

$$P^+_{u\to\beta} = \frac{\sum_{j=1}^{K} \sum_{\pi} \mathbf{P}(\pi | O_j, w) n(\beta, u, \pi, O_j)}{\sum_{j=1}^{K} \sum_{\pi} \mathbf{P}(\pi | O_j, w) n(u, \pi, O_j)}. \tag{11.13}$$

A version of the EM algorithm for SCFGs, called tree-grammar EM, is developed in [460]. It has the advantage of scaling as $O(N^2)$, but requires *folded* RNA as training samples. The folding structure provides more information than the raw sequence but less information than a complete parse. If a complete parse is available, one could just count the number of occurrences of each production rule. The folding structure, on the other hand, provides a skeleton tree where the leaves are labeled with the letters of the sequence, but not the interior nodes. From the skeleton one can tell which nucleotides are base paired, but one cannot directly tell whether a letter was emitted by a match or insert nonterminal symbol. The tree-grammar EM estimates the probabilities associated with the nonterminal symbols.

It is also possible to consider a global iterative training algorithm, as in [460], where at each step (1) the current grammar is used to fold the train-

Data set	Type of tRNA	Total	Zero	MT10CY10	MT100	Random
ARCHAE	archaea	103	0	0	0	50
CY	cytoplasm	230	0	10	0	100
CYANELCHLORO	cyanelle and chloroplast	184	0	0	0	100
EUBACT	eubacteria	201	0	0	0	100
VIRUS	viruses	24	0	0	0	10
MT	mitochondria	422	0	10	100	200
PART III	part III	58	0	0	0	58
Total		1222	0	20	100	618

Table 11.3: Validation Results for SCFG RNA Models of tRNA Families.

ing sequences, then (2) the folded sequences are used to optimize the grammar parameters—for instance, using tree-grammar EM. Production rules can be added or removed from the grammar, as in the algorithms for adjusting the length of a standard HMM architecture.

11.6.2 Gradient Descent and Viterbi Learning

While to the best of our knowledge only the EM algorithm has been used in the SCFG literature, it is clear that one could use other learning algorithms, such as gradient descent and Viterbi learning (simulated annealing remains too expensive for complex SCFGs).

As with HMMs, we can reparameterize a SCFG by

$$P_{u\to\beta} = \frac{e^{w_{u\to\beta}}}{\sum_\gamma e^{w_{u\to\gamma}}}. \tag{11.14}$$

The online gradient-descent learning equation is then

$$\Delta w_{u\to\beta} = \eta(n_{u\to\beta} - n_u P_{u\to\beta}), \tag{11.15}$$

where η is the learning rate. In the case of Viterbi learning for SCFGs, all counts of the form $n(\beta, u, \pi, O)$, which are averaged over all derivations π, are replaced by the counts $n(\beta, u, \pi^*, O)$ associated with the most likely derivation only. Most of the other remarks on gradient descent and Viterbi learning made in the case of HMMs apply to SCFGs, with the proper modifications. Viterbi learning from folded sequences is essentially equivalent to initializing an SCFG from a preexisting multiple alignment.

11.7 Applications of SCFGs

A trained SCFG can be used in much the same way as we used HMMs in chapters 7 and 8. For each example sequence, we can compute its Viterbi parse

Data set	ZeroTrain	MT10CY10	MT100	RandomTRNA618
ARCHAE	94.87	100.00	100.00	100.00
CY	98.28	99.76	99.89	99.87
CYANELCHLORO	96.22	99.64	99.64	99.79
EUBACT	99.69	99.86	99.86	99.86
VIRUS	96.83	100.00	100.00	100.00
MT	89.19	98.33	98.91	98.93
PART III	55.98	81.10	83.21	83.00

Table 11.4: Percentages of Base Pairs in the Original Alignment That Are Also Present in the Secondary Structure Predicted by Each of the Four Grammars.

tree. For RNA sequences, the syntactic structure or the equivalent parse tree provide a candidate for optimal folding that can be used to predict secondary structure. This approach complements previous methods for RNA secondary structure prediction based on phylogenetic analysis or thermodynamic considerations. The parse trees can also be used to derive multiple alignments where aligned columns or pairs of columns are associated with nonterminal main states. Gaps must be added in the obvious way. This is useful to determine common consensus patterns. Negative log-likelihood (or log-posterior) scores can be computed for any sequence. As in the case of HMMs, the score of a sequence depends on its length and must be normalized, as discussed in chapter 8. These scores in turn can be used to discriminate members of the family from non-members, to search databases, and possibly to discover new members of the family. In generative mode, SCFG could be used to generate new putative members of a given family, although this has not been tested. Finally, SCFGs can also be combined in modular ways. An example is discussed in [156] in which a tRNA SCFG grammar is combined with an intron grammar to search for tRNA genes.

11.8 Experiments

Here we report the validation results in [460] for SCFG RNA models of tRNA families. Similar results are described in [156]. The original data set consists of the sequences and alignments of 1222 unique tRNAs extracted from the database described in [502]. The length varies between 51 and 93 bases, and the sequences are subdivided into seven disjoint sets corresponding to different tRNA types (table 11.3).

For discrimination experiments, 2016 non-tRNA test sequences are generated from the non-tRNA features (including mRNA, rRNA, and protein coding

	Above 5 σ				Between 4 and 5 σ				Below 4 σ			
Data Set	ZT	MT10	MT100	R618	ZT	MT10	MT100	R618	ZT	MT10	MT100	R618
ARCHAE	66	103	103	103	19	0	0	0	18	0	0	0
CY	135	230	230	230	53	0	0	0	42	0	0	0
CYANELCH	61	184	184	184	52	0	0	0	71	0	0	0
EUBACT	160	201	201	201	30	0	0	0	11	0	0	0
VIRUS	16	24	24	24	4	0	0	0	4	0	0	0
MT (train)	N/A	10	99	193	N/A	0	1	6	N/A	0	0	1
MT (test)	64	389	313	218	89	10	7	3	269	13	2	1
PART III	0	9	7	29	1	15	14	8	57	34	37	21
NON-TRNA	0	0	0	0	0	0	1	1	2016	2016	2015	2015
Totals	502	1150	1161	1182	248	25	23	18	2488	2063	2054	2038

Table 11.5: Number of tRNAs in Each Family That Are Successfully Discriminated from the Non-tRNAs Using a Threshold on the Discrimination Score.

regions) in GenBank. Roughly 20 non-tRNA sequences are created for each length in the interval between 20 and 120. Four different grammars are then created. The first grammar (**ZeroTrain**) is a control that is not trained on any sequence and contains only prior information on tRNA. The other three grammars (**MT10CY10**, **MT100**, and **RandomTRNA618**) are trained from different sets as shown in table 11.3, using the tree-grammar EM algorithm. The four grammars are compared on three tasks: multiple alignments, secondary structure prediction, and discrimination.

11.8.1 Multiple Alignments

All 1222 tRNA sequences in the data set are aligned using each of the four grammars. The best multiple alignment is obtained with **RandomTRNA618**. The predicted alignment agrees substantially with the original data set alignment (figure 11.3). Boundaries of helices and loops are the same. The major difference is the extra arm, which is highly variable in both length and sequence. There are also cases [460] where the grammar alignments suggest small improvements over the original alignment.

11.8.2 RNA Secondary Structure Prediction

As for secondary structure, in most cases the Viterbi parse tree gives the correct secondary structure. Table 11.4 gives the percentages of base pairs in the original alignment that are also present in the secondary structure predicted by each grammar. For ARCHAE and VIRUS, all three trained grammars achieve 100% recognition. For CY, CYANELCHLORO, and EUBACT the agreement is also very good. In the case of PART III, it is substantially weaker.

11.8.3 Discrimination

Discrimination for each of the four grammars is tested by computing the normalized scores of all 2016 non-tRNA sequences and comparing them against the scores of the 1222 tRNA sequences in the data set. Non-tRNA rarely have a normalized score above 4, so that a discrimination threshold is set at 5. Table 11.5 summarizes the results by displaying the number of tRNAs in each family that are successfully discriminated from the non-tRNAs in this way. Some of the respective histograms are given in figure 11.4.

Training with as few as 20 sequences significantly improves the detection rates, as seen by comparing the results of **MT10CY10** and **ZeroTrain**. **MT10CY10** perfectly discriminates tRNAs from non-tRNA sequences, except for the subsets MT and PART III, where the **ZeroTrain** grammar fails. **MT10CY10** discriminates reasonably well on the MT subset but not on PART III. Setting aside PART III sequences, **MT10CY10** discriminates 399 out of 422 mitochondrial sequences, performing almost as well as the grammars trained on many more tRNA sequences. None of the grammars achieves good discrimination on PART III sequences, not even **RandomTRNA618**, which is trained in part on such sequences. Training on PART III sequences improves performance on some of these sequences, but half of them remain with a normalized score below the threshold of 5.

11.9 Future Directions

We have reviewed the basic theory of formal languages and grammars. We have seen how stochastic grammars can be applied to biological sequences by generalizing the dice model and the HMM ideas. SCFGs, in particular, and the corresponding learning algorithms have been used to derive statistical models of tRNA. The trained grammars have been used to align, fold, and discriminate tRNA sequences with good results. The SCFG approach is a viable method for determining tRNA secondary structure. It complements the two preexisting methods, one based on phylogenetic analysis of homologous sequences [186, 565, 278] and the other on thermodynamic considerations [521, 222, 527, 585]. SCFGs for RNA, however, have been less thoroughly tested than HMMs for protein families and additional work is required to establish this approach further. Whereas the SCFGs are capable of finding global structural alignments of RNA, a new dynamical programming algorithm for obtaining local structural alignments has recently been giving good results [220, 221]. This local method is an extension of the Smith–Waterman alignment scheme combined with another dynamical programming technique for finding the maximal number of complementary nucleotide pairs.

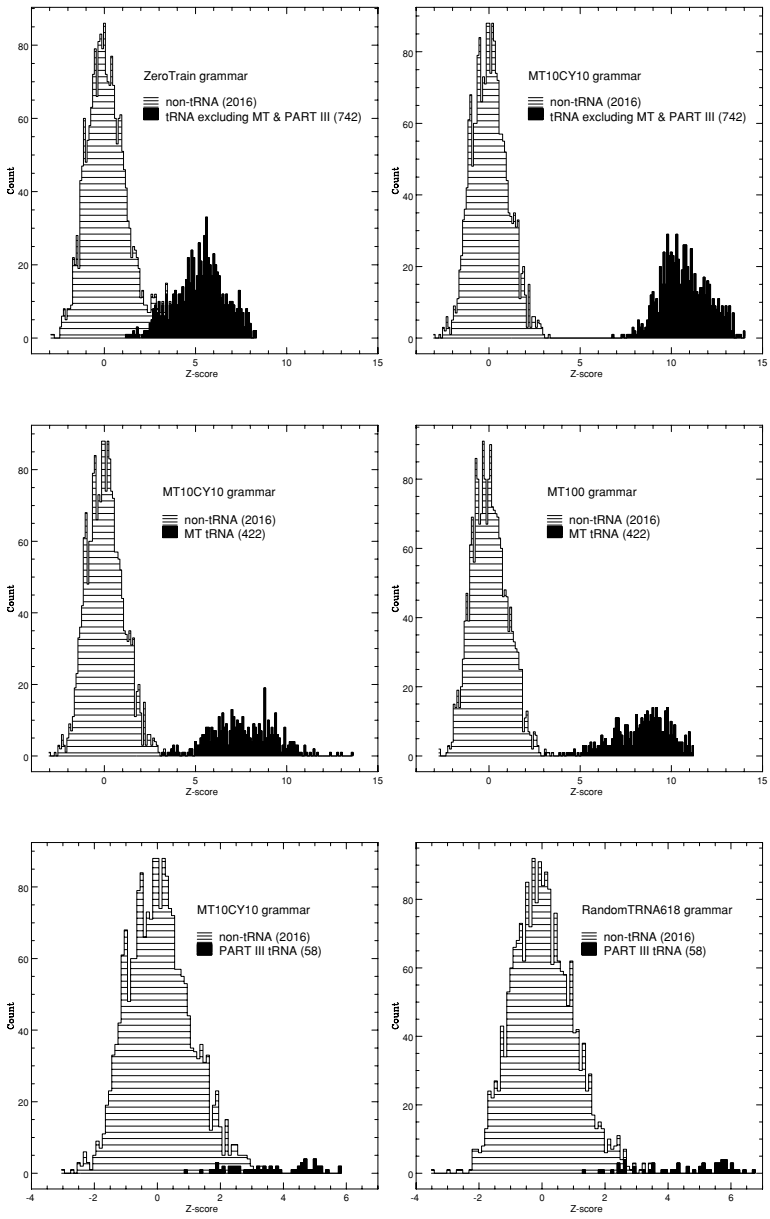


Figure 11.4: Some of the Normalized Score Histograms Showing Results of Discrimination Tests with Various Grammars.

The grammar methods described in this chapter have some limitations. First, they are computationally intensive, so that, in their present form, they become somewhat impractical for long sequences, typically above $N = 200$ or so. Second, not all RNA structures can be captured by an SCFG. The associated parse trees cannot capture tertiary interactions such as pseudoknots and non-pairwise interactions, which so far have been ignored. Third, the method as described in this chapter does not include a model for introns that are present in some tRNA genes. Such limitations point to a few obvious directions for future work, including the following:

- Algorithmic and perhaps hardware speed improvements
- Development of grammars, perhaps graph grammars, or other models, and the corresponding training algorithms to incorporate RNA tertiary structures, and possibly the tertiary structure of other molecules
- Combination of SCFGs in modular ways, as for HMMs, to model more complex RNA sequences, including the corresponding introns—work in this direction is reported in [156]
- Modeling larger and more challenging RNA sequences, such as rRNA
- Finally, along the lines of chapter 9, developing hybrid SCFG/NN architectures (or SG/NN), where an NN is used to compute the parameters of a SCFG and/or to modulate or mix different SCFGs.

This page intentionally left blank

Chapter 12

Microarrays and Gene Expression

12.1 Introduction to Microarray Data

A number of new microarray-based technologies have been developed over the last few years, and technological development in this area is likely to continue at a brisk pace. These technologies include DNA hybridization arrays (gene expression arrays, oligonucleotide arrays for sequencing and polymorphism), protein arrays, tissue arrays, and combinatorial chemistry arrays. By making possible the combinatorial interaction of a large number of molecules with a large library, these high-throughput approaches are rapidly generating terabytes of information that are overwhelming conventional methods of biological analysis. In this chapter, we focus primarily on DNA gene expression microarrays. We closely follow the derivation in [44] and show how the general probabilistic framework can be applied systematically to array data. A more complete treatment of DNA microarrays can be found in [43].

DNA gene expression microarrays allow biologists to study genome-wide patterns of gene expression [148, 160, 263] in any given cell type, at any given time, and under any given set of conditions. In these arrays, total RNA is reverse-transcribed to create either radioactive- or fluorescent-labeled cDNA that is hybridized with a large DNA library of gene fragments attached to a glass or membrane support. Phosphorimaging or other imaging techniques are used to produce expression measurements for thousands of genes under various experimental conditions. Use of these arrays is producing large amounts of data, potentially capable of providing fundamental insights into biological processes ranging from gene function to development, cancer, and

aging, and pharmacology [498, 567, 7, 217, 354, 511, 7, 554, 369, 169, 171]. Even partial understanding of the available information can provide valuable clues. For instance, co-expression of novel genes may provide leads to the functions of many genes for which information is not available currently. Data analysis techniques for microarray data, however, are still at an early stage of development [581].

Gene expression array data can be analyzed on at least three levels of increasing complexity. The first level is that of single genes, where one seeks to establish whether each gene in isolation behaves differently in a control versus a treatment situation. The second level is multiple genes, where clusters of genes are analyzed in terms of common functionalities, interactions, co-regulation, etc. The third level attempts to infer the underlying gene and protein networks that ultimately are responsible for the patterns observed.

To begin with, we assume for simplicity that for each gene X the data consists of a set of measurements $x_1^c, \ldots, x_{n_c}^c$ and $x_1^t, \ldots, x_{n_t}^t$ representing expression levels, or rather their logarithms, in both a control and treatment situation. Treatment is of course taken in a broad sense to mean any condition different from the control. For each gene, the fundamental question we wish to address is whether the level of expression is significantly different in the two situations. While it might seem that standard statistical techniques could easily address such a problem, this is in fact not the case.

One approach commonly used in the literature is a simple-minded fold approach, in which a gene is declared to have significantly changed if its average expression level varies by more than a constant factor, typically 2, between the treatment and control conditions. Inspection of gene expression data suggests, however, that such a simple "2-fold rule" is unlikely to yield optimal results, since a factor of 2 can have quite different significance in different regions of the spectrum of expression levels.

A related approach to the same question is the use of a t-test, for instance on the logarithm of the expression levels. This is similar to the fold approach because the difference between two logarithms is the logarithm of their ratio. This approach is not necessarily identical to the first because the logarithm of the mean is not equal to the mean of the logarithms; in fact, it is always strictly greater by convexity of the logarithm function. But with a reasonable degree of approximation, a test of the significance of the difference between the log expression levels of two genes is equivalent to a test of whether or not their fold change is significantly different from 1.

In a t-test, the empirical means m_c and m_t and variances s_c^2 and s_t^2 are used to compute a normalized distance between the two populations in the form

$$t = (m_c - m_t)/\sqrt{\frac{s_c^2}{n_c} + \frac{s_t^2}{n_t}} \tag{12.1}$$

where, for each population, $m = \sum_i x_i / n$ and $s^2 = \sum_i (x_i - m)^2 / (n-1)$ are the well-known estimates for the mean and standard deviation. It is well known from the statistics literature that t follows approximately a Student distribution (appendix A), with

$$f = \frac{[(s_c^2/n_c) + (s_t^2/n_t)]^2}{\frac{(s_c^2/n_c)^2}{n_c - 1} + \frac{(s_t^2/n_t)^2}{n_t - 1}} \tag{12.2}$$

degrees of freedom. When t exceeds a certain threshold depending on the confidence level selected, the two populations are considered to be different. Because in the t-test the distance between the population means is normalized by the empirical standard deviations, this has the potential for addressing some of the shortcomings of the fixed fold-threshold approach. The fundamental problem with the t-test for microarray data, however, is that the repetition numbers n_c and/or n_t are often small because experiments remain costly or tedious to repeat, even with current technology. Small populations of size $n = 1, 2$, or 3 are still very common and lead, for instance, to significant underestimates of the variances. Thus a better framework is needed to address these shortcomings.

12.2 Probabilistic Modeling of Array Data

12.2.1 Gaussian Model

Array data requires a probabilistic approach because it is highly noisy and variable, and many relevant variables remain unobserved behind the massive data sets. In order to develop a probabilistic approach for array data, the lessons learnt with sequence data are worth remembering. In sequence data, we saw in chapter 3 that the simplest probabilistic model is that of a die associated with the average composition of the family of DNA, RNA, or protein sequences under study. The next level of modeling complexity is a first-order Markov model with one die per position or per column in a multiple alignment. We have seen how, in spite of their simplicity, these models are still useful as a background, for instance, against which the performances of more sophisticated models are assessed.

In array data, the simplest model would assume that all data points are independent from one another and extracted from a single continuous distribution, for instance a Gaussian distribution. While trivial, this "Gaussian die" model still requires the computation of interesting quantities, such as the average level of activity and its standard deviation, that can be useful to calibrate or assess global properties of the data. The next equivalent level of modeling

is a set of independent distributions, one for each dimension, for instance each gene. While it is obvious that genes interact with one another in complex ways and therefore are not independent, the independence approximation is still useful and underlies *any* attempt, probabilistic or other, to determine whether expression-level differences are significant on a *gene-by-gene* basis.

Here we first assume that the expression-level measurements of a gene in a given situation have a roughly Gaussian distribution. In our experience, with common technologies this assumption is reasonable, especially for the logarithm of the expression levels, corresponding to lognormal raw expression levels. To the best of our knowledge, large-scale replicate experiments have not yet been carried out to make more precise assessments. It is clear, however, that other distributions, such as gammas or mixtures of Gaussians/gammas, could be introduced at this stage. These would impact the details of the analysis (see also [558, 403]), but not the general Bayesian probabilistic framework.

Thus, in what follows we assume that the data has been pre-processed—including taking logarithms if needed—to the point that we can model the corresponding measurements of each gene in each situation (treatment or control) with a normal distribution $\mathcal{N}(x;\mu,\sigma^2)$. For each gene and each condition, we have a two-parameter model $w = (\mu,\sigma^2)$, and by focusing on one such model we can omit indices identifying the gene or the condition. Assuming that the observations are independent, the likelihood is given by

$$\begin{aligned}\mathbf{P}(D|\mu,\sigma^2) &\approx \prod_i \mathcal{N}(x_i;\mu,\sigma^2)\\ &= C(\sigma^2)^{-n/2}\exp(-\sum_i (x_i-\mu)^2/2\sigma^2)\\ &= C(\sigma^2)^{-n/2}\exp(-(n(m-\mu)^2+(n-1)s^2)/2\sigma^2) \quad (12.3)\end{aligned}$$

where i ranges over replicate measurements. In this chapter, we write C to denote the normalizing constant of any distribution ($C = 1/Z$). The likelihood depends only on the sufficient statistics n, m, and s^2. In other words, all the information about the sample that is relevant for the likelihood is summarized in these three numbers. The case in which either the mean or the variance of the Gaussian model is supposed to be known is of course easier and is well studied in the literature [86, 431].

A full Bayesian treatment requires introducing a prior $\mathbf{P}(\mu,\sigma^2)$. The choice of a prior is part of the modeling process, and several alternatives [86, 431] are possible, a sign of the flexibility of the Bayesian approach rather than its arbitrariness. Here a conjugate prior is convenient and adequately captures several properties of DNA microarray data including, as we shall see, the fact that μ and σ^2 are generally *not* independent.

12.2.2 The Conjugate Prior

Recall that when both the prior and the posterior have the same functional form, the prior is said to be a conjugate prior. When estimating the mean alone for a normal model of known variance, the obvious conjugate prior is also a normal distribution. In the case of dice models for biological sequences, we have seen that the standard conjugate prior is a Dirichlet distribution. The form of the likelihood in (12.3) shows that the conjugate prior density must also have the form $P(\mu|\sigma^2)P(\sigma^2)$, where the marginal $P(\sigma^2)$ correponds to a scaled inverse gamma distribution (equivalent to $1/\sigma^2$ having a gamma distribution, see appendix A), and the conditional distribution $P(\mu|\sigma^2)$ is normal.

This leads to a hierarchical model with a vector of four hyperparameters for the prior $\alpha = (\mu_0, \lambda_0, \nu_0$ and $\sigma_0^2)$ with the densities

$$\mathbf{P}(\mu|\sigma^2) = \mathcal{N}(\mu; \mu_0, \sigma^2/\lambda_0) \tag{12.4}$$

and

$$\mathbf{P}(\sigma^2) = \mathcal{I}(\sigma^2; \nu_0, \sigma_0^2). \tag{12.5}$$

The expectation of the prior is finite if and only if $\nu_0 > 2$. The prior $\mathbf{P}(\mu, \sigma^2) = \mathbf{P}(\mu, \sigma^2|\alpha)$ is given by

$$C\sigma^{-1}(\sigma^2)^{-(\nu_0/2+1)} \exp\left[-\frac{\nu_0}{2\sigma^2}\sigma_0^2 - \frac{\lambda_0}{2\sigma^2}(\mu_0 - \mu)^2\right]. \tag{12.6}$$

Notice that it makes perfect sense with array data to assume a priori that μ and σ^2 are *dependent*, as suggested immediately by visual inspection of typical microarray data sets (figure 12.1). The hyperparameters μ_0 and σ^2/λ_0 can be interpreted as the location and scale of μ, and the hyperparameters ν_0 and σ_0^2 as the degrees of freedom and scale of σ^2. After some algebra, the posterior has the same functional form as the prior:

$$\mathbf{P}(\mu, \sigma^2|D, \alpha) = \mathcal{N}(\mu; \mu_n, \sigma^2/\lambda_n)\mathcal{I}(\sigma^2; \nu_n, \sigma_n^2) \tag{12.7}$$

with

$$\mu_n = \frac{\lambda_0}{\lambda_0 + n}\mu_0 + \frac{n}{\lambda_0 + n}m \tag{12.8}$$

$$\lambda_n = \lambda_0 + n \tag{12.9}$$

$$\nu_n = \nu_0 + n \tag{12.10}$$

$$\nu_n\sigma_n^2 = \nu_0\sigma_0^2 + (n-1)s^2 + \frac{\lambda_0 n}{\lambda_0 + n}(m - \mu_0)^2. \tag{12.11}$$

The parameters of the posterior combine information from the prior and the data in a sensible way. The mean μ_n is a convex weighted average of the

prior mean and the sample mean. The posterior degree of freedom ν_n is the prior degree of freedom plus the sample size. The posterior sum of squares $\nu_n\sigma_n^2$ is the sum of the prior sum of squares $\nu_0\sigma_0^2$, the sample sum of squares $(n-1)s^2$, and the residual uncertainty provided by the discrepancy between the prior mean and the sample mean.

While it is possible to use a prior mean μ_0 for gene expression data, in many situations it is sufficient to use $\mu_0 = m$. The posterior sum of squares is then obtained precisely as if one had ν_0 additional observations all associated with deviation σ_0^2. While superficially this may seem like setting the prior after having observed the data [372], a similar effect is obtained using a preset value μ_0 with $\lambda_0 \to 0$, i.e., with a very broad standard deviation so that the prior belief about the location of the mean is essentially uniform and vanishingly small. The selection of the hyperparameters for the prior is discussed in more detail below.

It is not difficult to check that the conditional posterior distribution $P(\mu|\sigma^2, D, \alpha)$ of the mean is normal $\mathcal{N}(\mu_n, \sigma^2/\lambda_n)$. The marginal posterior $P(\mu|D, \alpha)$ of the mean is Student $t(\nu_n, \mu_n, \sigma_n^2/\lambda_n)$, and the marginal posterior $P(\sigma^2|D, \alpha)$ of the variance is scaled inverse gamma $\mathcal{I}(\nu_n, \sigma_n^2)$.

In the literature, semi-conjugate prior distributions also are used where the functional form of the prior distributions on μ and σ^2 are the same as in the conjugate case (normal and scaled inverse gamma, respectively) but independent of each other, i.e. $\mathbf{P}(\mu, \sigma^2) = \mathbf{P}(\mu)\mathbf{P}(\sigma^2)$. However, as previously discussed, this assumption of independence is unlikely to be suitable for DNA microarray data. More complex priors also could be constructed using mixtures, mixture of conjugate priors leading to mixtures of conjugate posteriors.

12.2.3 Parameter Point Estimates

The posterior distribution $\mathbf{P}(\mu, \sigma^2|D, \alpha)$ is the fundamental object of Bayesian analysis and contains the relevant information about *all* possible values of μ and σ^2. However, in order to perform the t-test described above, for instance, we need to collapse this information-rich distribution into single point estimates of the mean and variance of the expression level of a gene in a given situation. This can be done in a number of ways. In general, the most robust answer is obtained using the mean of the posterior (MP) estimate. An alternative is to use the mode of the posterior, or MAP (maximum a posteriori) estimate. For completeness, we derive both kinds of estimates.

By integration, the MP estimate is given by

$$\mu = \mu_n \quad \text{and} \quad \sigma^2 = \frac{\nu_n}{\nu_n - 2}\sigma_n^2 \tag{12.12}$$

provided $\nu_n > 2$. If we take $\mu_0 = m$, we then get the following MP estimate

$$\mu = m \quad \text{and} \quad \sigma^2 = \frac{\nu_n \sigma_n^2}{\nu_n - 2} = \frac{\nu_0 \sigma_0^2 + (n-1)s^2}{\nu_0 + n - 2} \tag{12.13}$$

provided $\nu_0 + n > 2$. This is the default estimate implemented in the Cyber-T software described below. From (12.7), the MAP estimates are

$$\mu = \mu_n \quad \text{and} \quad \sigma^2 = \frac{\nu_n \sigma_n^2}{\nu_n - 1} \tag{12.14}$$

If we use $\mu_0 = m$, these reduce to

$$\mu = m \quad \text{and} \quad \sigma^2 = \frac{\nu_n \sigma_n^2}{\nu_n - 1} = \frac{\nu_0 \sigma_0^2 + (n-1)s^2}{\nu_0 + n - 1}. \tag{12.15}$$

Here the modes of the marginal posterior are given by

$$\mu = \mu_n \quad \text{and} \quad \sigma^2 = \frac{\nu_n \sigma_n^2}{\nu_n + 2}. \tag{12.16}$$

In practice, (12.13) and (12.15) give similar results and can be used with gene expression arrays. The slight differences between the two closely parallel what is seen with Dirichlet priors on sequence data in chapter 3, (12.13) being generally a slightly better choice. The Dirichlet prior is equivalent to the introduction of pseudo-counts to avoid setting the probability of any amino acid or nucleotide to zero. In array data, few observation points are likely to result in a poor estimate of the variance. With a single point ($n = 1$), for instance, we certainly want to refrain from setting the corresponding variance to zero; hence the need for regularization, which is achieved by the conjugate prior. In the MP estimate, the empirical variance is modulated by ν_0 "pseudo-observations" associated with a background variance σ_0^2.

12.2.4 Full Bayesian Treatment and Hyperparameter Point Estimates

At this stage of modeling, each gene is associated with two models $w_c = (\mu_c, \sigma_c^2)$ and $w_t = (\mu_t, \sigma_t^2)$, two sets of hyperparameters α_c and α_t, and two posterior distributions $\mathbf{P}(w_c|D, \alpha_c)$ and $\mathbf{P}(w_t|D, \alpha_t)$. A full probabilistic treatment would require introducing prior distributions over the hyperparameters. These could be integrated out to obtain the true posterior probabilities $\mathbf{P}(w_c|D)$ and $\mathbf{P}(w_t|D)$, which then could be integrated over all values of w_t and w_c to determine whether or not the two models are different. Notice that this approach is significantly more general than the plain t-test and could in principle detect interesting changes that are beyond the scope of the t-test.

For instance, a gene with the same mean but a very different variance between the control and treatment situations goes undetected by a t-test, although the change in variance might be biologically relevant. Even if we restrict ourselves to only the means μ_c and μ_t and these have a Gaussian posterior distribution, the probability $\mathbf{P}(|\mu_c - \mu_t| < \epsilon)$ must be estimated numerically. While the latter is not difficult to perform with today's computers, it is also possible to use simpler and more approximate strategies to the full Bayesian treatment that rely solely on point estimates.

Point estimates, however, entail hyperparameters that can be addressed in a number of ways [372, 375]. Here, again, one possibility is to define a prior on the hyperparameters and try to integrate them out in order to compute the true posterior $\mathbf{P}(w|D)$ and determine the location of its mode, leading to true MAP estimates of w. More precisely, this requires integrating $\mathbf{P}(w|\alpha)$ and $\mathbf{P}(w|\alpha|D)$ with respect to the hyperparameter vector α. An alternative that avoids the integration of the hyperparameters is the evidence framework described in [372]. In the evidence framework, we compute point estimates of the hyperparameters by MAP estimation (MP would again require integrating over hyperparameters) over the posterior

$$\mathbf{P}(\alpha|D) = \frac{\mathbf{P}(D|\alpha)\mathbf{P}(\alpha)}{\mathbf{P}(D)}. \tag{12.17}$$

If we take a uniform prior $\mathbf{P}(\alpha)$, then this is equivalent to maximizing the evidence $\mathbf{P}(D|\alpha)$

$$\begin{aligned} \mathbf{P}(D|\alpha) &= \mathbf{P}(D|w,\alpha)\mathbf{P}(w|\alpha)/\mathbf{P}(w|D,\alpha) \\ &= \mathbf{P}(D|w)\mathbf{P}(w|\alpha)/\mathbf{P}(w|D,\alpha). \end{aligned} \tag{12.18}$$

In principle, computing the evidence requires integrating out the parameters w of the model. Using the expression for the likelihood and the conjugate prior and posterior, however, we can obtain the evidence without integration, directly from (12.18):

$$\mathbf{P}(D|\alpha) = (2\pi)^{-n/2}\frac{\sqrt{\lambda_0}}{\sqrt{\lambda_n}}\frac{(\nu_0/2)^{\nu_0/2}}{(\nu_n/2)^{\nu_n/2}}\frac{\sigma_0^{\nu_0}}{\sigma_n^{\nu_n}}\frac{\Gamma(\nu_n/2)}{\Gamma(\nu_0/2)}. \tag{12.19}$$

The partial derivatives and critical points of the evidence are discussed in [44] where it is shown, for instance, that the mode is achieved for $\mu_0 = m$.

12.2.5 Bayesian Hypothesis Testing

In essence so far we have modeled the log-expression level of each gene in each situation using a Gaussian model. If all we care about is whether a given

gene has changed or not, we could model directly the difference between the log-expression levels in the control and treatment cases. These differences can be considered pairwise or in paired fashion, as is more likely the case with current microarray technology where the logarithm of the ratio between the expression levels in the treatment and control situations is measured along two different channels (red and green).

We can model again the differences $x^t - x^c$ with a Gaussian $\mathcal{N}(\mu, \sigma^2)$. Then the null hypothesis H, given the data, is that $\mu = 0$ (no change). To avoid assigning a probability of 0 to the null hypothesis, a Bayesian approach here must begin by giving a non-zero prior probability for $\mu = 0$, which may appear a little contrived. In any case, following the lines of the previous derivation for the conjugate prior, we can set $\mathbf{P}(\sigma^2) = \mathcal{I}(\sigma^2; \nu_0, \sigma_0^2)$. For the mean μ, we use the mixture

$$\mu = \begin{cases} 0 & : \text{ with probability } p \\ \mathcal{N}(0, \sigma^2/\lambda) & : \text{ with probability } 1-p \end{cases} \tag{12.20}$$

The parameter p could be fixed from previous experiments, or treated as an hyperparameter with, for instance, a Dirichlet prior. We leave as an exercise for the reader to compute the relevant statistics $\log[\mathbf{P}(\tilde{H})/\mathbf{P}(H)]$.

12.2.6 Implementation

For efficiency, an intermediate solution has been implemented in a Web server called Cyber-T[1] [44, 366]. In this approach, we use the t-test with the regularized standard deviation of (12.13) and the number of degrees of freedom associated with the corresponding augmented populations of points, which incidentally can be fractional. In Cyber-T, plain and Bayesian versions of the t-test can be performed on both the raw data and the log-transformed data.

In the simplest case, where we use $\mu_0 = m$, one must select the values of the background standard deviation σ_0^2, and its strength ν_0. The parameter ν_0 represents the degree of confidence in the background variance σ_0^2 versus the empirical variance. The value of ν_0 can be set by the user. The smaller n, the larger ν_0 ought to be. A simple rule of thumb is to assume that $l > 2$ points are needed to estimate the standard deviation properly and keep $n + \nu_0 = l$. This allows a flexible treatment of situations in which the number n of available data points varies from gene to gene. A reasonable default is to use $l = 10$. A special case can be made for genes with activity levels close to the minimal detection level of the technology being used. The measurements for these genes being particularly unreliable, it may be wise to use a stronger prior for them with a higher value of ν_0.

[1] Accessible at: http://128.200.5.223/CyberT/.

For σ_0, one could use the standard deviation of the entire set of observations or, depending on the situation, of particular categories of genes. In a flexible implementation, the background standard deviation is estimated by pooling together all the neighboring genes contained in a window of size ws. Cyber-T automatically ranks the expression levels of all the genes and lets the user choose this window size. The default is $ws = 101$, corresponding to 50 genes immediately above and below the gene under consideration. Adaptive window sizes and regression estimates for σ_0^2 can also be considered.

12.2.7 Simulations

We have used the Bayesian approach and Cyber-T to analyze a number of published and unpublished data sets. In every high density array experiments we have analyzed, we have observed a strong scaling of the expression variance over replicated experiments with the average expression level (on both a log-transformed and raw scale). As a result, a threshold for significance based solely on fold changes is likely to be too liberal for genes expressed at low levels and too conservative for highly expressed genes. While several biologically relevant results are reported elsewhere, we have found that the Bayesian approach compares favorably to a simple fold approach or a straight t-test and partially overcomes deficiencies related to low replication in a statistically consistent way [366].

One particularly informative data set for comparing the Bayesian approach to simple t-test or fold change is the high density array experiment reported in [19] comparing wild type *Escherichia coli* cells to mutant cells for the global regulatory protein IHF (integration host factor). The main advantage of this data set is its four-fold replication for both wild type and mutant alleles. The regularizing effect of the prior based on the background standard deviation is shown for this data in Figure 12.1 and in the simulation described below. The figure clearly shows that standard deviations vary substantially over the range of expression levels, in this case roughly in a monotonic decreasing fashion, although other behaviors also have been observed. Interestingly, in these plots the variance in log-transformed expression levels is higher for genes expressed at lower levels rather than at higher ones. These plots confirm that genes expressed at low or near background levels may require a stronger value of ν_0, or alternatively could be ignored in expression analyses. The variance in the measurement of genes expressed at a low level is large enough that in many cases it will be difficult to detect significant changes in expression for this class of loci.

In analyzing the data we found that large fold changes in expression were often associated with p-values not indicative of statistical change in

the Bayesian analysis, and conversely subtle fold changes were often highly significant as judged by the Bayesian analysis. In these two situations, the conclusions drawn using the Bayesian approach appear robust relative to those drawn from fold change alone, as large non-statistically significant fold changes were often associated with large measurement errors, and statistically significant genes showing less than two fold changes were often measured very accurately. As a result of the level of experimental replication seen in [19], we were able to look at the consistency of the Bayesian estimator relative to the *t*-test. We found that in independent samples of size 2 drawn from the IHF data set (i.e., two experiments versus two controls) the set of 120 most significant genes identified using the Bayesian approach had approximately 50% of their members in common, whereas the set of 120 most significant genes identified using the *t*-test had only approximately 25% of their members in common. This suggests that for two fold replication the Bayesian approach is approximately twice as consistent as a simple *t*-test at identifying genes as up- or down-regulated, although with only two fold replication there is a great deal of uncertainty associated with high density array experiments.

To further assess the Bayesian approach, an artificial data set can be generated assuming Gaussian distribution of log expressions, with means and variances in ranges similar to those encountered in the data set of [19], with 1000 replicates for each parameter combination. Selected means for the log data and associated standard deviations (in brackets) are as follows: −6 (0.1), −8 (0.2), −10 (0.4), −11 (0.7), −12 (1.0). On this artificially generated data, we can compare the behavior of a simple ratio (2-fold and 5-fold) approach, with a simple *t*-test, with the Bayesian *t*-test using the default settings of Cyber-T. The main results, reported in Table 12.1, can be summarized as follows:

- By 5 replications (5 control and 5 treatment) the Bayesian approach and *t*-test give similar results.
- When the number of replicates is "low" (2 or 3), the Bayesian approach performs better than the *t*-test.
- The false positive rate for the Bayesian and *t*-test approach are as expected (0.05 and 0.01 respectively) except for the Bayesian with very small replication (i.e., 2) where it appears elevated.
- The false positive rate on the ratios is a function of expression level and is much higher at lower expression levels. At low expression levels the false positive rate on the ratios is unacceptably high.
- For a given level of replication the Bayesian approach at $p < 0.01$ detects more differences than a 2-fold change except for the case of low expression levels (where the false positive rate from ratios is elevated).

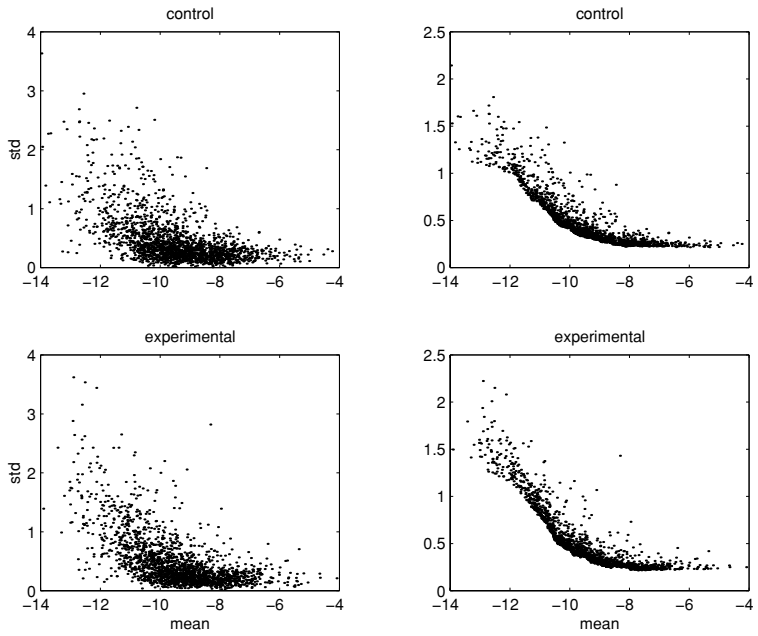


Figure 12.1: DNA Microarray Experiment on *Escherichia coli*. Data obtained from reverse transcribed P^{33} labeled RNA hybridized to commercially available nylon arrays (Sigma Genosys) containing each of the 4,290 predicted *E. coli* genes. The sample included a wild-type strain (control) and an otherwise isogenic strain lacking the gene for the global regulatory gene, integration host factor (IHF) (treatment). $n = 4$ for both control and experimental situations. The horizontal axis represents the mean μ of the logarithm of the expression levels, and the vertical axis shows the corresponding standard deviations (std=σ). The left column corresponds to raw data, the right column to regularized standard deviations using Equation (12.13). Window size is $ws = 101$ and $l = 10$ (see main text). Data are from [19].

- The Bayesian approach with 2 replicates outperforms the t-test with 3 replicates (or 2 versus 4 replicates).

- The Bayesian approach has a similar level of performance when comparing 3 treatments to 3 controls, or 2 treatments to 4 controls. This suggests an experimental strategy where the controls are highly replicated and a number of treatments less highly replicated.

	Log expression		Ratio		Plain t-test		Bayes	
n	from	to	2-fold	5-fold	$p < 0.05$	$p < 0.01$	$p < 0.05$	$p < 0.01$
2	−8	−8	1	0	38	7	73	9
2	−10	−10	13	0	39	11	60	11
2	−12	−12	509	108	65	10	74	16
2	−6	−6.1	0	0	91	20	185	45
2	−8	−8.5	167	0	276	71	730	419
2	−10	−11	680	129	202	47	441	195
3	−8	−8	0	0	42	9	39	4
3	−10	−10	36	0	51	11	39	6
3	−12	−12	406	88	44	5	45	4
3	−6	−6.1	0	0	172	36	224	60
3	−8	−8.5	127	0	640	248	831	587
3	−10	−11	674	62	296	139	550	261
5	−8	−8	0	0	53	13	39	8
5	−10	−10	9	0	35	6	31	3
5	−12	−12	354	36	65	11	54	4
5	−6	−6.1	0	0	300	102	321	109
5	−8	−8.5	70	0	936	708	966	866
5	−10	−11	695	24	688	357	752	441
2v4	−8	−8	0	0	35	4	39	6
2v4	−10	−10	38	0	36	9	40	3
2v4	−12	−12	446	85	46	17	43	5
2v4	−6	−6.1	0	0	126	32	213	56
2v4	−8	−8.5	123	0	475	184	788	509
2v4	−10	−11	635	53	233	60	339	74

Table 12.1: Number of Positives Detected out of 1000 Genes. Data was generated using normal distribution on a log scale in the range of Arfin et al. 2000 [19], with 1000 replicates for each parameter combination. Means of the log data and associated standard deviations (in brackets) are as follows: −6 (0.1), −8 (0.2), −10 (0.4), −11 (0.7), −12 (1.0). For each value of n, the first three experiments correspond to the case of no change and therefore yield false positive rates. Analysis was carried out using Cyber-T with default settings ($ws = 101$, $l = 10$) and degrees of freedom $n + \nu_0 - 2$.

12.2.8 More Complex Probabilistic Models

We have developed a probabilistic framework for array data analysis to address a number of current approach shortcomings related to small sample bias and the fact that fold differences have different significance at different expression levels. The framework is a form of hierarchical Bayesian modeling with Gausssian gene-independent models. Although the Gaussian representation requires further testing, other distributions can easily be incorporated in a similar framework. While there can be no perfect substitute for experimental replication (see also [355]), in simulations and controlled replicated experiments [366] it has been shown that the approach has a regularizing effect on the data, that it compares favorably to a conventional t-test, or simple fold-approach and that it can partially compensate for the absence of replication.

Depending on goals and implementation constraints, the method can be extended in a number of directions. For instance, regression functions could

be computed off-line to establish the relationship between standard deviation and expression levels and used to produce background standard deviations. Another possibility is to use adaptive window sizes to compute the local background variance, where the size of the window could depend, for instance, on the derivative of the regression function. In an expression range in which the standard deviation is relatively flat (i.e. between -8 and -4 in figure 12.1), the size of the window is less relevant than in a region where the standard deviation varies rapidly (i.e., between -12 and -10 in figure 12.1). A more complete Bayesian approach could also be implemented, for instance integrating the marginal posterior distributions, which in the case considered here are Student distributions, to estimate the probability $\mathbf{P}(\mu_c \approx \mu_t | D, \alpha_t, \alpha_c)$.

The approach also can be extended to more complex designs and/or designs involving gradients of an experimental variable and/or time series designs. Examples would include a design in which cells are grown in the presence of different stressors (urea, ammonia, oxygen peroxide), or when the molarity of a single stressor is varied (0, 5, 10 mM). Generalized linear and nonlinear models can be used in this context. The most challenging problem, however, is to extend the probabilistic framework towards the second level of analysis, taking into account possible interactions and correlations amongst genes. If two or more genes have similar behavior in a given treatment situation, decisions regarding their expression changes can be made more robustly at the level of the corresponding cluster. Multivariate normal models and Gaussian processes (appendix E) could provide the starting probabilistic models for this level of analysis.

With a multivariate normal model, for instance, μ is a vector of means and Σ is a symmetric positive definite covariance matrix with determinant $|\Sigma|$. The likelihood has the form

$$C|\Sigma|^{-n/2} \exp[-\frac{1}{2} \sum_{i=1}^{n} (X_i - \mu)^t \Sigma^{-1} (X_i - \mu)]. \tag{12.21}$$

The conjugate prior, generalizing the normal-scaled-inverse-gamma distribution, is based on the inverse Wishart distribution (appendix A), which generalizes the scaled inverse gamma distribution and provides a prior on Σ. In analogy with the one-dimensional case, the conjugate prior is parameterized by $(\mu_0, \Lambda_0/\lambda_0, \nu_0, \Lambda_0)$. Σ has an inverse Wishart distribution with parameters ν_0 and Λ_0^{-1}. Conditioned on Σ, μ has a multivariate normal prior $\mathcal{N}(\mu; \mu_0, \Sigma/\lambda_0)$. The posterior then has the same form, a product of a multivariate normal with an inverse Wishart, parameterized by $(\mu_n, \Lambda_n/\lambda_n, \nu_n, \Lambda_n)$. The parameters satisfy

$$\mu_n = \frac{\lambda_0}{\lambda_0 + n} \mu_0 + \frac{n}{\lambda_0 + n} m$$

$$\begin{aligned}\lambda_n &= \lambda_0 + n \\ \nu_n &= \nu_0 + n \\ \Lambda_n &= \Lambda_0 + \sum_1^n (X_i - m)(X_i - m)^t \\ &\quad + \frac{\lambda_0 n}{\lambda_0 + n}(m - \mu_0)(m - \mu_0)^t. \end{aligned} \tag{12.22}$$

Estimates similar to (12.13) can be derived for the multidimensional case.

While multivariate normal and other related models may provide a good starting point, good probabilistic models for higher-order effects influencing array data are still at an early stage of development. Most approaches so far have concentrated on more or less *ad hoc* applications of clustering methods.

12.3 Clustering

12.3.1 Overview

At the next level of complexity, we want to remove the simplistic assumption that, for instance, all the genes are independent. This is where we want to begin to look at the covariance matrix of the genes, whether there exist particular clusters of related genes, and so forth. Besides array data, clustering can be applied to many other problems in bioinformatics, including several sequence-analysis problems. Therefore here we also try to provide a brief but broad perspective on clustering that extends somewhat beyond the analysis of array data.

Clustering is a fundamental technique in exploratory data analysis and pattern discovery, aimed at extracting underlying cluster structures. Clustering, however, is a "fuzzy" notion without a single precise definition. Dozens of clustering algorithms exist in the literature and a number of *ad hoc* clustering procedures, ranging from hierarchical clustering to k-means, have been applied to DNA microarray data [160, 7, 253, 511, 484, 124, 194], without any clear emerging consensus. Because of the variety and "open" nature of clustering problems, it is unlikely that a systematic exhaustive treatment of clustering can be given. There are a number of important issues to consider in clustering and clustering algorithms, especially in the context of gene expression.

Data Types

At the highest level, clustering algorithms can be distinguished depending on the nature of the data being clustered. The standard case is when the data

points are vectors in Euclidean space. But this is by no means the only possibility. In addition to vectorial data, or numerical data expressed in absolute coordinates, there is the case of relational data, where data is represented in relative coordinates by giving the pairwise distance between any two points. In many cases the data is expressed in terms of a pairwise similarity (or dissimilarity) measure that often does not satisfy the three axioms of a distance (positivity, symmetry, and triangle inequality). There exist situations where data configurations are expressed in terms of ternary or higher-order relationships or where only a subset of all the possible pairwise similarities is given. More importantly, there are cases where the data is not vectorial or relational in nature, but essentially qualitative, as in the case of answers to a multiple-choice questionnaire. This is sometimes also called nominal data. While at the present time gene expression array data is predominantly numerical, this is bound to change in the future. Indeed, the dimension "orthogonal to the genes" covering different experiments, different patients, different tissues, different times, and so forth is at least in part non-numerical. As databases of array data grow, in many cases the data will be mixed with both vectorial and nominal components.

Supervised/Unsupervised

One important distinction amongst clustering algorithms is supervised versus unsupervised. In supervised clustering, clustering is based on a set of given reference vectors or classes. In unsupervised clustering, no predefined set of vectors or classes is used. Hybrid methods are also possible in which an unsupervised approach is followed by a supervised one. At the current early stage of gene expression array experiments, unsupervised methods such as k-means and self-organizing maps [511] are most commonly used. However supervised methods have also been tried [194], in which clusters are predetermined using functional information or unsupervised clustering methods, and then new genes are classified in the various clusters using a classifier, such as a neural network or a support vector machines (appendix E), that can learn the decision boundaries between the data classes.

Similarity

The starting point of several clustering algorithms, including several forms of hierarchical clustering, is a matrix of pairwise similarities between the objects to be clustered. The precise definition of similarity is crucial and can of course greatly impact the output of the clustering algorithm. In sequence analysis, for instance, similarity can be defined using a score matrix for gaps and substitutions and an alignment algorithm. In gene expression analysis, different

measures of similarity can be used. Two obvious examples are Euclidean distance (or more generally L^p distances) and correlation between the vectors of expression levels. The Pearson correlation coefficient is just the dot product of two normalized vectors, or the cosine of their angle. It can be measured on each pair of genes across, for instance, different experiments or different time steps. Each measure of similarity comes with its own advantages and drawbacks depending on the situation, and may be more or less suitable to a given analysis. The correlation, for instance, captures similarity in shape but places no emphasis on the magnitude of the two series of measurements and is quite sensitive to outliers. Consider, for instance, measuring the activity of two unrelated genes that are fluctuating close to the background level. Such genes are very similar in Euclidean distance (distance close to 0), but dissimilar in terms of correlation (correlation close to 0). Likewise, consider the two vectors 1000000000 and 0000000001. In a sense they are similar since they are almost always identical and equal to 0. On the other hand, their correlation is close to 0 because of the two "outliers" in the first and last position.

The Number of Clusters

The choice of the number K of clusters is a particularly thorny issue that depends, among other things, on the scale at which one looks at the data. While there have been attempts to develop methods for the automatic determination of the number of clusters [484], it is safe to say that an educated semi-manual trial-and-error approach still remains one of the most efficient techniques, and this is particularly true at the present stage for array data.

Cost Function and Probabilistic Interpretation

Any rigorous discussion of clustering on a given data set presupposes a principled way of comparing different ways of clustering the same data, hence the need for some kind of global cost/error function that can easily be computed. The goal of clustering then is to try to minimize such a function. This is also called parametric clustering in the literature, as opposed to nonparametric clustering, where only local functions are available [72].

In general, at least for numerical data, this function will depend on quantities such as the centers of the clusters, the distance from each point in a cluster to the corresponding center, the average degree of similarity of the points in a given cluster, and so forth. Such a function is often discontinuous with respect to the underlying clustering of the data. Here again there are no universally accepted functions and the cost function must be tailored to the problem, since different cost functions can lead to different answers.

Because of the advantages of probabilistic methods and modeling, it is tempting to associate the clustering cost function with the negative log-likelihood of an underlying probabilistic model. While this is formally always possible, it is of most interest when the structure of the underlying probabilistic model and the associated independence assumptions are clear. This is when the additive terms of the cost function reflect the factorial structure of the underlying probabilities and variables. As we shall see, this is the case with mixture models, where the k-means clustering algorithm can be viewed as a form of EM.

In the rest of this section, we describe in more detail two basic clustering algorithms that can be applied to DNA microarray data, hierarchical clustering, and k-means. Many other related approaches, including vector quantization [104, 484], principal component analysis, factorial analysis, self-organizing maps, NNs, and SVMs, can be found in the references.

12.3.2 Hierarchical Clustering

Clusters can result from a hierarchical branching process. Thus there exist methods for automatically building a tree from data given in the form of pairwise similarities. In the case of gene expression, this is the approach used in [160]. The output of such a method is a tree and not a set of clusters. In particular, it is usually not obvious how to define clusters from the tree since clusters are derived by cutting the branches of the tree at more or less arbitrary points.

The standard algorithm used in [160] recursively computes a dendrogram that assembles all the elements into a tree, starting from the correlation (pr distance or similarity) matrix C. At each step of the algorithm,

- The two most similar elements of the current matrix (highest correlation) are computed and a node joining these two elements is created.
- An expression profile (or vector) is created for the node by averaging the two expression profiles (or vectors) associated with the two points (missing data can be ignored and the average can be weighted by the number of elements in the vectors). Alternatively, a weighted average of the distances is used to estimate the new distance between centers without actually computing the profile.
- A new, smaller correlation matrix is computed using the newly computed expression profile or vector and replacing the two joined elements with the new node.
- With N starting points, the process is repeated at most $N-1$ times, until a single node remains.

This algorithm is familiar to biologists and has been used in sequence analysis, phylogenetic trees, and average-linkage cluster analysis. As already pointed out, after the construction of such a dendogram there is still a problem in how to display the result and which clusters to choose. At each node, either of the two elements joined by the node can be ordered to the left or the right of the other. Since there are $N-1$ joining steps, the number of linear orderings consistent with the structure of the tree is 2^{N-1}. An optimal linear ordering maximizing the combined similarity of all neighboring pairs in the ordering cannot general be computed efficiently. A heuristic approximation is used in [160] by weighting genes using average expression level, chromosome position, and time of maximal induction. The main clusters obtained on a set of gene expression data are shown indeed to have biological relevance.

12.3.3 K-Means, Mixture Models, and EM

K-Means

Of all clustering algorithms, k-means [153] has probably the cleanest probabilistic interpretation as a form of EM (expectation maximization) on the underlying mixture model. In a typical implementation of the k-means algorithm, the number of clusters is fixed to some value K. K representative points or centers are initially chosen for each cluster more or less arbitrarily. These are also called centroids or prototypes. Then at each step,

- Each point in the data is assigned to the cluster associated with the closest representative.
- After the assignment, new representative points are computed, for instance by averaging or taking the center of gravity of each computed cluster.
- The two procedures above are repeated until the system converges or fluctuations remain small.

Hence notice that k-means requires choosing the number of clusters and also being able to compute a distance or similarity between points and compute a representative for each cluster given its members.

When the cost function corresponds to an underlying probabilistic mixture model [172, 522], k-means is an online approximation to the classical EM algorithm, and as such in general is bound to converge towards a solution that is at least a local ML or MAP solution. A classical case is when Euclidean distances are used in conjunction with a mixture of Gaussian models. A related application to a sequence clustering algorithm is described in [28].

Mixtures Models and EM

To see this in more detail, imagine a data set $D = (d_1, \ldots, d_N)$ and an underlying mixture model with K components of the form

$$\mathbf{P}(d) = \sum_{k=1}^{K} \mathbf{P}(M_k)\mathbf{P}(d|M_k) = \sum_{k=1}^{K} \lambda_k \mathbf{P}(d|M_k), \tag{12.23}$$

where $\lambda_k \geq 0$ and $\sum_k \lambda_k = 1$ and M_k is the model for cluster k. The Lagrangian associated with the log-likelihood and the normalization constraints on the mixing coefficients is given by

$$\mathcal{L} = \sum_{i=1}^{N} \log\Big(\sum_{k=1}^{K} \lambda_k \mathbf{P}(d_i|M_k)\Big) - \mu\Big(\sum_{k=1}^{K} \lambda_k - 1\Big) \tag{12.24}$$

with the corresponding critical equation

$$\frac{\partial \mathcal{L}}{\partial \lambda_k} = \sum_{i=1}^{N} \frac{\mathbf{P}(d_i|M_k)}{\mathbf{P}(d_i)} - \mu = 0. \tag{12.25}$$

Multiplying each critical equation by λ_k and summing over k immediately yields the value of the Lagrange multiplier $\mu = N$. Multiplying again the critical equation across by $\mathbf{P}(M_k) = \lambda_k$, and using Bayes's theorem in the form

$$\mathbf{P}(M_k|d_i) = \mathbf{P}(d_i|M_k)\mathbf{P}(M_k)/\mathbf{P}(d_i) \tag{12.26}$$

yields

$$\lambda_k^* = \frac{1}{N} \sum_{i=1}^{N} \mathbf{P}(M_k|d_i). \tag{12.27}$$

Thus the ML estimate of the mixing coefficients for class k is the sample mean of the conditional probabilities that d_i comes from model k. Consider now that each model M_k has its own vector of parameters (w_{kj}). Differentiating the Lagrangian with respect to w_{kj} gives

$$\frac{\partial \mathcal{L}}{\partial w_{kj}} = \sum_{i=1}^{N} \frac{\lambda_k}{\mathbf{P}(d_i)} \frac{\partial \mathbf{P}(d_i|M_k)}{\partial w_{kj}}. \tag{12.28}$$

Substituting (12.26) in (12.28) finally provides the critical equation

$$\sum_{i=1}^{N} \mathbf{P}(M_k|d_i) \frac{\partial \log \mathbf{P}(d_i|M_k)}{\partial w_{kj}} = 0 \tag{12.29}$$

for each k and j. The ML equations for estimating the parameters are weighted averages of the ML equations $\partial \log \mathbf{P}(d_i|M_k))/\partial w_{kj} = 0$ arising from each point separately. As in (12.27), the weights are the probabilities of membership of the d_i in each class.

As was precisely the case for HMMs, the ML equations (12.27) and (12.29) can be used iteratively to search for ML estimates, yielding also another instance of the EM algorithm. In the E step, the membership probabilities (hidden variables) of each data point are estimated for each mixture component. The M step is equivalent to K separate estimation problems with each data point contributing to the log-likelihood associated with each of the K components with a weight given by the estimated membership probabilities. Different flavors of the same algorithm are possible depending on whether the membership probabilities $\mathbf{P}(M|d)$ are estimated in hard or soft fashion during the E step. The description of k-means given above correspond to the hard version where these membership probabilities are either 0 or 1, each point being assigned to only one cluster. This is analogous to the use of the Viterbi version of the EM algorithm for HMMs, where only the optimal path associated with a sequence is used, rather than the family of all possible paths. Different variations are also possible during the M step of the algorithms depending, for instance, on whether the parameters w_{kj} are estimated by gradient descent or by solving (12.29) exactly. It is well known that the center of gravity of a set of points minimizes its average quadratic distance to any fixed point. Therefore in the case of a mixture of spherical Gaussians, the M step of the k-means algorithm described above maximizes the corresponding quadratic log-likelihood and provides an ML estimate for the center of each Gaussian component.

It is also possible to introduce priors on the parameters of each cluster in the form

$$\mathbf{P}(d) = \sum_{k=1}^{K} \mathbf{P}(d|M_k, w_k)\mathbf{P}(w_k|M_k)\mathbf{P}(M_k) \tag{12.30}$$

and/or on the mixture coefficients. This leads to more complex hierarchical probabilistic models that may prove useful for DNA array data, or even sequence data. In sequence data, for instance, this could amount to having sequences produced by different dice, the dice coming from different factories, the factories coming from different countries, and so forth, with probabilistic distributions at each level of the hierarchy and on the corresponding properties. To the best of our knowledge, these hierarchical mixture models have not yet been explored systematically in this context.

12.4 Gene Regulation

Finally, at the third level of analysis DNA microarray expression data naturally leads to many questions of gene regulation. Understanding gene regulation at the system level is one of the most interesting and challenging problems in biology, but one where most of the principles remain to be discovered. Here, we mention only some of the main directions of research and provide a few pointers to the literature.

One direction of analysis consists in mining regulatory regions, searching, for instance, for transcription factor DNA binding sites and other regulatory motifs. To some extent, such searches can be done on a genomic scale using purely computational tools [530, 531, 232]. The basic idea is to compute the number of occurrences of each N-mer, typically for values of N in the range of 3 to 10, within an entire genome or within a particular subset of a genome, such as all gene-upstream regions. N-mers that are overrepresented are of particular interest and have been shown to comprise a number of known regulatory motifs. Distribution patterns of overreprsented N-mers can also be very informative [232]. In any case, overrepresentation of course must be assessed with respect to a good statistical background model, which can be a Markov model of some order derived from the actual counts. When in addition gene-expression data becomes available, further tuning of these mining procedures becomes possible by looking, for instance, at overrepresentation in upstream regions of genes that appear to be up-regulated (or down-regulated) under a given condition [89, 231, 535, 111, 270]. Probabilistic algorithms such as EM and Gibbs sampling naturally play an essential role in motif finding, due to both the structural and positional variability of motifs (see programs such as MEME and CONSENSUS). In any case, only a small subset of the motifs found nowadays by these techniques are typically found also in the TRANSFAC [560] database or in the current literature, and most must await future experimental verification.

A second, more ambitious direction is to attempt to model and infer regulatory networks on a global scale, or along more specific subcomponents [532, 190, 584] such as a pathway or a set of coregulated genes. One of the major obstacles here is that we do not yet understand all the details of transcription at the molecular level. For instance, we do not entirely understand the role that noise plays in gene regulation [383, 243]. Furthermore, there are very few examples of regulatory circuits for which detailed information is available, and they all appear to be very complex [579]. On the theoretical side, several mathematical formalisms have been applied to model genetic networks. These range from discrete models, such as Boolean networks, as in the pioneering work of Kauffman [310, 311, 312] to continuous models based on differential equations, such as continuous recurrent neural

networks [391] or power-law formalism [537, 466, 258], probabilistic graphical models, and Bayesian networks [190]. None of these formalisms appear to capture all the dimensions of gene regulation, and most of the work in this area remains to be done. Additional references in this active area of research can be found in the proceedings of the ISMB, PSB, and RECOMB conferences of the last few years. Understanding biology at the system level (for instance [88, 309, 239, 289, 576]), not only gene networks, but also protein networks, signaling networks, metabolic networks, and specific systems, such as the immune system or neuronal networks, is likely to remain at the center of the bioinformatics efforts of the next few decades.

This page intentionally left blank