Perceptron Learning of SAT

Matthew Blaschko

Center for Visual Computing
École Centrale Paris

Inria Saclay - Île-de-France

December, 2012

1Joint work with Alex Flint, University of Oxford.
The Davis-Putnam-Logemann-Loveland Algorithm
Perceptron Learning
Some Theoretical Results
Feature Space
Empirical Results
The SAT problem is to determine whether a sentence $\Omega$ in propositional logic is satisfiable.

- A binary variable $q$ takes on one of two possible values, $\{0, 1\}$.
- A literal $p$ is a proposition of the form $q$ (a “positive literal”) or $\neg q$ (a “negative literal”).
- A clause $\omega_k$ is a disjunction of $n_k$ literals, $p_1 \lor p_2 \lor \cdots \lor p_{n_k}$.
- A unit clause contains exactly one literal.
- A sentence $\Omega$ in conjunctive normal form (CNF) is a conjunction of $m$ clauses, $\omega_1 \land \omega_2 \land \cdots \land \omega_m$. 
The Davis-Putnam-Logemann-Loveland Algorithm

if $\Omega$ contains only unit clauses and no contradictions then
  return YES
end if

if $\Omega$ contains an empty clause then
  return NO
end if

for all unit clauses $\omega \in \Omega$ do
  $\Omega := \text{UnitPropagate}(\Omega, \omega)$
end for

for all literals $p$ such that $\neg p \notin \Omega$ do
  remove all clauses containing $p$ from $\Omega$
end for

$p := \text{PickBranch}(\Omega)$

return $\text{DPLL}(\Omega \land p) \lor \text{DPLL}(\Omega \land \neg p)$
UnitPropagate simplifies $\Omega$ under the assumption $p$.

PickBranch applies a heuristic to choose a literal in $\Omega$.

State-of-the-art SAT systems are based on the DPLL algorithm with engineered heuristics for PickBranch.

Good heuristics have been empirically shown to reduce processing time by several orders of magnitude.
The DPLL Algorithm Continued

- UnitPropagate simplifies $\Omega$ under the assumption $p$.
- PickBranch applies a heuristic to choose a literal in $\Omega$.
- State-of-the-art SAT systems are based on the DPLL algorithm with engineered heuristics for PickBranch.
- Good heuristics have been empirically shown to reduce processing time by several orders of magnitude.
- In our work we learn heuristics by optimizing over a family of the form,

$$\arg\max_p f(x, p)$$

where $x$ is a node in the search tree, $p$ is a candidate literal, and $f$ is a priority function mapping possible branches to real numbers.
- We are unaware of any branching heuristics in the literature that cannot be expressed in this form.
Overview

- The Davis-Putnam-Logemann-Loveland Algorithm
- **Perceptron Learning**
- Some Theoretical Results
- Feature Space
- Empirical Results
Learn a binary function of the form \( \text{sign}(\langle f, x \rangle) \).

For each training sample \((x_i, y_i) \in \mathbb{R}^d \times \{-1, 1\}\)

\[
f_i = f_{i-1} + \begin{cases} 
0 & \text{if } y_i \langle f_{i-1}, x_i \rangle > 0 \\
\eta_i y_i x_i & \text{otherwise}
\end{cases}
\]

where \(\eta_i\) is a learning rate, e.g. 1.

Extremely simple learning rule, guaranteed optimality in a small number of updates if data are linearly separable.
$x_j$ - a node that is visited in the application of the DPLL algorithm.

$\varphi_i(x_j)$ - the feature map associated with instantiating literal $p_i$.

Decision function at $x_j$ takes the form

$$\arg\max_i \langle f, \varphi_i(x_j) \rangle_H.$$
DPLL Feature Map and Margin

- $x_j$ - a node that is visited in the application of the DPLL algorithm.
- $\varphi_i(x_j)$ - the feature map associated with instantiating literal $p_i$.
- Decision function at $x_j$ takes the form

$$\arg\max_i \langle f, \varphi_i(x_j) \rangle_H.$$ 

- $y_{ij}$ is $+1$ if the instantiation of $p_i$ at $x_j$ leads to the shortest possible proof, and $-1$ otherwise.
- Margin:

$$\max \gamma \text{ s.t. } \langle f, \varphi_i(x_j) \rangle_H - \langle f, \varphi_k(x_l) \rangle_H \geq \gamma$$

$$\forall \{(i, j) | y_{ij} = +1\}, \{(k, l) | y_{kl} = -1\}$$
Restrict to satisfiable sentences.
Restrict to satisfiable sentences.

\[ \nabla_{\text{DPLL}} = \sum_{(i,j) \in S_+} \frac{\varphi_i(x_j)}{|S_-|} - \sum_{(k,l) \in S_-} \frac{\varphi_k(x_l)}{|S_+|} \]

\[ f_{t+1} = f_t - \eta \nabla_{\text{DPLL}} \]
∀i, j, k, l \quad \| \varphi_i(x_j) - \varphi_k(x_l) \| \leq R

**Theorem**

For any training sequence that is separable by a margin of size \(\gamma\) with \(\|f\| = 1\), using the DPLL stochastic gradient with \(\eta = 1\), the number of errors (updates) made during training on satisfiable sentences is bounded above by \(R^2 / \gamma^2\).
Overview

- The Davis-Putnam-Logemann-Loveland Algorithm
- Perceptron Learning
- Some Theoretical Results
- Feature Space
- Empirical Results
Polynomial Time Equivalences

**Theorem**

\[ \exists \text{ a polynomial time computable } \varphi \text{ with } \gamma > 0 \iff \Omega \text{ belongs to a subset of satisfiable sentences for which there exists a polynomial time algorithm to find a valid valuation.} \]
Theorem

∃ a polynomial time computable ϕ with γ > 0 ⇐⇒ Ω belongs to a subset of satisfiable sentences for which there exists a polynomial time algorithm to find a valid valuation.

Corollary

∃ polynomial time computable feature space with γ > 0 for SAT ⇐⇒ P = NP
The Davis-Putnam-Logemann-Loveland Algorithm
Perceptron Learning
Some Theoretical Results
Feature Space
Empirical Results
Feature Space

- $C(p)$ – number of occurrences of $p$ in $\Omega$.
- $C_k(p)$ – number of occurrences of $p$ among clauses of size $k$.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Dimensions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>is-positive</td>
<td>1</td>
<td>1 if $p$ is positive, 0 otherwise</td>
</tr>
<tr>
<td>lit-unit-clauses</td>
<td>1</td>
<td>$C_1(p)$, occurrences of literal in unit clauses</td>
</tr>
<tr>
<td>var-unit-clauses</td>
<td>1</td>
<td>$C_1(q)$, occurrences of variable in unit clauses</td>
</tr>
<tr>
<td>lit-counts</td>
<td>3</td>
<td>$C_i(p)$ for $i = 2, 3, 4$, occurrences in small clauses</td>
</tr>
<tr>
<td>var-counts</td>
<td>3</td>
<td>$C_i(q)$ for $i = 2, 3, 4$, as above, by variable</td>
</tr>
<tr>
<td>bohm-max</td>
<td>3</td>
<td>$\max(C_i(p), C_i(\neg p))$, $i = 2, 3, 4$</td>
</tr>
<tr>
<td>bohm-min</td>
<td>3</td>
<td>$\max(C_i(p), C_i(\neg p))$, $i = 2, 3, 4$</td>
</tr>
<tr>
<td>lit-total</td>
<td>1</td>
<td>$C(p)$, total occurrences by literal</td>
</tr>
<tr>
<td>neg-lit-total</td>
<td>1</td>
<td>$C(\neg p)$, total occurrences of negated literal</td>
</tr>
<tr>
<td>var-total</td>
<td>1</td>
<td>$C(q)$, total occurrences by variable</td>
</tr>
<tr>
<td>lit-smallest</td>
<td>1</td>
<td>$C_m(p)$, where $m$ is the size of the smallest unsatisfied clause</td>
</tr>
<tr>
<td>neg-lit-smallest</td>
<td>1</td>
<td>$C_m(\neg p)$, as above, for negated literal</td>
</tr>
<tr>
<td>jw</td>
<td>1</td>
<td>$J(p)$ Jeroslow–Wang cue, see main text</td>
</tr>
<tr>
<td>jw-neg</td>
<td>1</td>
<td>$J(\neg p)$ Jeroslow–Wang cue, see main text</td>
</tr>
<tr>
<td>activity</td>
<td>1</td>
<td>minisat activity measure</td>
</tr>
<tr>
<td>time-since-active</td>
<td>1</td>
<td>$t - T(p)$ time since last activity (see main text)</td>
</tr>
<tr>
<td>has-been-active</td>
<td>1</td>
<td>1 if this $p$ has ever appeared in a conflict clause; 0 otherwise</td>
</tr>
</tbody>
</table>

Nearly all recently proposed heuristics can be represented linearly in this space.
Theorem

There is a margin for Horn clauses in our feature space.

Theorem

Under our feature space, $\mathcal{H}$ contains a priority function that recognizes 2–SAT sentences in polynomial time.
Overview

- The Davis-Putnam-Logemann-Loveland Algorithm
- Perceptron Learning
- Some Theoretical Results
- Feature Space
- Empirical Results
Planar Graph Coloring

- Starting with an empty $L \times L$ grid, sample $K$ cells at random and labelled them $1 \ldots K$.
- Repeatedly picked a labelled cell with at least one unlabelled neighbour and copied its label to its neighbour until all cells are labelled.
- Form a $K \times K$ adjacency matrix $A$ with $A_{ij} = 1$ iff there is a pair of adjacent cells with labels $i$ and $j$.
- Generate a SAT sentence over $4K$ variables (each variable corresponds to a particular colouring of a particular vertex).
  - Clauses express the constraints that each vertex must be assigned one and only one colour.
  - No pair of adjacent vertices may be assigned the same colour.
Planar graph coloring
Selection of problems from the annual SAT competition.

Tens of thousands of variables per sentence.

No known polynomial time algorithm for this problem.

Large economic impact – taken from a real industrial problem for which companies invest large amounts of money to develop commercial systems.

We have extended the popular Minisat system, which is the basis for many submissions to the SAT competition.
SAT is a canonical NP-complete decision problem.

In real world settings, SAT instances have shared characteristics and substructures, meaning it can be approached as a learning problem.

Polynomial time solvability $\iff$ polynomial time separability.

Learning can be done in a bounded number of simple linear updates.

(Probably) unseparable data still shows strong empirical improvement.

Order of magnitude improvement over a popular SAT solver on a hardware verification task.
Contact

Matthew Blaschko
Center for Visual Computing
École Centrale Paris & INRIA Saclay - Île-de-France

matthew.blaschko@inria.fr