Boolean Functions: Quick reference

vishvAs vAsuki

March 25, 2012

Contents

Part I

Basic properties, normal forms

1 Notation and themes

C: A function class.

1.1 Themes

The structure and power of various classes of boolean functions. The relationships between them.

2 Function Definition

2.1 Boolean function

$f : \{0,1\}^n \to \{0,1\}$ or $\{\pm 1\}^n \to \{\pm 1\}$. Ones and zeros: $I^f = \{x : f(x) = 1\}$; O^f.

2.1.1 Randomized boolean function

See algebra ref. Maps hypercube points to RV's taking value 1 with some probability.

2.2 Variables and their range

2.2.1 Changing input basis

$x_i \to x_i'; 0 \to 1; 1 \to -1 : x_i = \frac{1-x_i'}{2}$.
For fixing polynomial for change in basis: $\{\pm 1\} \to \{0,1\} : x_i \to (\frac{1+x_i}{2})$.

2.2.2 Variables and literals

Consider $x \in \{0,1\}^n$. Every x_i is a variable, and $x_i, \bar{x_i}..$ are literals.

3 Properties

3.1 Dependence on input variables

3.1.1 r-junta

f depends on only r variables. k-junta has at most 2^k non 0 coefficients. Each of them is 2^{-k+1} heavy: Make two equally high stacks unequal.
Witness to the relevance of variable x_i: an assignment a which flips f(a) when ith bit is flipped.

3.1.2 Monotone fn f

Flipping x_i from 1 to -1 cannot flip f(x) from -1 to 1. $(\forall i, x_i \leq y_i) \implies f(x_1, ..x_n) \leq f(y_1, ..y_n)$.
Define $g(x_i, x_{-i}) = f(x)$ naturally. Then $g(-1, x_{-i}) = -g(1, x_{-i})$ iff $g(-1, x_{-i}) = -1 \wedge g(1, x_{-i}) = 1$;
$E_{x_{-i}}[g(1, x_{-i}) - g(-1, x_{-i})] = 2Pr_{x_{-i}}(g(1, x_{-i})$
$= 1 \wedge g(-1, x_{-i}) = -1)$.

3.1.2.1 Influence

$\hat{f}(\{i\}) = 2^{-1}(E_{x_{-i}}[g(1, x_{-i})] - E_{x_{-i}}[g(-1, x_{-i})]) = I_i(f)$.

3.1.2.2 Total influence

As $\|x\|_1 \leq \sqrt{n}\,\|x\|_2$: $I(f) = \sum_i \hat{f}(\{i\}) \leq \sqrt{n}\sum_i \hat{f}(\{i\})^2 \leq \sqrt{n}$.
For majority: $I(f) \approx n(\frac{2}{(n-1)\pi})^{0.5}$.

3.1.3 Fairness

f is fair if it is 1 on exactly half the inputs.

3.2 Combinatorial properties

3.2.1 Sensitivity to bit i

Aka Influence of bit i. $x^{(i)}$: x with ith bit flipped. Sensitivity wrt bit i: $I_i(f) = Pr_x(f(x) \neq f(x^{(i)})$.
Total influence or avg sensitivity or Energy: $I(f) = \sum_i I_i(f)$.

3.2.2 Noise sensitivity

Aka noise stability. $y = N_\epsilon(x)$:
x with ϵ noise in each bit. $NS_\epsilon(f) = Pr_{x,y}(f(x) \neq f(y)) = 2Pr_{x,y}(f(x) = 1 \wedge f(y) = -1)$ by symmetry of sign changing bit flips.

3.3 Expressiveness measures

Let C be a class of boolean function. Viewing c as the dichotomy it induces on a given set is profitable here.

3.3.1 Count the number of f in C

One can use $|C|$ or packing/ covering numbers. See the functional analysis survey for details.

3.3.2 Dichotomy count fn

Take S with $|S| = m$. $D_C(S)$: set of dichotomies on S induced by C; Max dichotomy count: $D_C(m) = \max_{|S|=m} |D_C(S)|$.

3.3.2.1 Shattering and VCD d

If $D_C(S) = 2^m$, it is said to be shattered by C. VCD(C) is the size of maximal set which is shattered by C.

3.3.2.2 Bounding using the VCD

If $m <= d$, $D_C(m) = \sum_{i=1}^{m} \binom{n}{i} = 2^m = O(m^d)$. If $m > d$, we use Sauer's lemma. These bounds are important in quantifying the number of samples required to accurately estimate probabilities of a class of events (an example event: a classifier is wrong.). For details, see statistics and computational learning theory surveys.

Sauer's lemma (Sauer) If d = vcd(C), by induction and intermediate function,
$D_C(m) \leq \sum_{i=1}^{d} \binom{m}{i} \leq (\frac{em}{d})^d = O(m^d)$.
Pf: Let $s = |X|$. By induction on s+d. [**Find proof**]Intuition: Can't produce some dichotomy on some d+1 set.

3.3.3 Prove VCD(C) = d

Show some d-sized set shattered by C (Lower bound); show that no $d+1$ sized set is shattered (Upper bound). Use geometric intuition.

3.3.3.1 Lower bound

Try making a matrix of points shattered, whose rows are bit vectors of the shattered points.

3.3.3.2 Upper bound

Find optimal shattering geometry, optimal strategy; Use adversarial labelling. Find $D_C(m)$, solve $D_C(m) < 2^d$. If C finite, shatters S, $|C| \geq 2^{|S|}$, so $\log |C| \geq d$.

3.3.4 VCD's of some R

Intervals in R: 2. Halfspaces in R^n: n+1. Axis aligned rectangles: 4. d-gons in a plane: 2d+1. $VCD(C_G) \leq 2ds \log(es)$.
$VCD(C_{r,n}) \geq r(\log(n/r))$ if $C_{r,n}$ embedding closed, contains function f with exactly r relevant vars. Take f. Key idea: A sample point s_j is the witness of relevance of var j in f. Key idea: Make a matrix whose rows are points shattered by C. Let $k = \log(n/r)$. Build a $rk \times r2^k$ block matrix: 1 row block for every s_j. In row block j: every column block $i \neq j$ has same entry: $s_{j,i}$; but column block i has all separate entries from $0, 1^k$. To select any set of points/ rows, select embedding of f into the right set of relevant vars.
So, for decision list of r relevant vars: $r(\log(n/r)) \leq d \leq$rlog n.

4 Views and normal forms

4.1 Views

The truth table. Subset of the hypercube. The concept.
A set system: Can consider c as a set: set of 'selected' points, induces dichotomies on X.
Social (binary) choice or voting scheme of n people.

4.2 Normal forms

4.2.1 Write as sign of multinomial of n variables

$x \in \{0, 1\}^n$. Take DNF, change $\bar{x_i} \to (1 - x_i)$.

4.2.2 k-DNF and k-CNF

$k \leq n$. DNF is a disjunction of conjunctions. CNF is a conjunction of disjunctions.

4.2.2.1 Derivation from truth tables.

DNF c for boolean function f is obtainable from truth table of f:
$\{x : f(x) = 0\}$.
CNF d for boolean function f is obtainable from truth table of f: get the CNF of $\bar{f}(x)$, negate it to get a d.

4.2.2.2 Connection between DNF and CNF.

Any term in d and any clause in c cannot be disjoint [**Find proof**].
t term CNF can be reduced to t-DNF using distributive laws.
t-DNF to t-term CNF is also possible, but this may take exponential time as $RP \neq NP$: See colt ref about hardness of proper learning k-DNF.

4.2.2.3 Minimization of number of terms

Drawing rectangles in Karnaugh / Veitch tables.

Minterms and maxterms Minterms of a DNF are terms such that: Each var appears ≤ 1 times. Maxterms of a CNF are similarly define. So, a minimal DNF is a sum of minterms, while the minimal CNF is a product of maxterms. A minterm as $\{\pm 1\}^n$ fn: $\prod(\frac{1+x_i}{2})$. Constant term is coefficient of $p_\phi := 1$.

4.2.2.4 DNF norm

In the uniform distr fn space, Mansour's conjecture:
$\exists$ polynomial p: $\|DNF\|_1 \leq O(n^{\frac{1}{\epsilon}}); \|f - p\|^2 \leq \epsilon$.

4.2.3 Size s DNF f

Aka s-term DNF. Some notation: Term-length t, terms $\{T_i\}$, s = size(f) = num of terms: maybe poly(n), n variables.
Actual representation size $\leq ns \log n$.
$|k - DNF| \leq (2n)^k$: count the number of possible ways to make a k-DNF.
Number of functions: $O(\binom{3^n}{s}) = O(3^{ns})$. So, number of polysize DNF: '$3^{n^{O(\log n)}}$.
t-DNF reducible to disjunction upon feature expansion. With distributive laws, size s DNF reducible to s-CNF and thence to negation of s-DNF. Negation of size s DNF is size s CNF.
Strictly more expressive than decision lists: [**Find proof**]: so more powerful.

4.2.3.1 DNF f to l-augmented Decision tree of rank $\leq \frac{2n}{l} \log s + 1$

Let p = terms of length $> l$, make most commonly occuring literal x_i in the p terms (occurs atleast in $\frac{pl}{2n}$ by pigeon hole), make kids DNF's with x_i set to 0 (make many terms disappear) and 1; repeat till all terms have lower term length: rank r(n, p) increases when r(kids): $r(n-1, p-\frac{pl}{2n})$ and r(n-1,p) are same; r(n,0) = r(0, p) = 0; solving recurrance, rank $\leq \frac{2n}{l} \log s + 1$.

4.2.3.2 DNF f to PTF

$sign(T_1 \cdots + T_s - 2^{-1})$. So, Any f needs $\leq n$ PTF: Any f is at most n-DNF.
Has PTF of degree $O(\sqrt{t}\log s)$: Let $T_i = x_1 \ldots x_t, a_i(\bar{x}) = \frac{x_1+\ldots x_t}{t}$, Chebyshev pol (see Approx theory sheet) $C_{\sqrt{t}}((1+t^{-1})a_i(\bar{x})) = Q_i, deg(Q_i) = \sqrt{t}$; for $T_i(x) = 1, |Q_i| \geq 2$, else $|Q_i| \leq 1$; PTF is $\sum Q_i^{\log s} - s - 2^{-1}$.
Has $O(n^{1/3}\log s)$ PTF:
Make $n^{2/3}$ augmented Decision tree of rank $2n^{1/3}\log s + 1$; change each $n^{2/3}$ DNF at leaf to $n^{1/3}\log s$ PTF; change Decision tree to $2n^{1/3}\log s + 2$ Decision list (k long, terms: $\{T_i'\}$) which output PTF's $\{P_i\}$; $\exists$ PTF with polynomial $2^k T_1' P_1 + 2^{k-1} T_2' P_2 \ldots$ of degree $O(2n^{1/3}\log s + n^{1/3}\log s)$.

4.2.3.3 Lower bounds on PTF degree

f = parity fn: $\{0,1\}^n \to \pm 1$, an $O(2^n)$ size DNF needs n PTF: Let sign(Q(x)) be parity PTF: $x_i^2 = x^i$, so all polynomials (even Q(x)) multilinear; $Q(x) = Q(\pi(x))$; let $Q'(x) = \sum_\pi Q(\pi(x_1 \ldots x_i))$; Q' symmetric, sign(Q') computes parity, degree(Q') = degree(Q); $Q'(x) = Q''(\sum x_i)$; $Q''(0) < 0, Q''(1) > 0, Q''(2) < 0 \ldots$; so Q" has n roots; so Q', Q has degree n.
Some polysize DNF in n variables needs PTF of degree $n^{1/3}$: DNF f = '1 in a box' fn (Minsky, Papert): $s = m; T_i = \bigwedge_{j=1}^{4m^2} x_{i,j}$; let x_i be variables in box T_i; let sign(Q(x)) be PTF for f; $Q'(x) = \sum_{\pi_{1,i}\ldots\pi_{m,i'}} Q(\pi_{1,i}(x_1), \ldots \pi_{m,i'}(x_m))$; sign(Q') also is f, degree(Q') same; $Q''(\sum x_{1,i}, \cdots \sum x_{m,i})$ with same degree $Q''(x) > 0$ iff $\sum x_{1,i} \geq 4m^2$; let $p(t) = Q''(4m^2 - (t-1)^2, 4m^2 - (t-3)^2 \ldots 4m^2 - (t-(2m-1))^2)$; $deg(p) = 2 * deg(Q'')$; $p(1) > 0, p(2) < 0, p(3) > 0 \ldots$; $degree(p) \geq 2m$, so $deg(Q) \geq m$.

4.2.4 Boolean circuits (ckt)

A DAG with n inputs: C_n; internal nodes represent logic gates/ operations; directed edges represent inputs and outputs to these nodes. Circuit basis: the gates: usually AND, OR. Bounded vs unbounded fanin. Size and depth of ckt. Using De Morgan's laws, any ckt can be changed to have all NOT gates at input. Ckt with bounded fan in k can be changed to have bounded fan in 2 with constant blowup in depth. Unbounded fanin ckt convertible to bounded fanin ckt with $\log(size(ckt))$ blowup in depth.
Turing machine halting in time $T(n)$ convertible to ckt of depth $T(n)$ and size $(T(n))^2$.

4.2.4.1 Boolean formula f

Aka Boolean expressions, tree like circuits. Subset of Boolean circuits: Only 1 output per gate. Can be rewritten as a 3 CNF.

4.2.4.2 Balance tree-like circuits, reduce depth

(Spira): Take tree O with n nodes, find subtree A with (2/3)n nodes; A could be T or F: create 2 trees BT, BF out of the other n/3 nodes with either A=T or A=F; Now, $O = (A \wedge BT) \vee (\sim A \wedge BF)$; keep repeating this for A, BF, BT to get tree of depth $log_{\frac{3}{2}} n$.

4.2.5 Turing machines

See Computational Complexity ref.

5 Fourier analysis of boolean fn

Fourier analysis of ANY real valued boolean fn.

5.1 The function space for Uniform distribution

5.1.1 Dimensionality

Take the real space R^{2^n}: Not ∞ dimensional Hilbert space. Any real valued boolean function can be represented by a point in this space.
Can also be applied to randomized boolean functions: Alter expectations and probabilites to handle extra randomness.

5.1.2 Inner product

$\langle f, g \rangle := 2^{-n} \sum_x f(x)g(x) = E_{x \in_U \{\pm 1\}^n}[f(x)g(x)]$: defined thus scaled to make $\|p_S\| = \|f\| = 1$. If $S \neq T$: $\langle p_S, p_T \rangle = 0$.

5.1.2.1 The basis

These form an alternate basis: Parity functions p_S, where S is the index vector. The standard basis is just $\{e_i\}$.

5.2 Transforms

5.2.1 Discrete Fourier series

Discrete Fourier series for any real valued fn over $\{\pm 1\}^n$: $f = \sum_{S \subseteq [n]} \hat{f}(S) p_S$. $\|f\|^2 = \sum_S \hat{f}(S)^2 = E_x[f(x)^2] = 1$: [Parseval].

5.2.2 Correlation / Discrete Fourier coefficient

Aka Fourier weights on basis vectors or Fourier spectrum. $\hat{f}(S) = \langle f, p_S \rangle = E_x[f(x)p_S(x)] = Pr_x(f(x) = p_S(x)) - Pr_x(p(x) \neq p_S(x))$: so, a measure of how good an approx of f p_S is. $\langle f, g \rangle = \sum_S \hat{f}(S)\hat{g}(S)$.

5.2.3 Fourier transform of f

$\hat{f}(S) = 2^{-n} \sum_x f(x) p_S(x)$. All $\hat{f}(S)$ can be computed in time $O(n2^n)$ using FFT alg, given all f(x). [**Find proof**]Inverse FFT can be similarly done. Also see Functional analysis ref.

5.3 Estimating coefficients

5.3.1 Estimate all Fourier coefficients of f using FFT

f_R: A boolean fn on $\{0,1\}^m$: Take random m*n matrix, define $f_R(y) = f(yR)$. Fourier coefficients of f_R are Fourier coefficients of f restricted to a random subspace.
For random non singular m*n R, with m large enough to give good estimation of $Pr(f(x)p_a(x) = 1)$ whp. Thence find approximations of all the Fourier coefficients of f by finding all coefficients of f_R using FFT.

5.3.2 Relationship with total influence

$I(f) = \sum_S |S| \hat{f}(S)^2$. [**Find proof**]

5.3.3 Relationship with noise sensitivity

Eg: $NS_\epsilon(\bigwedge_i x_i) = \frac{2}{2^n}(1 - (1-\epsilon)^n) \approx 0$; For parity: $NS_\epsilon(p_{1^n}(x)) = 2^{-1} - 2^{-1}(1-2\epsilon)^n \approx 2^{-1}$; note correlation with good/ bad Fourier concentration.
$NS_\epsilon(f) = 2^{-1} - 2^{-1} \sum_S (1-2\epsilon)^{|S|} \hat{f}(S)^2$:
$Pr_{x,y}(f(x) \neq f(y)) = 2^{-1} - 2^{-1} E_{x,y}[f(x)f(y)]$;
But $E_{x,y}[f(x)f(y)] = E_{x,y}[(\sum_S \hat{f}(S)p_S(x))(\sum_T \hat{f}(T)p_T(x))]$
$= \sum_{S,T} \hat{f}(S)\hat{f}(T) E_{x,y}[p_S(x)p_T(y)] = \sum_S \hat{f}(S)\hat{f}(T) E_{x,y}[p_S(x)p_S(y)]$; but
$E_{x,y}[p_S(x)p_S(y)] = \prod_i E[p_{\{i\}}(x)p_{\{i\}}(y)] = (1-2\epsilon)^{|S|}$.
$\sum_{|S| \geq \epsilon^{-1}} \hat{f}(S)^2 \leq NS_\epsilon(f)$, k const: $2NS_\epsilon(f) = 1 - \sum_S (1-2\epsilon)^{|S|} \hat{f}(S)^2 = \sum_S \hat{f}(S)^2 - \sum_S (1-2\epsilon)^{|S|} \hat{f}(S)^2 = \sum_S \hat{f}(S)^2 (1 - (1-2\epsilon)^{|S|}) \approx \sum_S \hat{f}(S)^2 (1 - e^{-2}) \geq k' \sum_S \hat{f}(S)^2$.
As, $\sum_{|S| \geq \epsilon^{-1}} \hat{f}(S)^2 \leq NS_\epsilon(f)$, any f is $(NS_\epsilon(f), \epsilon^{-1})$ concentrated.

5.3.4 Concentration in the spectrum

Aka Fourier concentration.

5.3.4.1 (eps, d) concentrated function f

$\left\| \sum_{|S|>d} \hat{f}(S) p_S \right\|_2^2 \leq \epsilon$. Visualize with histogram.
If C = polysize DNF formulae, every $c \in C$ is $(\epsilon, \log|c| \log \frac{1}{\epsilon})$ concentrated. [**Find proof**]
Depth d ckts of size $|c|$: $(\epsilon, \log|c| \log \frac{1}{\epsilon})^{d-1}$ concentrated. [**Find proof**]

5.3.5 Best fitting d degree polynomial

$\|f-g\|^2 = \sum_S (\hat{f}(S) - \hat{g}(S))^2 = E_x[(f(x)-g(x)))^2]$;
so $\min_g E_x[(f(x)-g(x))^2] = \sum_{|S|\leq d} \hat{f}(S) p_S$.

5.3.6 t - sparse f

Number of non zero coefficients in f $\leq t$.

5.3.6.1 Sparse approximator for any f

$\exists \frac{\|f\|_1^2}{\epsilon}$
sparse g with $\|f-g\|^2 \leq \epsilon$: Remove all $\hat{f} < \frac{\epsilon}{\|f\|_1}$; then g is $\frac{\|f\|_1^2}{\epsilon}$ sparse: $\|f\|_1 = \sum \hat{f}(S)$ and each $\hat{f}(S)$ has min size $\frac{\epsilon}{\|f\|_1}$. $\|f-g\|^2 = \sum_{p_S \notin basis(g)} \hat{f}(S)^2 \leq (\max_{p_S \notin basis(g)}) \hat{f}(S) \sum_{p_S \notin basis(g)} \hat{f}(S) \leq \epsilon$.
So, sgn(g) approximates f.

5.3.6.2 Weak parity learning problem

Let f have a t-heavy Fourier component; then find a t/2 heavy Fourier component. Thence find weakly correlated parity.
KM alg solves this; See colt ref.

Part II

Boolean function classes

6 Decision lists and trees

6.1 Decision lists

Decision list is fully specified by a sequence of k variables (x_i) and outputs $r(x_i), r'(x_k)$. It is like an 'if .. elseif .. elseif .. else' statement. It can be visualised as a chain of variables, each with one outgoing edges representing an output.
In a d-decision list, $d-CNF$'s are used in place of $\{x_i\}$.

6.1.1 Generality

This can be writ as halfspace: $sign(\sum 2^{k-i} x_i o(x_i))$.
We can write conjunctions, disjunctions as decision lists.

6.2 Decision trees

DAG with internal nodes labelled with variables; leaves are labelled 0 or 1 (range of the 'label' variable to be predicted). A decision tree is same as a nested 'if .. else ..' statement.

6.2.1 Power

They include decision lists, but are strictly more powerful: can represent $x_1 \oplus x_2$ as a decision tree.

6.2.2 Evaluation and interpretation

Apart form the overall classification error, one can consider classification errors particular to various decision paths. The leaves of the decision trees, which represent a decision path can be labeled with the error rate observed for that particular path.

6.2.3 t augmented Decision tree

Decision tree with t-DNF's at leaves. t augmented Decision tree of rank r reducible to t + r + 1 decision list.

6.2.4 Rank of decision trees

Rank of a leaf variable is 1. Rank of decision tree (not max depth) = Max (ranks of kids) if kids have diff ranks; else rank of kid + 1. So, $rank \leq \log(nodes)$.

6.2.5 Number of decision trees

Number of Decision trees = 2^{2^n}. Number of poly size Decision trees probably same as number of polysize DNF's: they're interchangeable [**Check**].

6.2.6 Rank r decision tree to r+1 decision list

Easy decision list for Rank 1 subtree; Take node x with kid subtrees T_1 and T_2, append $\bar{x}$ to conjunctions of DL of T_1 and x to conjunctions of DL of T_2, join them.

6.2.7 Decision tree to polysize DNF f

Taking $\vee$ of AND's corresponding to paths to leaves with label 1; only one of the terms can be true at a time. So, f = $\sum$ AND terms, with $\|AND(..)\|_1 \leq 1$. So, $\|f\|_1 \leq t$.

6.2.8 Sparsity

As $\|f\|_1 \leq t$, Decision tree with t leaves has approximator with sparsity $O(t)$.

7 Other function classes

7.1 Conjunctions and disjunctions

View as a set.

7.2 Boolean functions from real valued functions

Take epigraph or subgraph. See complex analysis ref.

7.3 Halfspace

Aka Linear Threshold fn (LTF). $f = sign(\sum a_i x_i + c) = sgn(a^T x + c)$, $a_i, c \in Z$. x called the weight vector, c called the bias.
$a^T x + c = 0$ is a Hyperplane: take pts x and x' on the hyperplane, use $a^T(x - x') = 0$; so a specifies orientation. Distance from origin : $\frac{a^T x}{\|a\|} = \frac{-c}{\|a\|}$. Distance of any pt x from hyperplane: $\frac{a^T x - (-c)}{\|a\|} = \frac{f(x)}{\|a\|}$ by geometry.
Weight of halfspace W = $\sum |a_i| + |c|$. Low weight LTF has W = poly(n).

7.3.1 Intuition

+1 on surface of one half of the unit sphere; -1 elsewhere; find orientation of halfspace. Like line in R^2.

7.3.2 Noise sensitivity and Fourier concentration

$NS_a(f) \leq \sqrt{a}$. [**Find proof**]So, every halfspace is (a, a^{-2}) concentrated. By union Bound, if F is fn of k LTF, $NS_a(F) \leq k\sqrt{a}$. So, F is $(a, k^2(a)^{-2})$ concentrated.

7.3.3 Conversion to PTF

$\exists$ Rational function $R(x) = \frac{p(x)}{q(x)}$ of degree $O(l \log t)$ like sign function for $|x| \in [1, 2^l]$: ergo $\exists$ degree $O(l \log t)$ P(x)=p(x)q(x) doing the same. So, get O(log W) PTF.
So, Functions of $\{f_i\}$ W weight halfspaces: $\bigwedge^s f_i$ has O(log W) PTF: Use P(x) on $\sum^s f_i - s$.

7.3.4 Make origin centered halfspace

Add an extra dimension: Map f to $f' = sgn(\sum_{i=1}^{n+1} a_i x_i) : x_{n+1} = 1, a_{n+1} = c$.

7.4 Automata

See [1].

7.4.1 Grammar

7.4.1.1 Definition

A grammar specifies a language by describing structural rules which can be used to accept or reject a sentence. So, each grammar is associated with a language.
It is fully described by (Variables, Terminal chars/ alphabet, Start symbol, Production rules).
Production rules are of the form: string including variable V $\rightarrow$ string with variables and terminal symbols, formed by replacing V with somwething else.
A given string is accepted if, starting with the start symbol, using a finite number of production rules, one can arrive at the string.

7.4.1.2 Parsing connection

The rules involved in generating a given string (sentence or word) is closely connected to the semantic meaning underlying that string.

7.4.1.3 Characteristics

There are conflicting demands on grammar design/ development.
Due to the parsing connection, unambiguity is desirable.
A good grammar should define a language with which a wide variety of semantics can be expressed. So, expressiveness is desirable.
At the same time, if the resultant language is being used for communicaiton, brevity is desirable.

7.4.1.4 Context free grammars

(CFG) constrain production rules so that their left hand side (LHS) has exactly one variable/ start symbol.

7.4.1.5 Normal forms

Chomsky normal form for CFG: only $A \rightarrow BC$ or $A \rightarrow \alpha$. Greibach normal form for CFG: only $A \rightarrow \alpha C$.

7.4.1.6 Other traits

Pumping Lemma for CFG (How could a sufficiently long string be produced by a CFG?); use in checking if language is CFG.
Syntax trees and ambiguity.

7.4.2 Regular expression

This is a symbolic representation of a certain type of languages (defined by its syntax). This language consists of strings which may be produced by the application of production rules specified by the regular expression. THe production rules may also be viewed as producing a set of strings.

7.4.2.1 Production rules

Fundamental production rules include concatenation (cartesian product of string sets), alternation (union of k sets) represented by | , and the 'kleene star' represented by * (union with empty set of the concatenation-closure of a certain set).

7.4.2.2 Grouping

To clarify operator precedence, expressions are grouped together. Generally, groupings are specified by enclosing them within ().

7.4.2.3 Example

abc(def—hig)*uvw.
Also see programming languages' use of regular expressions for string search and for the associated richer syntax.

7.4.3 Finite state automata

This is a class of boolean functions over strings that can be formed using a finite alphabet.

7.4.3.1 States and transitions

Finite state automaton is a collection of states and transitions between the states. The states include special states such as exactly 1 start-state and ≥ 1 accept-states.
Every transition between two states is labeled with a (input) character from the alphabet. Depending on the number of transitions corresponding to a given (state, letter) pair, the transition is said to be deterministic or non-deterministic.

7.4.3.2 As a directed graph

One can use a directed graph where nodes and edges correspond to states and transitions. The edges are labelled with appropriate input characters.

7.4.3.3 Acceptance

A given string is accepted or rejected based on whether one can reach a 'final-state' by making transitions appropriate to characters in the string.

7.4.3.4 Resources

Uses constant memory.

7.4.3.5 Expressiveness

Accept regular languages (describable by a regular expression). Pumping lemma for regex: Middle part of strings in regular languages can be pumped. Strictly more powerful than decision trees. [**Find proof**]

7.4.3.6 (Non) determinism

Determinsitic (DFA) and non deterministic representations.

7.4.4 Pushdown automata

Determinsitic Push-down automata and stack memory. Non deterministic push-down automata; accepts context-free languages (use Greibach).

7.5 Polynomial Threshold functions (d-PTF)

Multivariate poly bool $f(..) = sign(p(..))$: degree $d = \sum degree(x_i)$. Reduce to halfspace by feature expansion: make n^d variables.

7.6 Neural network

G: Directed layered acyclic graph with s internal nodes of indegree r, n inputs, 1 output. C_G (in R^n): G composition of C (in R^r); each node in G assigned $c \in C$. **Neural net**: C_G with each $c \in C$: r weights w_i, threshold θ.

7.7 Important functions

7.7.1 Parity function

Important in error checking, Fourier analysis.
$p : \{-1,1\}^n \to \{-1,1\}$ better than $\{0,1\}^n$ defn: $p_S(x) := \prod_{x_i \in S} x_i$, $p_\phi := 1$.
Length of $p_S = |S|$.
Also writ as $p_S(x) : S = \{0,1\}^n$, S is indicator vector or index. Or as $p_S(x) = (-1)^{S^T x}$, where $S.x$ is the GF(2) inner product.
Also GF_2 form: $f = \sum_{i \in S} x_i \bmod 2$: can be writ as a bit vector. Assignment x are also bitvector; evaluation: $f(x) = \langle x, f \rangle$.
$p_S p_T = p_{S \Delta T}$. $p_S(x \oplus y) = p_S(x) p_S(y)$ [**Check**].

7.7.2 Majority function

$maj(x) = sgn(\sum_i x_i)$.

7.7.2.1 Noise sensitivity

$NS_\epsilon(maj) = \sqrt{\epsilon}$: Random walks on Z starting from 0 induced by noise and by x; whp total deviation due to noise is $\leq \sqrt{\epsilon n}$; after rand walk by x, prob of remaining within $\sqrt{n}$ of 0 is $\sqrt{\epsilon}$ by looking at lengths of line segments.
[**Incomplete**]

Bibliography

[1] Peter Linz. *An introduction to formal languages and automata.* D. C. Heath and Company, Lexington, MA, USA, 1990.