Learning Selection Strategies in Buchberger’s Algorithm

Dylan Peifer

Cornell University

31 October 2019
The efficiency of Buchberger’s algorithm strongly depends on a choice of selection strategy. By phrasing Buchberger’s algorithm as a reinforcement learning problem and applying standard reinforcement learning techniques we can learn new selection strategies that can match or beat the existing state-of-the-art.

1. Gröbner Bases and Buchberger’s Algorithm
2. Reinforcement Learning and Policy Gradient
3. Results
1. Gröbner Bases and Buchberger’s Algorithm
\[ R = K[x_1, \ldots, x_n] \] a polynomial ring over some field \( K \)

\[ I = \langle f_1, \ldots, f_k \rangle \subseteq R \] an ideal generated by \( f_1, \ldots, f_k \in R \)
\[ R = K[x_1, \ldots, x_n] \quad \text{a polynomial ring over some field } K \]

\[ I = \langle f_1, \ldots, f_k \rangle \subseteq R \quad \text{an ideal generated by } f_1, \ldots, f_k \in R \]

**Example**

\[ R = \mathbb{Q}[x, y] = \{ \text{polynomials in } x \text{ and } y \text{ with rational coefficients} \} \]

\[ I = \langle x^2 - y^3, xy^2 + x \rangle = \{ a(x^2 - y^3) + b(xy^2 + x) : a, b \in R \} \]
\[ R = K[x_1, \ldots, x_n] \] a polynomial ring over some field \( K \)

\[ I = \langle f_1, \ldots, f_k \rangle \subseteq R \] an ideal generated by \( f_1, \ldots, f_k \in R \)

**Example**

\[
R = \mathbb{Q}[x, y] = \{ \text{polynomials in } x \text{ and } y \text{ with rational coefficients} \}
\]

\[
I = \langle x^2 - y^3, xy^2 + x \rangle = \{ a(x^2 - y^3) + b(xy^2 + x) : a, b \in R \}
\]

**Question**

*In the above example, is \( x^5 + x \) an element of \( I \)?
Question

Consider the ideal \( I = \langle x^2 + x - 2 \rangle \) in the ring \( \mathbb{Q}[x] \). Is \( x^3 + 3x^2 + 5x + 4 \) an element of \( I \)?
Question

Consider the ideal \( I = \langle x^2 + x - 2 \rangle \) in the ring \( \mathbb{Q}[x] \). Is \( x^3 + 3x^2 + 5x + 4 \) an element of \( I \)?

\[
x^2 + x - 2 \quad \overline{\quad x + 2}
\]

\[
x^3 + 3x^2 + 5x + 4
\]

\[
- \quad (x^3 + x^2 - 2x)
\]

\[
2x^2 + 7x + 4
\]

\[
- \quad (2x^2 + 2x - 4)
\]

\[
5x + 8
\]
Question

Consider the ideal \( I = \langle x^2 + x - 2 \rangle \) in the ring \( \mathbb{Q}[x] \). Is \( x^3 + 3x^2 + 5x + 4 \) an element of \( I \)?

\[
\begin{align*}
x^2 + x - 2 & \quad \longdiv{x^3 + 3x^2 + 5x + 4} \\
& \quad - (x^3 + x^2 - 2x) \\
& \quad 2x^2 + 7x + 4 \\
& \quad - (2x^2 + 2x - 4) \\
& \quad 5x + 8
\end{align*}
\]

\[x^3 + 3x^2 + 5x + 4 = (x + 2)(x^2 + x - 2) + (5x + 8)\]
Question

Consider the ideal \( I = \langle x^2 + x - 2 \rangle \) in the ring \( \mathbb{Q}[x] \). Is \( x^3 + 3x^2 + 5x + 4 \) an element of \( I \)?

\[
x^2 + x - 2 \overline{) x^3 + 3x^2 + 5x + 4}
\]
\[
\begin{array}{rl}
\quad & x + 2 \\
\quad - & (x^3 + x^2 - 2x) \\
\hline
\quad & 2x^2 + 7x + 4 \\
\quad - & (2x^2 + 2x - 4) \\
\hline
\quad & 5x + 8
\end{array}
\]

\[
x^3 + 3x^2 + 5x + 4 = (x + 2)(x^2 + x - 2) + (5x + 8)
\]

\[
\implies x^3 + 3x^2 + 5x + 4 \not\in \langle x^2 + x - 2 \rangle
\]
Definition
Let $x^\alpha$ denote an arbitrary monomial where $\alpha$ is the vector of exponents. A monomial order on $R = k[x_1, \ldots, x_n]$ is a relation $>$ on the monomials of $R$ such that

1. $>$ is a total ordering
2. $>$ is a well-ordering
3. if $x^\alpha > x^\beta$ then $x^\gamma x^\alpha > x^\gamma x^\beta$ for any $x^\gamma$ (i.e., $>$ respects multiplication).

Example
Lexicographic order (lex) is defined by $x^\alpha > x^\beta$ if the leftmost nonzero component of $\alpha - \beta$ is positive. For example, $x > y > z$, $xy > y^4$, and $xz > y^2$. 
Definition
Let $x^\alpha$ denote an arbitrary monomial where $\alpha$ is the vector of exponents. A monomial order on $R = k[x_1, \ldots, x_n]$ is a relation $>$ on the monomials of $R$ such that

1. $>$ is a total ordering
2. $>$ is a well-ordering
3. if $x^\alpha > x^\beta$ then $x^\gamma x^\alpha > x^\gamma x^\beta$ for any $x^\gamma$ (i.e., $>$ respects multiplication).

Example
Lexicographic order (lex) is defined by $x^\alpha > x^\beta$ if the leftmost nonzero component of $\alpha - \beta$ is positive. For example, $x > y > z$, $xy > y^4$, and $xz > y^2$. 
Divide $x^5 + x$ by the generators $x^2 - y^3$ and $xy^2 + x$
Divide $x^5 + x$ by the generators $x^2 - y^3$ and $xy^2 + x$

\[
\begin{array}{c|c c c c}
 & x^3 & -xy & & \\
q_1: & x^2 & -y & + & 1 \\
q_2: & x^2y & -y^2 & + & 0 \\
\hline
x^2 & x^5 & +x & & \\
x^2-xy^3 & - & (x^5-xy^3) & & \\
x^3y^2 & +x & & & \\
x^3y & - (x^3y^2+xy^4) & & & \\
x^2y^2 & +x & & & \\
x^2 & - (x^2+y^2) & & & \\
0 & & & & \\
\end{array}
\]

\[x^5 + x = (x^3 - xy)(x^2 - y^3) + (x^2y - y^2 + 1)(xy^2 + x) + 0\]
Divide \( x^5 + x \) by the generators \( x^2 - y^3 \) and \( xy^2 + x \)

\[
x^2 - y^3
\]

\[
xy^2 + x
\]

\[
\begin{align*}
q_1 : & \quad x^3 - xy \\
q_2 : & \quad x^2y - y^2 + 1 \\
\hline
& \quad x^5 + x \\
- & \quad (x^5 - x^3y^3) \\
\hline
& \quad x^3y^3 + x \\
- & \quad (x^3y^3 + x^3y) \\
\hline
& \quad -x^3y + x \\
- & \quad (-x^3y + xy^4) \\
\hline
& \quad -xy^4 + x \\
- & \quad (-xy^4 - xy^2) \\
\hline
& \quad xy^2 + x \\
- & \quad (xy^2 + x) \\
\hline
0
\end{align*}
\]

\[
x^5 + x = (x^3 - xy)(x^2 - y^3) + (x^2y - y^2 + 1)(xy^2 + x) + 0
\]

\[\implies x^5 + x \in \langle x^2 - y^3, \ xy^2 + x \rangle\]
Definition

When $F$ is set of polynomials and dividing $h$ by the $f_i \in F$ using the division algorithm leads to the remainder $r$ we write $h^F \rightarrow r$ or say $h$ reduces to $r$.

Lemma

If $h^F \rightarrow 0$ then $h$ is in the ideal generated by $F$.

Unfortunately, the converse is false.

Example

Using the same ideal $I = \langle x^2 - y^3, xy^2 + x \rangle$, note that $y^2(x^2 - y^3) - x(y^2 + x) = -x^2 - y^5 \in I$ however, multivariate division produces the nonzero remainder $-y^5 - y^3$. 
Definition
When $F$ is set of polynomials and dividing $h$ by the $f_i \in F$ using the division algorithm leads to the remainder $r$ we write $h^F \rightarrow r$ or say $h$ reduces to $r$.

Lemma
If $h^F \rightarrow 0$ then $h$ is in the ideal generated by $F$. 
Definition

When $F$ is set of polynomials and dividing $h$ by the $f_i \in F$ using the division algorithm leads to the remainder $r$ we write $h^F \rightarrow r$ or say $h$ reduces to $r$.

Lemma

If $h^F \rightarrow 0$ then $h$ is in the ideal generated by $F$.

Unfortunately, the converse is false.

Example

Using the same ideal $I = \langle x^2 - y^3, xy^2 + x \rangle$, note that

$$y^2(x^2 - y^3) - x(xy^2 + x) = -x^2 - y^5 \in I$$

However, multivariate division produces the nonzero remainder $-y^5 - y^3$.
Definition

Given a monomial order, a **Gröbner basis** $G$ of a nonzero ideal $I$ is a set of generators $\{g_1, g_2, \ldots, g_s\}$ of $I$ such that any of the following equivalent conditions hold:

\[(i) \quad f^G \to 0 \iff f \in I\]
\[(ii) \quad f^G \text{ is unique for all } f \in R\]
\[(iii) \quad \langle \text{LT}(g_1), \text{LT}(g_2), \ldots, \text{LT}(g_s) \rangle = \langle \text{LT}(I) \rangle\]

where $\text{LT}(f)$ is the **leading term** of $f$ and $\langle \text{LT}(I) \rangle = \langle \text{LT}(f) \mid f \in I \rangle$ is the ideal generated by all leading terms of $I$. 

Example

Using the same ideal $I = \langle x^2 - y^3, xy^2 + x \rangle$, the set $\{x^2 - y^3, xy^2 + x\}$ is not a Gröbner basis of $I$. 
Definition
Given a monomial order, a Gröbner basis $G$ of a nonzero ideal $I$ is a set of generators $\{g_1, g_2, \ldots, g_s\}$ of $I$ such that any of the following equivalent conditions hold:

(i) $f^G \to 0 \iff f \in I$

(ii) $f^G$ is unique for all $f \in R$

(iii) $\langle \text{LT}(g_1), \text{LT}(g_2), \ldots, \text{LT}(g_s) \rangle = \langle \text{LT}(I) \rangle$

where $\text{LT}(f)$ is the leading term of $f$ and $\langle \text{LT}(I) \rangle = \langle \text{LT}(f) \mid f \in I \rangle$ is the ideal generated by all leading terms of $I$.

Example
Using the same ideal $I = \langle x^2 - y^3, xy^2 + x \rangle$, the set $\{x^2 - y^3, xy^2 + x\}$ is not a Gröbner basis of $I$. 
Definition

Let \( S(f, g) = \frac{x^\gamma}{\text{LT}(f)} f - \frac{x^\gamma}{\text{LT}(g)} g \) where \( x^\gamma \) is the least common multiple of the leading monomials of \( f \) and \( g \). This is the \textit{s-polynomial} of \( f \) and \( g \), where \( s \) stands for subtraction or syzygy.

Example

\[
S(x^2 - y^3, xy^2 + x) = \frac{x^2 y^2}{x^2 - y^3} (x^2 - y^3) - \frac{x^2 y^2}{xy^2 + x} (xy^2 + x) = y^2 (x^2 - y^3) - x (xy^2 + x) = -x^2 - y^5
\]
**Definition**

Let \( S(f, g) = \frac{x^\gamma}{\text{LT}(f)} f - \frac{x^\gamma}{\text{LT}(g)} g \) where \( x^\gamma \) is the least common multiple of the leading monomials of \( f \) and \( g \). This is the \( s \)-polynomial of \( f \) and \( g \), where \( s \) stands for subtraction or syzygy.

**Example**

\[
S(x^2 - y^3, xy^2 + x) = \frac{x^2 y^2}{x^2} (x^2 - y^3) - \frac{x^2 y^2}{xy^2} (xy^2 + x) \\
= y^2(x^2 - y^3) - x(xy^2 + x) \\
= -x^2 - y^5
\]
Definition

Let \( S(f, g) = \frac{x^\gamma}{\text{LT}(f)} f - \frac{x^\gamma}{\text{LT}(g)} g \) where \( x^\gamma \) is the least common multiple of the leading monomials of \( f \) and \( g \). This is the \textit{s-polynomial} of \( f \) and \( g \), where \( s \) stands for subtraction or syzygy.

Example

\[
S(x^2 - y^3, xy^2 + x) = \frac{x^2 y^2}{x^2} (x^2 - y^3) - \frac{x^2 y^2}{xy^2} (xy^2 + x) \\
= y^2(x^2 - y^3) - x(xy^2 + x) \\
= -x^2 - y^5
\]

Theorem (Buchberger's Criterion)

Let \( G = \{g_1, g_2, \ldots, g_s\} \) generate the ideal \( I \). If \( S(g_i, g_j)^G \rightarrow 0 \) for all pairs \( g_i, g_j \) then \( G \) is a Gröbner basis of \( I \).
Algorithm Buchberger’s Algorithm

input a set of polynomials \( \{f_1, \ldots, f_k\} \)

output a Gröbner basis \( G \) of \( I = \langle f_1, \ldots, f_k \rangle \)

procedure \textsc{Buchberger}(\{f_1, \ldots, f_k\})

\[ G \leftarrow \{f_1, \ldots, f_k\} \] \quad \text{▷ the current basis}

\[ P \leftarrow \{(f_i, f_j) \mid 1 \leq i < j \leq k\} \] \quad \text{▷ the remaining pairs}

while \( |P| > 0 \) do

\( (f_i, f_j) \leftarrow \text{select}(P) \)

\[ P \leftarrow P \setminus \{(f_i, f_j)\} \]

\[ r \leftarrow S(f_i, f_j)^G \]

if \( r \neq 0 \) then

\[ P \leftarrow P \cup \{(f, r) : f \in G\} \]

\[ G \leftarrow G \cup \{r\} \]

end if

end while

return \( G \)

end procedure
Example
\[ I = \langle x^2 - y^3, xy^2 + x \rangle \]
Example

\[ I = \langle x^2 - y^3, xy^2 + x \rangle \]

initialize \( G \) to \( \{ x^2 - y^3, xy^2 + x \} \)
initialize \( P \) to \( \{ (x^2 - y^3, xy^2 + x) \} \)
Example

\[ I = \langle x^2 - y^3, xy^2 + x \rangle \]

initialize \( G \) to \( \{ x^2 - y^3, xy^2 + x \} \)
initialize \( P \) to \( \{ (x^2 - y^3, xy^2 + x) \} \)

select \( (x^2 - y^3, xy^2 + x) \) and compute \( S(x^2 - y^3, xy^2 + x)^G \rightarrow -y^5 - y^3 \)
update \( G \) to \( \{ x^2 - y^3, xy^2 + x, -y^5 - y^3 \} \)
update \( P \) to \( \{ (x^2 - y^3, -y^5 - y^3), (xy^2 + x, -y^5 - y^3) \} \)
Example

\[ I = \langle x^2 - y^3, xy^2 + x \rangle \]

initialize \( G \) to \( \{x^2 - y^3, xy^2 + x\} \)
initialize \( P \) to \( \{(x^2 - y^3, xy^2 + x)\} \)

select \((x^2 - y^3, xy^2 + x)\) and compute \( S(x^2 - y^3, xy^2 + x)^G \rightarrow -y^5 - y^3 \)
update \( G \) to \( \{x^2 - y^3, xy^2 + x, -y^5 - y^3\} \)
update \( P \) to \( \{(x^2 - y^3, -y^5 - y^3), (xy^2 + x, -y^5 - y^3)\} \)

select \((x^2 - y^3, -y^5 - y^3)\) and compute \( S(x^2 - y^3, -y^5 - y^3)^G \rightarrow 0 \)
Example

\[ I = \langle x^2 - y^3, xy^2 + x \rangle \]

initialize \( G \) to \( \{x^2 - y^3, xy^2 + x\} \)
initialize \( P \) to \( \{(x^2 - y^3, xy^2 + x)\} \)

select \((x^2 - y^3, xy^2 + x)\) and compute \( S(x^2 - y^3, xy^2 + x)^G \rightarrow -y^5 - y^3 \)
update \( G \) to \( \{x^2 - y^3, xy^2 + x, -y^5 - y^3\} \)
update \( P \) to \( \{(x^2 - y^3, -y^5 - y^3), (xy^2 + x, -y^5 - y^3)\} \)

select \((x^2 - y^3, -y^5 - y^3)\) and compute \( S(x^2 - y^3, -y^5 - y^3)^G \rightarrow 0 \)

select \((xy^2 + x, -y^5 - y^3)\) and compute \( S(xy^2 + x, -y^5 - y^3)^G \rightarrow 0 \)
Example

\[ l = \langle x^2 - y^3, xy^2 + x \rangle \]

initialize \( G \) to \( \{ x^2 - y^3, xy^2 + x \} \)
initialize \( P \) to \( \{ (x^2 - y^3, xy^2 + x) \} \)

select \( (x^2 - y^3, xy^2 + x) \) and compute \( S(x^2 - y^3, xy^2 + x)^G \rightarrow -y^5 - y^3 \)
update \( G \) to \( \{ x^2 - y^3, xy^2 + x, -y^5 - y^3 \} \)
update \( P \) to \( \{ (x^2 - y^3, -y^5 - y^3), (xy^2 + x, -y^5 - y^3) \} \)

select \( (x^2 - y^3, -y^5 - y^3) \) and compute \( S(x^2 - y^3, -y^5 - y^3)^G \rightarrow 0 \)

select \( (xy^2 + x, -y^5 - y^3) \) and compute \( S(xy^2 + x, -y^5 - y^3)^G \rightarrow 0 \)

return \( G = \{ x^2 - y^3, xy^2 + x, -y^5 - y^3 \} \)
**Algorithm**  Buchberger’s Algorithm

**input**  a set of polynomials \( \{f_1, \ldots, f_k\} \)

**output**  a Gröbner basis \( G \) of \( I = \langle f_1, \ldots, f_k \rangle \)

**procedure**  \textsc{Buchberger}(\{\( f_1, \ldots, f_k \}\))

\[
\begin{align*}
G & \leftarrow \{f_1, \ldots, f_k\} & \text{▷ the current basis} \\
P & \leftarrow \{(f_i, f_j) \mid 1 \leq i < j \leq k\} & \text{▷ the remaining pairs} \\
\text{while } |P| > 0 \text{ do} & \\
(f_i, f_j) & \leftarrow \text{select}(P) \\
P & \leftarrow P \setminus \{(f_i, f_j)\} \\
r & \leftarrow S(f_i, f_j)^G \\
\text{if } r \neq 0 \text{ then} & \\
P & \leftarrow P \cup \{(f, r) : f \in G\} \\
G & \leftarrow G \cup \{r\} \\
\text{end if} & \\
\text{end while} & \\
\text{return } G & \\
\text{end procedure}
\]
In general, we should select “small” pairs \((f_i, f_j)\) first.
In general, we should select “small” pairs \((f_i, f_j)\) first.

- **First:**
  among the pairs with minimal \(j\), pick the pair with smallest \(i\)

- **Degree:**
  pick the pair with smallest degree of \(\text{lcm}(\text{LT}(f_i), \text{LT}(f_j))\)

- **Normal:**
  pick the pair with smallest \(\text{lcm}(\text{LT}(f_i), \text{LT}(f_j))\) in the monomial order

- **Sugar:**
  pick the pair with smallest sugar degree of \(\text{lcm}(\text{LT}(f_i), \text{LT}(f_j))\), which is the degree it would have had if we had homogenized at the beginning
The number of pair reductions performed is a rough estimate of how much time was spent. Smaller numbers are better.

<table>
<thead>
<tr>
<th>example</th>
<th>First</th>
<th>Degree</th>
<th>Normal</th>
<th>Sugar</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>cyclic6</td>
<td>371</td>
<td>655</td>
<td>620</td>
<td>343</td>
<td>793</td>
</tr>
<tr>
<td>cyclic7</td>
<td>2217</td>
<td>5664</td>
<td>5781</td>
<td>2070</td>
<td>-</td>
</tr>
<tr>
<td>katsura7</td>
<td>164</td>
<td>164</td>
<td>164</td>
<td>164</td>
<td>285</td>
</tr>
<tr>
<td>eco6</td>
<td>67</td>
<td>72</td>
<td>61</td>
<td>64</td>
<td>97</td>
</tr>
<tr>
<td>reimer5</td>
<td>552</td>
<td>212</td>
<td>211</td>
<td>301</td>
<td>-</td>
</tr>
<tr>
<td>noon4</td>
<td>71</td>
<td>71</td>
<td>71</td>
<td>71</td>
<td>100</td>
</tr>
<tr>
<td>cyclic5 (lex)</td>
<td>112</td>
<td>132</td>
<td>1602</td>
<td>108</td>
<td>-</td>
</tr>
<tr>
<td>katsura5 (lex)</td>
<td>231</td>
<td>1631</td>
<td>769</td>
<td>67</td>
<td>-</td>
</tr>
<tr>
<td>eco5 (lex)</td>
<td>30</td>
<td>34</td>
<td>22</td>
<td>26</td>
<td>28</td>
</tr>
<tr>
<td>eco6 (lex)</td>
<td>104</td>
<td>147</td>
<td>96</td>
<td>68</td>
<td>175</td>
</tr>
</tbody>
</table>
Summary

- A Gröbner basis of an ideal in a polynomial ring is a special generating set that is useful for many computational problems.
- Buchberger’s algorithm produces a Gröbner basis from any initial generating set of an ideal by repeatedly choosing pairs \((f_i, f_j)\) of the current generating set and adding the reduction of the s-polynomial of \(f_i\) and \(f_j\) to the generating set if it is not zero.
- The selection strategy used to pick which pair to choose next can make a big difference in the efficiency of Buchberger’s algorithm.
2. Reinforcement Learning and Policy Gradient
Reinforcement learning tries to understand and optimize goal-directed behavior driven by interaction with the world.
Reinforcement learning tries to understand and optimize goal-directed behavior driven by interaction with the world.

- playing games (backgammon, chess, Go, StarCraft, ...)
- flying a helicopter or driving a car
- controlling a power station or data center
- managing a portfolio of stocks or other financial assets
- allocating resources to research projects
Reinforcement learning problems can be phrased as the interaction of an agent and an environment.

The agent chooses actions and the environment processes actions and gives back the updated state and a reward. The agent wants to maximize its return, which is the amount of reward it gets in the long run.
Definition

A Markov Decision Process (MDP) is a collection of states $S$ and actions $A$ with transition dynamics given by

$$ p : S \times \mathbb{R} \times S \times A \rightarrow [0, 1] $$

where

$$ p(s', r|s, a) = \Pr[S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a] $$

returns the probability that the next state is $s'$ and the next reward is $r$ given that the current state is $s$ and the chosen action is $a$. 
Definition

A Markov Decision Process (MDP) is a collection of states $S$ and actions $A$ with transition dynamics given by

$$p : S \times \mathbb{R} \times S \times A \to [0, 1]$$

where

$$p(s', r|s, a) = \Pr[S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a]$$

returns the probability that the next state is $s'$ and the next reward is $r$ given that the current state is $s$ and the chosen action is $a$.

An environment implements an MDP by computing $p(\cdot, \cdot|s, a)$ for the current state $s$ and action $a$ provided by the agent and then sampling from the resulting distribution to return a new state $s'$ and reward $r$. 
State: the positions of all pieces on the board
Action: a valid move of one of your pieces
Reward: 1 if you win immediately after the transition, otherwise 0
CartPole

State: the cart and pole positions and velocities
Action: push the cart left or right
Reward: 1 for every transition the pole is still upright
Definition
A policy $\pi$ is a function

$$\pi : A \times S \rightarrow [0, 1]$$

where

$$\pi(a|s) = Pr(A_t = a|S_t = s)$$

returns the probability that the next action is $a$ given that the current state is $s$. 
Definition

A policy \( \pi \) is a function

\[
\pi : A \times S \to [0, 1]
\]

where

\[
\pi(a|s) = Pr(A_t = a|S_t = s)
\]

returns the probability that the next action is \( a \) given that the current state is \( s \).

An agent follows a policy by computing \( \pi(\cdot|s) \) for the current state \( s \) and sampling from the resulting probability distribution to choose the next action.
Definition

A trajectory, episode, or rollout $\tau$ of a policy $\pi$ is a series of states, actions, and rewards $(S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots, R_T, S_T)$ obtained by following the policy $\pi$ one time through the environment.

Definition

The return of a trajectory is the sum of rewards

$$\sum_{t=1}^{T} R_t$$

along the trajectory.
The Reinforcement Learning Problem

Given an MDP, determine a policy $\pi$ that maximizes the expected return

$$\mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=1}^{T} R_t \right]$$

over full trajectories sampled by following the policy $\pi$. 
The Reinforcement Learning Problem

Given an MDP, determine a policy $\pi$ that maximizes the expected return

$$\mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=1}^{T} R_t \right]$$

over full trajectories sampled by following the policy $\pi$.

If we know the exact transition dynamics of the MDP this is a planning problem. In the full learning problem the dynamics are either unknown or infeasible to compute. All we can do is sample from the environment.
Consider a parametrized policy function $\pi_\theta$ which maps states to probability distributions on actions. The expected return is now a function

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[ \sum_{t=1}^{T} R_t \right]$$

of the parameters $\theta$ of the policy.
Consider a parametrized policy function $\pi_\theta$ which maps states to probability distributions on actions. The expected return is now a function

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[ \sum_{t=1}^{T} R_t \right]$$

of the parameters $\theta$ of the policy.

Starting from any value of the parameters $\theta_1$, we can improve the policy by repeatedly moving the parameters in the direction of $\nabla_\theta J(\theta)$

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\theta)|_{\theta_k}$$

where $\alpha$ is some small learning rate.
Theorem (Policy Gradient Theorem)

Suppose $\pi_\theta$ is a parametrized policy that is differentiable with respect to its parameters $\theta$. Then the gradient of

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[ \sum_{t=1}^{T} R_t \right]$$

is

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t | S_t) \sum_{t'=t+1}^{T} R_{t'} \right] .$$

Intuitively, we should increase the probability of taking the action we chose proportional to the future reward we received and the derivative of the log probability of choosing that action again.
Theorem (Policy Gradient Theorem)

Suppose $\pi_\theta$ is a parametrized policy that is differentiable with respect to its parameters $\theta$. Then the gradient of

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[ \sum_{t=1}^{T} R_t \right]$$

is

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t|S_t) \sum_{t'=t+1}^{T} R_{t'} \right].$$

Intuitively, we should increase the probability of taking the action we chose proportional to the future reward we received and the derivative of the log probability of choosing that action again.
Reinforcement learning can be phrased as the interaction of an agent and an environment, where an agent picks actions and is trying to maximize the total reward it receives from the environment over a full trajectory.

A policy is a function that takes in a state and returns a probability distribution on actions.

Policy gradient methods improve a parametrized policy by moving the parameters in the direction of the gradient of expected return.
3. Results
Algorithm Buchberger’s Algorithm

input a set of polynomials \( \{f_1, \ldots, f_k\} \)

output a Gröbner basis \( G \) of \( I = \langle f_1, \ldots, f_k \rangle \)

procedure \textsc{Buchberger}(\{f_1, \ldots, f_k\})

\[
G \leftarrow \{f_1, \ldots, f_k\} \\
\triangleright \text{the current basis} \\
P \leftarrow \{(f_i, f_j) \mid 1 \leq i < j \leq k\} \\
\triangleright \text{the remaining pairs}
\]

while \(|P| > 0\) do

\[
(f_i, f_j) \leftarrow \text{select}(P) \\
P \leftarrow P \setminus \{(f_i, f_j)\} \\
r \leftarrow S(f_i, f_j)^G
\]

if \( r \neq 0 \) then

\[
P \leftarrow P \cup \{(f, r) : f \in G\} \\
G \leftarrow G \cup \{r\}
\]

end if

end while

return \( G \)

end procedure
Buchberger

\[ G = \{ x^2 - y^3, xy^2 + x, -y^5 - y^3 \} \]
\[ P = \{ (x^2 - y^3, -y^5 - y^3), (xy^2 + x, -y^5 - y^3) \} \]

State: the current basis and pair set
Action: a pair from the pair set
Reward: -1 for every transition until the pair set is empty
\[ \vec{h} = \sigma_1(W_1 \vec{x} + \vec{b}_1) \]
\[ \vec{y} = \sigma_2(W_2 \vec{h} + \vec{b}_2) \]
\[ G = \{xy^6 + 9y^2z^4, z^4 + 1212z, xy^3 + 961xy^2, x^4yz + 12518xz, xyz^2 + 20y\} \]
\[ P = \{(1, 2), (1, 3), (2, 3), (1, 4), (2, 4), (3, 4), (1, 5), (2, 5), (3, 5), (4, 5)\} \]
$G = \{ xy^6 + 9y^2z^4, z^4 + 1212z, xy^3 + 961xy^2, x^4yz + 12518xz, xyz^2 + 20y \}$

$P = \{ (1, 2), (1, 3), (2, 3), (1, 4), (2, 4), (3, 4), (1, 5), (2, 5), (3, 5), (4, 5) \}$

Fix a number $n$ of variables and pick a fixed number $k$ of lead monomials that the agent will be able to see. Concatenate the exponent vectors of the lead $k$ terms in each pair. Place each pair in the row of a matrix.

$$
\begin{bmatrix}
1 & 6 & 0 & 0 & 2 & 4 & 0 & 0 & 4 & 0 & 0 & 1 \\
1 & 6 & 0 & 0 & 2 & 4 & 1 & 3 & 0 & 1 & 2 & 0 \\
0 & 0 & 4 & 0 & 0 & 1 & 1 & 3 & 0 & 1 & 2 & 0 \\
1 & 6 & 0 & 0 & 2 & 4 & 4 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 4 & 0 & 0 & 1 & 4 & 1 & 1 & 1 & 0 & 1 \\
1 & 3 & 0 & 1 & 2 & 0 & 4 & 1 & 1 & 1 & 0 & 1 \\
1 & 6 & 0 & 0 & 2 & 4 & 1 & 1 & 2 & 0 & 1 & 0 \\
0 & 0 & 4 & 0 & 0 & 1 & 1 & 1 & 2 & 0 & 1 & 0 \\
1 & 3 & 0 & 1 & 2 & 0 & 1 & 1 & 2 & 0 & 1 & 0 \\
4 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 2 & 0 & 1 & 0
\end{bmatrix}
$$
(IPI, 1, 2nk) \rightarrow 1x1 \text{ Conv2D} + \text{ReLU} \rightarrow (IPI, 1, h) \rightarrow 1x1 \text{ Conv2D} + \text{softmax} \rightarrow (IPI, 1, 1)
The network weights are initialized randomly. Training then proceeds through epochs. In each epoch:

1. Perform 100 rollouts using the current policy network.
2. Compute future rewards for each action on each trajectory, baseline by the size of the current pair set in the state, and normalize these scores across the epoch.
3. Update the policy network using gradient ascent and the policy gradient theorem.
The network weights are initialized randomly. Training then proceeds through epochs. In each epoch:

1. Perform 100 rollouts using the current policy network.
The network weights are initialized randomly. Training then proceeds through epochs. In each epoch:

1. Perform 100 rollouts using the current policy network.
2. Compute future rewards for each action on each trajectory, baseline by the size of the current pair set in the state, and normalize these scores across the epoch.
The network weights are initialized randomly. Training then proceeds through epochs. In each epoch:

1. Perform 100 rollouts using the current policy network.
2. Compute future rewards for each action on each trajectory, baseline by the size of the current pair set in the state, and normalize these scores across the epoch.
3. Update the policy network using gradient ascent and the policy gradient theorem.
Example 1: Matching Degree
- \( R = \mathbb{Z}/32003[x, y, z] \), grevlex ordering
- ideals generated by 5 random binomials of homogeneous degree 5
- agent sees only lead monomials, and network has one hidden layer of size 48 (385 parameters)
- total training time of 15 minutes
Before training there is no relation between the degree of a pair and the agent’s preference. After training the agent clearly prefers pairs that have smaller degree.
Example 2: Better Performance
- \( R = \mathbb{Z}/32003[x, y, z] \), grevlex ordering
- ideals generated by 10 random binomials of degree \( \leq 20 \)
- agent sees lead two monomials, and network has two hidden layers of size 48 (3025 parameters)
- total training time of 8 hours
Example 3: Binned Ideals
$\mathbb{R} = \mathbb{Z}/32003[a, b, c, d, e]$, grevlex ordering

- ideals generated by 5 random binomials of degree $\leq 10$
- agent sees lead two monomials, and network has two hidden layers of size 64 (5569 parameters)
- total training time of 26 hours
Summary

- Policy gradient agents that only see lead terms learned strategies that approximate degree selection.
- Policy gradient agents that see full binomials learned strategies that performed 10-20% fewer pair reductions than known strategies.
- A major challenge is the high variance in how hard different Gröbner bases are to compute within the same distribution.