Hidden Markov Models

P.Antal

antal@mit.bme.hu

BUTE-AIT

Overview

1. Motivations
2. HMMs
3. Inference
4. pFam
5. Learning
6. Genscan

HMM: definition

Markov chain models for sequence x (modeling sequence by states $s \in S$):

$$p(x) = \prod_{i=1}^{L} p(x_i|x_{i-1}) = p(x_1) \prod_{i=2}^{L} a_{x_{i-1},x_i} \tag{1}$$

Compare Bayes factors of different Markov chain models of DNA: homogeneous M_h vs inhomogenous-by-period-3 M_{i_3}

$$\frac{p(x|M_{i_3})}{p(x|M_h)} \tag{2}$$

Now fuse them into a single model $\Rightarrow$ hidden state
Hidden Markov Models (definitions/notations following DEKM)

1. π denotes a state sequence (of a Markov chain), π_i is the ith state
2. a_{kl} the transition probabilities $p(\pi_i = l|\pi_{i-1} = k)$ in the MC (extra state 0 for start/end)
3. $e_k(b)$ are the emission probabilities $p(x_i = b|\pi_i = k)$

Note, stochastic finite state automations/regular grammars, later we discuss the application of stochastic context-free grammars (SCFG) for RNA (3-D structure,..palindromes!).

Inferences in HMMs

Note $|\pi| = \mathcal{O}(|S|^L)$

-,L	$p(x, \pi) = a_{0\pi_l} \prod_{i=1}^{L} e_{\pi_i}(x_i) a_{\pi_i \pi_{i+1}}$
?,L	"decoding": $\pi^* = \arg\max_\pi p(x, \pi)$
?,L	sequence probability: $p(x) = \sum_\pi p(x, \pi)$ (or $p(x\|M)$ "model likelihood" or filtering)
?,L	smoothing/posterior decoding: $p(\pi_i = k\|x)$
?,OK?	parametric inference (training/parameteresation)
?,OK?	structural inference (model selection)

HMM: Viterbi algorithm

Goal: "decoding": $\pi^* = \arg\max_\pi p(x, \pi)$

Note: "best joint-state-sequence explanation"$\neq$ "joint sequence of best-state-explanations"

Inductive idea: extend most probable paths with length i to i+1

$v_k(i)$ denotes the probability of the most probable path ending in state k with observation i

Then

$$v_l(i+1) = e_l(x_{i+1}) \max_k (v_k(i) a_{kl}) \tag{3}$$

Require: HMM,x

Ensure: $\pi^* = \arg\max_\pi p(x, \pi)$

Ini: (i=0): $v_0(0) = 1$,$v_k(0) = 0$ for 0<k

for $i = 1$ to L **do**

$v_l(i) = e_l(x_i) \max_k (v_k(i-1) a_{kl})$

$ptr_i(l) = \arg\max_k (v_k(i-1) a_{kl})$

End: $p(x, \pi^*) = \max_k (v_k(L) a_{k0})$, $\pi^*_L = \arg\max_k (v_k(L) a_{k0})$

for $i = L$ to 1 **do** {Traceback}

$\pi^*_{i-1} = ptr_i(\pi^*_i)$

Note, small probabilities may cause positive underflow (length can be up to 10^3 <)=> log

Note, $\pi^* = \arg\max_\pi p(x, \pi) = \arg\max_\pi p(\pi|x)$

HMM: forward algorithm

Goal: sequence probability:$p(x) = \sum_\pi p(x, \pi)$ (or $p(x|M)$ "model likelihood" or filtering)
Approximation: $p(x) = \sum_\pi p(x, \pi) \approx p(x, \pi^*) = a_{0\pi_l^*} \prod_{i=1}^{L} e_{\pi_i^*}(x_i) a_{\pi_i^* \pi_{i+1}^*}$ (π^* by Viterbi)
Inductive idea(dynamic programming): extend the probability of generating observations $x_{1:i}$ being in state k at i to i+1
By introducing $f_k(i) = p(x_{1:i}, \pi_i = k)$, we can proceed

$$f_l(i+1) = e_l(x_{i+1}) \sum_k (f_k(i) a_{kl}) \tag{4}$$

Require: HMM M,x
Ensure: $p(x|M)$
Ini: (i=0): $f_0(0) = 1$,$f_k(0) = 0$ for 0<k
for $i = 1$ to L **do**
$f_l(i) = e_l(x_i) \sum_k (f_k(i-1) a_{kl})$
End: $p(x|M) = \sum_k (f_k(L) a_{k0})$

Note, we have to sum small probabilities! => log transformation is not enough, scaling methods..

HMM: backward algorithm

Goal: smoothing/posterior decoding $p(\pi_i = k|x)$

Idea: $p(\pi_i = k|x) = \frac{p(\pi_i=k,x)}{p(x)}$ ($p(x)$ can be computed by the forward algorithm)

$$p(\pi_i = k, x) = p(\pi_i = k, x_{1:i})p(x_{i+1:L}|\pi_i = k, x_{1:i}) = f_k(i) \underbrace{p(x_{i+1:L}|\pi_i = k)}_{b_k(i)}$$

Ensure: $b_k(i) = p(x_{i+1:L}|\pi_i = k)$

Ini: (i=L): $b_k(L) = a_{k0}$ for all k

for $i = L - 1$ to 1 **do**

$b_k(i) = \sum_l a_{kl} e_l(x_{i+1}) b_l(i + 1)$

End: $p(x|M) = \sum_l a_{0l} e_l(x_1) b_l(1)$

Note, conditionally most probable state at i $\neq$ state in most probable explanation at i.

The ”profile" HMMs (pHMMs)

Define a structure (allowed transitions) over states with cardinality n. Note, $\mathcal{O}(n^2)$ parameters can be reduced to linear. . .)

Substitutions: match states (boxes). Note, level 1 implements already a position specific scoring.

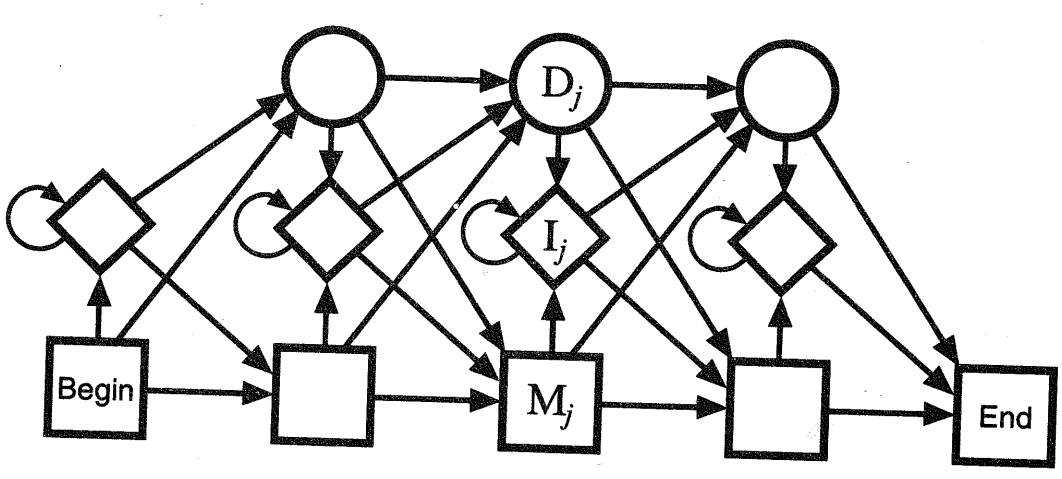
Inserts: insert states (diamonds). Note that length distribution of inserts follows a geometric distribution with parameter p of probability of stay (mean $p/(1-p)$ and variance $p/(1-p)^2$).

Deletes: transitions ”jumping" over match states. Problem: high number of parameters. Solution: further parametric restriction over transition probabilities using silent delete states (circles). Note the possible reduction of $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ representing a position specific gap length penalty or even to 1 representing a gap length penalty.

Note that delete states are so called silent/null states without emission. If there are no loops as in pHMMs =>emulate their effect in Viterbi/forward/backward algs treating separately the probability of transitions without emissions, e.g. accumulating upward $f_l(i+1) += \sum_k f_k(i+1)a_{kl}$ through silent states $k < l$.

The ”profile" HMMs (pHMMs) II.

The profile HMM.



Usage: 1, exploration/visualization of a sequence family 2, deciding membership (for transferring annotations about functionality/structure) 3, (the most probable) multiple alignment

Application: see Pfam.

The probability of alignment

Earlier we see that without indels the pairwise model $p(x_i, y_j|M)$ and independent model R $q(x_i)$ allows the use of likelihood ratio/Bayes factor...
Goal: not PSS, but with indels a full probabilistic approach to alignment.
Imagine a simple global pairwise alignment problem with linear gap penalty: given $p(x_i, y_j)$, q_i and δ probabilities for matched symbols x_i, y_j and $q_i\delta$ for a gap-ith symbol pair. Using a log transformation the dynamic programing approach to global pairwise alignment problem gives the most probable alignment, i.e. the most probable path from (0,0) to (m,n). If we change the maximization to summation it gives the total probability of alignment at (m,n).
Note the similarity to the Viterbi/forward algorithms.
Now consider the case of affine gap penalty $\gamma(g) = -d - (g-1)e$. The global pairwise alignment algorithm can be rewritten using three states and the formalism of the Viterbi algorithm as follows

$$V^M(i,j) = s(x_i, y_j) + max(V^M(i-1, j-), V^X(i-1, j-1), V^Y(i-1, j-1)) \quad (5)$$

$$V^X(i,j) = max(V^M(i-1, j) - d, V^X(i-1, j) - e) \quad (6)$$

$$V^Y(i,j) = max(V^M(i, j-1) - d, V^Y(i, j-1) - e). \quad (7)$$

(Note that gaps cannot be merged, because of the affine gap penalty $\Rightarrow$ no transition between X, Y)

HMM learning

Assume n independent/exhangeable sequences $x^{(1)}, \ldots, x^{(n)}$

$$p(x^{(1)}, \ldots, x^{(n)}|\theta) = \prod_{i=1}^{n} p(x^{(i)}|\theta) \tag{8}$$

Note, here θ corresponds to a simplified (descriptive) model class (HMMs) relying on the "molecular clock hypothesis, and not to the more general (generative/biologically inspired) model class of phylogenetic trees. Furthermore the sequences, in fact, can be (weakly?)dependent through the common evolutionary tree. . . .

1. structure known, state sequences are known: ML parameter computation from counts
2. structure known, state sequences are unknown
   (a) manual/heuristic matching: ML parameter computation from counts
   (b) : Viterbi training: iterative "multiple alignment-ML parameter computation from counts"
   (c) : Baum-Welch training: iterative computation of mean counts and improved parameters from mean counts (EM-based)
3. structure unknown, state is unknown

Estimation using known state sequences

Recall relative frequency is a maximum likelihood estimator in multinomial sampling. Assume $i = 1, \ldots K$ outcomes assuming multinomial sampling with parameters $\theta = \{\theta_i\}$ and observed occurrencies $n = \{n_i\}$ ($N = \sum_i n_i$). Then

$$\log \frac{p(n|\theta^{ML})}{p(n|\theta)} = \log \frac{\prod_i (\theta_i^{ML})^{n_i}}{\prod_i (\theta_i)^{n_i}} = \sum_i n_i \log \frac{\theta_i^{ML}}{\theta_i} = N \sum_i \theta_i^{ML} \log \frac{\theta_i^{ML}}{\theta_i} > 0 \quad (9)$$

because $0 < KL(\theta^{ML}||\theta)$

$$-KL(p||q) = \sum_i p_i \log(q_i/p_i) \leq \sum_i p_i((q_i/p_i) - 1) = 0 \quad (10)$$

using $\log(x) \leq x - 1$.
Thus using the counts of state transitions A_{kl} and emissions $E_k(b)$

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}} \text{ and } \mathrm{e_k(b)} = \frac{\mathrm{E_k(b)}}{\sum_{\mathrm{b'}} \mathrm{E_k(b')}} \quad (11)$$

So called *pseudocounts* to avoid imprecise estimates (e.g. divison by 0) and *prior counts* to incorporate bias/expertise.
$\Rightarrow A'_{kl} = A'_{kl} + r_{kl}$ $E'_k(b) = E_k(b) + r_k(b)$

Entropy and mutual information

If p_i is a discrete probability distribution, its entropy is

$$\mathrm{H}(\underline{\boldsymbol{p}}) = -\sum_i p_i \log(p_i), \tag{12}$$

Conditional entropy $H(Y|X)$ is defined as $\sum_x p(x) \sum_y p(y|x) \log(p(y|x))$.
Mutual information is defined as $I(Y;X) = H(Y) - H(Y|X)$. The (conditional) *mutual information* can be written as

$$MI_p(X;Y|Z) = \mathrm{KL}(p(X,Y|Z)|p(X|Z)p(Y|Z)). \tag{13}$$

The chain rule for (joint distributions) and entropies:
$p(X_1,\ldots,X_n) = \prod_i p(X_i|X_1,\ldots,X_{i-1})$
$H(X_1,\ldots,X_n) = \sum_i H(X_i|X_1,\ldots,X_{i-1})$
And also

$$= \quad H(X_1,\ldots,X_n) \tag{14}$$

$$= \quad \sum_{i=1}^{n} H(X_i) - \sum_{i=1}^{n} I(X_i;X_1,\ldots,X_{i-1}). \tag{15}$$

Optimality of relative frequencies

Recall relative frequency is a maximum likelihood estimator in multinomial sampling. Assume $i = 1, \ldots K$ outcomes assuming multinomial sampling with parameters $\theta = \{\theta_i\}$ and observed occurrencies $n = \{n_i\}$ ($N = \sum_i n_i$). Then

$$\log \frac{p(n|\theta^{ML})}{p(n|\theta)} = \log \frac{\prod_i (\theta_i^{ML})^{n_i}}{\prod_i (\theta_i)^{n_i}} = \sum_i n_i \log \frac{\theta_i^{ML}}{\theta_i} = N \sum_i \theta_i^{ML} \log \frac{\theta_i^{ML}}{\theta_i} > 0 \quad (16)$$

We are ready, because the last quantity is the „KL-distance", which is always positive.
If $\hat{p}_i, p_i$ are discrete probability distributions, the *cross-entropy* H and the *Kullback-Leibler* (semi)distance KL are as follows (it is always positive)
$\mathrm{H}(\underline{\boldsymbol{p}}||\underline{\hat{\boldsymbol{p}}}) = -\sum_i p_i \log(\hat{p}_i)$
$\mathrm{KL}(\underline{\boldsymbol{p}}||\underline{\hat{\boldsymbol{p}}}) = \sum_i p_i \log(p_i/\hat{p}_i)$
$0 < KL(\theta^{ML}||\theta)$:

$$-KL(p||q) = \sum_i p_i \log(q_i/p_i) \leq \sum_i p_i((q_i/p_i) - 1) = 0 \quad (17)$$

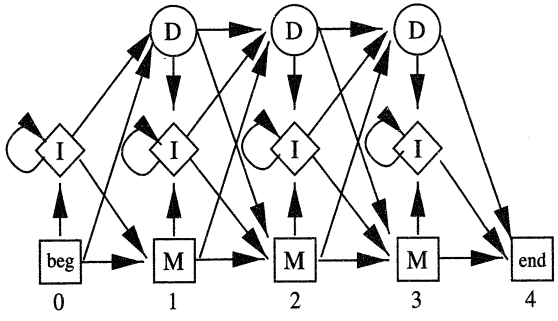
using $\log(x) \leq x - 1$.
Frequently *pseudocounts* are used to avoid imprecise estimates (e.g. divison by 0) and *prior counts* to incorporate bias/expertise.

pHMM parameter learning: heuristic

Assume an external (manual,biologically inspired) multiple alignment for sequences $x^{(1)}, \ldots, x^{(n)}$ (by evaluating the characteristics of substituted amino acids w.r.t. the secondary, tertiary structure and also considering homology, phylogenetic aspects, i.e. by adopting a system biology approach to the evolution of funtional/structural entities.
Note, that for a profile HMM (pHMM) the marking of columns with match or insert labels $M_0, I_0^+, \ldots, M_i, I_i^+$ determines the state sequence ($M_i \rightarrow \{m_i, d_i\}$,$I_i \rightarrow \{i_i\}$).
Basic profile HMM parameterisation: majority-based match/insert marking (2^L for length L)

	x	x	.	.	.	x
bat	A	G	-	-	-	C
rat	A	-	A	G	-	C
cat	A	G	-	A	A	-
gnat	-	-	A	A	A	C
goat	A	G	-	-	-	C
	1	2	.	.	.	3

(b) Profile-HMM architecture:



(c) Observed emission/transition counts

		model position 0	1	2	3
match emissions	A	-	4	0	0
	C	-	0	0	4
	G	-	0	3	0
	T	-	0	0	0
insert emissions	A	0	0	6	0
	C	0	0	0	0
	G	0	0	1	0
	T	0	0	0	0
state transitions	M-M	4	3	2	4
	M-D	1	1	0	0
	M-I	0	0	1	0
	I-M	0	0	2	0
	I-D	0	0	1	0
	I-I	0	0	4	0
	D-M	-	0	0	1
	D-D	-	1	0	0
	D-I	-	0	2	0

A MAP approach: compute the MAP probability of column i is a match.

HMM parameter learning: Viterbi

Idea: using the actual parameters compute the most probable paths $\pi^*(x^{(1)}), \ldots, \pi^*(x^{(n)})$ for the sequences and select ML parameters based on these.

Require: HMM structure, $x^{(1)}, \ldots, x^{(n)}$
Ensure: $\approx \arg\max_\theta p(x^{(1)}, \ldots, x^{(n)} | \theta, \pi^*(x^{(1)}, \theta), \ldots, \pi^*(x^{(n)}, \theta))$
  Ini: draw random model parameters θ_0 (e.g. from Dirichlet)
  repeat
    set A and E values to their pseudocount
    for $i = 1$ to n **do**
      Compute $\pi^*(x^{(i)})$ using θ_t with the Viterbi algorithm
    Set new ML parameters θ_{t+1} based on current counts A and E from $x^{(1)}, \ldots, x^{(n)}, \pi^*(x^{(1)}), \ldots, \pi^*(x^{(n)})$
    Compute model likelihood $L_{t+1} = p(x^{(1)}, \ldots, x^{(n)} | \theta_{t+1})$
  until NoImprovement(L_{t+1},L_t,t)

Note, that this finds a θ maximizing $p(x^{(1)}, \ldots, x^{(n)} | \theta, \pi^*(x^{(1)}, \theta), \ldots, \pi^*(x^{(n)}, \theta))$ and not the original goal $p(x^{(1)}, \ldots, x^{(n)} | \theta)$.

HMM parameter learning: Baum-Welch

Idea: compute the expected number of transitions/emissions A_t,E_t based on θ_t, then update to θ_{t+1} based on A_t,E_t...
The probability of $k \to l$ transition at position i in sequence x is

$$p(\pi_i = k, \pi_{i+1} = l|x) \tag{18}$$

$$= \frac{p(\overbrace{x_1, \ldots, x_i, \pi_i = k}^{f_k(i)}, x_{i+1}, \overbrace{\pi_{i+1} = l, x_{i+2}, \ldots, x_L}^{b_l(i+1)})}{p(x)} = \frac{f_k(i) a_{kl} e_l(x_{i+1}) b_l(i+1)}{p(x)} \tag{19}$$

The mean of the number of this transition and the mean of the number of emission b from state k is

$$A_{kl} = \sum_j \frac{1}{p(x^{(j)})} \sum_i f_k^{(j)}(i) a_{kl} e_l(x_{i+1}^{(j)}) b_l^{(j)}(i+1) \tag{20}$$

$$E_k(b) = \sum_j \frac{1}{p(x^{(j)})} \sum_{i|x_i^{(j)}=b} f_k^{(j)}(i) b_k^{(j)}(i), \tag{21}$$

Apply the same iterative algorithm as in Viterbi traning ($\theta_t \to A_t, E_t \to \theta_{t+1} \to \ldots$)
Why does it converge? Baum-Welch is an Expectation-Maximization algorithm

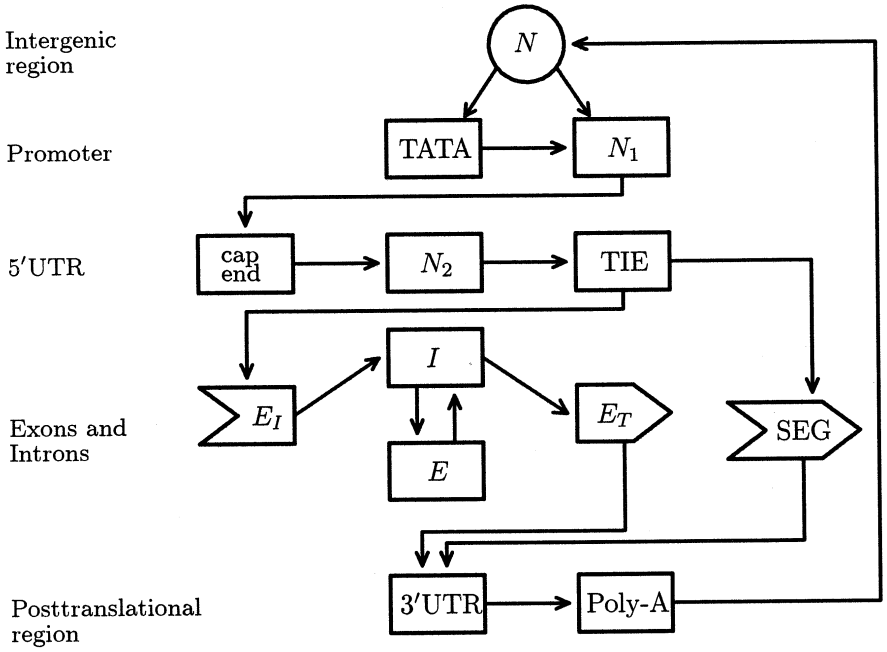
Gene finding:GENSCAN

Semihidden HMM a state can emit words with arbitrary length distribution L_S and symbols $Y_{S,l}$ (not just a symbol or words with length following a geometric distribution). A parse ϕ is a sequence of states and corresponding lengths (partition of observation is not trivial in such case!). HMM algorithms are more complex.
Application: parse of a DNA-segment with Viterbi.
Burge(1997)/EG:GENSCAN, human
Recall: what is a gene? (here we follow a protein-coding interpretation)
The training data: 380 genes, 142 single-exon genes, 1492 exons, coding region of 1619 genes

Gene finding:GENSCAN

Structural elements(of a protein coding gene): see slides.

1. Upstream region: promoter region: **TATA box**: present in 70% of genes at 28-34 bases upstream from the start of transcription.
2. **5' untranslated region** (5'UTR): follows the promoter starting with the **cap end** region (8 bases) and ending with **translation initiation end** (TIE) (18 bases).
3. $Exon - [intron - exon]^*$: recall intron types and structure
4. **3' untranslated region** (3'UTR) contains one or more **Poly-A** signal (6 bases).



Gene finding:GENSCAN II.

The **transition** probabilities are estimated from the data (to TATA, to SEG and to multi-exon).
Intergenic region: Distance between genes is modeled by a geometric distribution with mean $p/1-p = |genome|/|genes|$ and the sequence is generated with a fifth-order MC with parameters $3 \cdot 4^5$ called **intergenic null model** (INM).
TATA box is modeled with a 15-base weight matrix (independent multinomials). N_1 is from the INM with length distributed uniformly from 28 to 34. **Cap end** is modeled with an 8-base weight matrix. N_1 is from the INM with length from a geometric distribution with mean 735 bases. **TIE** is modeled with a 18-base weight matrix.
Single exon gene (SEG) is modeled with a nonhomogeneous (3-phase) fifth-order MC generating first the start codon atg and ending with the three stop codons taa, tag, tga. Length follows the empirical distribution.
Multiexon gene is modeled with the SEG model for the exons. The length of the introns are modeled with empirical distribution independently for initial, internal and terminal introns. The intron sequence generation starts with splitting a random codon with 1/3 probability to 0/3, 1/2 or 2/1. This prefix starts the intron, then the donor splice signal is modeled with a decomposed weight matrix with length 6, then the INM generates the intron, finally the acceptor splice signal is again modeled with a decomposed weight matrix with length 20, which is closed with postfix part of the splitted codon.
The **3'UTR** is modeled with the INM with geometric length of mean 450. The **Poly-A** is modeled with a 6-base weight matrix.

Extension to mRNA: SCFG

Position specific scoring of substitutions, inserts, deletions.
Hidden Markov Models
Stochastic Finite State Automaton
Stochastic grammars
Dynamic Bayesian Networks
Chomsky hierarchy of grammars (*:right/left, with/without ϵ;**:nondecreasing):

Grammar	Rule	Automaton	Parsing	Language
regular*	$W \rightarrow aW$	FSA	linear	a reg.expressio
context-free	$W \rightarrow \beta$	push-down	polynomial	palindromes
context-sensitive**	$\alpha_1 W \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$	linear bounded	exponential	copies
unrestricted		Turing machine (TM)	semidecidable	$KB - FOL$
-	-			halting TMs