Learning Prediction Suffix Trees with Winnow

Nikos Karampatziakis    Dexter Kozen

Cornell University

June 15, 2009
Introduction

Sequential prediction

Our task: Is a program behaving normally?

- Monitor a program and take appropriate actions based on the actual system calls and the predicted ones
- Deviations may signify a bug, a security problem, etc.
Introduction

Sequential prediction

Algorithm that learns small and accurate Prediction Suffix Trees (PSTs)
Introduction

Sequential prediction

Algorithm that learns small and accurate Prediction Suffix Trees (PSTs)

Our task: Is a program behaving normally?
Sequential prediction

Algorithm that learns small and accurate Prediction Suffix Trees (PSTs)

Our task: Is a program behaving normally?
  - Monitor a program and take appropriate actions based on the actual system calls and the predicted ones
Introduction

Sequential prediction

Algorithm that learns small and accurate Prediction Suffix Trees (PSTs)

Our task: Is a program behaving normally?

- Monitor a program and take appropriate actions based on the actual system calls and the predicted ones
- Deviations may signify a bug, a security problem, etc.
Known alphabet e.g. \{A, C, G, T\} or \{-1, +1\} or \{open(), read(), \ldots \} or \ldots
Sequential Prediction

Known alphabet e.g. \{A, C, G, T\} or \{-1, +1\} or \{open(), read(), \ldots\} or \ldots

Given $y_1, y_2, \ldots, y_{t-2}, y_{t-1}$ predict $y_t$
Known alphabet e.g. $\{A, C, G, T\}$ or $\{-1, +1\}$ or $\{\text{open()}, \text{read()}, \ldots \}$ or ... 

Given $y_1, y_2, \ldots, y_{t-2}, y_{t-1}$ predict $y_t$

PSTs (aka Context Trees) are popular models for this task [Pereira & Singer, 1999, Ron et al., 1996, Willems et al., 1995]
Assume $y_t \in \{-1, +1\}$
Assume $y_t \in \{-1, +1\}$

Now, a PST is a binary tree
Assume $y_t \in \{-1, +1\}$

Now, a PST is a binary tree

Each node has a value
Assume $y_t \in \{-1, +1\}$

Now, a PST is a binary tree

Each node has a value

Left links labeled $-1$, right $+1$
Assume $y_t \in \{-1, +1\}$

Now, a PST is a binary tree

Each node has a value

Left links labeled $-1$, right $+1$

To predict $y_t$ follow the path labeled $y_{t-1}, y_{t-2}, \ldots$
Assume \( y_t \in \{-1, +1\} \)

Now, a PST is a binary tree

Each node has a value

Left links labeled \(-1\), right \(+1\)

To predict \( y_t \) follow the path labeled \( y_{t-1}, y_{t-2}, \ldots \)

\( y_t \) is the sign of a weighted sum of visited values
Assume $y_t \in \{-1, +1\}$

Now, a PST is a binary tree

Each node has a value

Left links labeled $-1$, right $+1$

To predict $y_t$ follow the path labeled $y_{t-1}, y_{t-2}, \ldots$

$y_t$ is the sign of a weighted sum of visited values

Earlier symbols are discounted more than recent ones
Example

Discounting: Values discounted by $\left(\frac{1}{2}\right)^{\text{depth}}$

Input Sequence:
Decision:
Example

Discounting: Values discounted by \( \left( \frac{1}{2} \right)^{\text{depth}} \)

Input Sequence: \( \ldots, +1 \)

Decision: 0
Example

Discounting: Values discounted by \((\frac{1}{2})^{\text{depth}}\)

Input Sequence: \(\ldots, +1\)

Decision: \(0 + \frac{1}{2} \cdot -2\)
Example

Discounting: Values discounted by $(\frac{1}{2})^{\text{depth}}$

Input Sequence: \ldots, +1

Decision: \[ 0 + \frac{1}{2} \cdot -2 = -1 \text{ sign} -1 \]
Example

Discounting: Values discounted by $\left(\frac{1}{2}\right)^{\text{depth}}$

Input Sequence: $\ldots, +1, -1$

Decision: 0

Nikos Karampatziakis, Dexter Kozen
Learning Prediction Suffix Trees
Discounting: Values discounted by \(\left(\frac{1}{2}\right)^{\text{depth}}\)

Input Sequence: \(\ldots, +1, -1\)

Decision: \(0 + \frac{1}{2} \cdot -3\)
Discounting: Values discounted by $\left(\frac{1}{2}\right)^\text{depth}$

Input Sequence: $\ldots, +1, -1$

Decision: $0 + \frac{1}{2} \cdot -3 = -\frac{3}{2} \xrightarrow{\text{sign}} -1$
Example

Discounting: Values discounted by \((\frac{1}{2})^{\text{depth}}\)

Input Sequence: \(\ldots, -1, -1\)

Decision: 0
Example

Discounting: Values discounted by \( \left( \frac{1}{2} \right)^{\text{depth}} \)

Input Sequence: \( \ldots, -1, -1 \)

Decision: \( 0 + \frac{1}{2} \cdot -3 \)
Example

Discounting: Values discounted by \((\frac{1}{2})^{\text{depth}}\)

Input Sequence: \(\ldots, -1, -1\)

Decision: \(0 + \frac{1}{2} \cdot -3 + \frac{1}{4} \cdot 7\)
Example

Discounting: Values discounted by \((\frac{1}{2})^{\text{depth}}\)

Input Sequence: \(\ldots, -1, -1\)

Decision: \[0 + \frac{1}{2} \cdot -3 + \frac{1}{4} \cdot 7 = \frac{1}{4} \text{ sign} \rightarrow +1\]

Nikos Karampatziakis, Dexter Kozen

Learning Prediction Suffix Trees
A Linear Prediction Problem

Node indexed by the suffix \( s \) leading to it
A Linear Prediction Problem

Node indexed by the suffix $s$ leading to it

Let $w_{t,s}$ be the values in the tree at time $t$
A Linear Prediction Problem

Node indexed by the suffix $s$ leading to it

Let $w_{t,s}$ be the values in the tree at time $t$

Decision can be written as $\text{sign}(\langle w_t, x_t^+ \rangle)$

$$x_{t,s}^+ = \begin{cases} 
\beta^{|s|} & \text{if } s \text{ is a suffix of } y_1, \ldots, y_{t-1} \\
0 & \text{otherwise}
\end{cases}$$
A Linear Prediction Problem

Node indexed by the suffix $s$ leading to it

Let $w_{t,s}$ be the values in the tree at time $t$

Decision can be written as $\text{sign}(\langle w_t, x_t^+ \rangle)$

$$x_{t,s}^+ = \begin{cases} 
\beta |s| & \text{if } s \text{ is a suffix of } y_1, \ldots, y_{t-1} \\
0 & \text{otherwise}
\end{cases}$$

Online setting
\[ \theta_1 \leftarrow 0 \]

**for** \( t = 1, \ldots, T \) **do**

\[ w_{t,i} \leftarrow \frac{e^{\theta_{t,i}}}{\sum_j e^{\theta_{t,j}}} \]

\[ \hat{y}_t \leftarrow \langle w_t, x_t \rangle \]

**if** \( y_t \hat{y}_t \leq 0 \)

\[ \theta_{t+1} \leftarrow \theta_t + \alpha y_t x_t \]

**else**

\[ \theta_{t+1} \leftarrow \theta_t \]
Balanced Winnow

\[ \theta_1 \leftarrow 0 \]

\[ \textbf{for } t = 1, \ldots, T \textbf{ do} \]

\[ w_{t,i} \leftarrow \frac{e^{\theta_{t,i}}}{\sum_j e^{\theta_{t,j}}} \]

\[ \hat{y}_t \leftarrow \langle w_t, x_t \rangle \]

\[ \textbf{if } y_t \hat{y}_t \leq 0 \]

\[ \theta_{t+1} \leftarrow \theta_t + \alpha y_t x_t \]

\[ \textbf{else} \]

\[ \theta_{t+1} \leftarrow \theta_t \]

Important

\[ x_t = [x_t^+, -x_t^+] = \]

\[ [x_{t,1}^+, \ldots, x_{t,d}^+, -x_{t,1}^+, \ldots, -x_{t,d}^+] \]
Balanced Winnow

\[ \theta_1 \leftarrow 0 \]

\textbf{for} \ t = 1, \ldots , T \ \textbf{do}

\[ w_{t,i} \leftarrow \frac{e^{\theta_{t,i}}}{\sum_j e^{\theta_{t,j}}} \]

\[ \hat{y}_t \leftarrow \langle w_t, x_t \rangle \]

\textbf{if} \ y_t \hat{y}_t \leq 0

\[ \theta_{t+1} \leftarrow \theta_t + \alpha y_t x_t \]

\textbf{else}

\[ \theta_{t+1} \leftarrow \theta_t \]

\textbf{Important}

\[ x_t = [x_t^+, -x_t^+] = \]

\[ [x_{t,1}^+, \ldots , x_{t,d}^+, -x_{t,1}^+, \ldots , -x_{t,d}^+] \]

\textbf{Known fact}

Let \( \theta_t = [\theta^+_t, \theta^-_t] \) then

\[ \langle w_t, x_t \rangle \propto \sum_i \sinh(\theta^+_t) x^+_t \]

\[ \sinh(\theta^+_t) = 0 \text{ iff } \theta^+_t = 0 \]
Winnow can learn good $\theta^+_t$ values for the tree.
Learning a PST

Winnow can learn good $\theta_t^+$ values for the tree.

To keep the tree small $\theta_t^+$ must be sparse.
Winnow can learn good $\theta^+_t$ values for the tree.

To keep the tree small $\theta^+_t$ must be sparse.

Initially $\theta_1 = 0$. The tree has one node.
Winnow can learn good $\theta_t^+$ values for the tree.

To keep the tree small $\theta_t^+$ must be sparse.

Initially $\theta_1 = 0$. The tree has one node.

As mistakes are made, the tree grows.
Winnow can learn good $\theta_t^+$ values for the tree.

To keep the tree small $\theta_t^+$ must be sparse.

Initially $\theta_1 = 0$. The tree has one node.

As mistakes are made, the tree grows.

Winnow/Perceptron update quickly leads to large trees.
Input: \ldots, -1, +1, -1, +1, -1, ?

\[ x_{t,s}^+ = \begin{cases} 
\left(\frac{1}{2}\right)^{|s|} & \text{if } s \text{ is a suffix} \\
0 & \text{otherwise} 
\end{cases} \]

Decision:
Input: \ldots, -1, +1, -1, +1, -1, ?

\[ x_{t,s}^+ = \begin{cases} \left(\frac{1}{2}\right)^{|s|} & \text{if } s \text{ is a suffix} \\ 0 & \text{otherwise} \end{cases} \]

Decision: \( \frac{1}{2} \cdot \sinh(1) \)
Input: \ldots, -1, +1, -1, +1, -1, ?

\[ x_{t,s}^+ = \begin{cases} 
(\frac{1}{2})^{|s|} & \text{if } s \text{ is a suffix} \\
0 & \text{otherwise}
\end{cases} \]

Decision: \( \frac{1}{2} \cdot \sinh(1) > 0 \)
Illustration

Input: \ldots, -1, +1, -1, +1, -1, ?

Decision: \( \frac{1}{2} \cdot \sinh(1) > 0 \overset{\text{sign}}{\rightarrow} +1 \)

\[ x_{t,s}^+ = \begin{cases} (\frac{1}{2})^{|s|} & \text{if } s \text{ is a suffix} \\ 0 & \text{otherwise} \end{cases} \]
Input: \ldots, -1, +1, -1, +1, -1, -1

Decision: \frac{1}{2} \cdot \sinh(1) > 0 \ \rightarrow \ +1

Update: \theta_{t+1,s}^+ = \theta_{t,s}^+ - x_{t,s}^+
Illustration

Input: \ldots, -1, +1, -1, +1, -1, -1

\[
x_{t,s}^+ = \begin{cases} 
(\frac{1}{2})^{\mid s \mid} & \text{if } s \text{ is a suffix} \\
0 & \text{otherwise}
\end{cases}
\]

Decision: $\frac{1}{2} \cdot \sinh(1) > 0 \xrightarrow{\text{sign}} +1$

Update: $\theta_{t+1,s}^+ = \theta_{t,s}^+ - x_{t,s}^+$

Nikos Karampatziakis, Dexter Kozen

Learning Prediction Suffix Trees
Illustration

Input: \ldots, -1, +1, -1, +1, -1, -1

Decision: \frac{1}{2} \cdot \sinh(1) > 0 \Rightarrow +1

Update: \theta^+_{t+1,s} = \theta^+_t, s - x^+_t, s

\begin{align*}
x^+_t, s &= \begin{cases} 
\left(\frac{1}{2}\right)^{|s|} & \text{if } s \text{ is a suffix} \\
0 & \text{otherwise}
\end{cases} \\
\theta^+_t, s &= \ldots
\end{align*}
Observations and Ideas

Mistake at time $t$: $O(t)$ nodes are inserted
Mistake at time $t$: $O(t)$ nodes are inserted

$x_{t,s}^+$ is non-zero even when $s$ is very long
Mistake at time $t$: $O(t)$ nodes are inserted

$x_{t,s}^+$ is non-zero even when $s$ is very long

**Bad idea:** change $x_{t,s}^+$ to avoid this
Mistake at time $t$: $O(t)$ nodes are inserted

$x^+_{t,s}$ is non-zero even when $s$ is very long

Bad idea: change $x^+_{t,s}$ to avoid this

Better idea: Have an adaptive bound $d_t$ on the depth up to which the tree can grow on round $t$. 

Nikos Karampatziakis, Dexter Kozen
Learning Prediction Suffix Trees
Mistake at time $t$: $O(t)$ nodes are inserted

$x_{t,s}^+$ is non-zero even when $s$ is very long

Bad idea: change $x_{t,s}^+$ to avoid this

Better idea: Have an adaptive bound $d_t$ on the depth up to which the tree can grow on round $t$.

- $d_t$ will be growing slowly if necessary
Winnow for PSTs

New update: \( \theta_{t+1} = \theta_t + \alpha y_t x_t + \alpha n_t \)
Winnow for PSTs

New update: \( \theta_{t+1} = \theta_t + \alpha y_t x_t + \alpha n_t \)

\( n_t \): a “noise” vector that prunes the tree

\[
n_{t,s} = \begin{cases} 
- y_t x_{t,s} & \text{if } |s| > d_t, \ s \text{ suffix} \\
0 & \text{otherwise}
\end{cases}
\]
Winnow for PSTs

New update: \( \theta_{t+1} = \theta_t + \alpha y_t x_t + \alpha n_t \)

\( n_t \): a "noise" vector that prunes the tree

\[
n_{t,s} = \begin{cases} 
  -y_t x_{t,s} & \text{if } |s| > d_t, \text{ } s \text{ suffix} \\
  0 & \text{otherwise}
\end{cases}
\]

Let \( P_t = \sum_{i \in J_t} \|n_i\|_{\infty} = \sum_{i \in J_t} \beta^{d_i+1} \). \( J_t \) is the set of rounds up to \( t \) in which mistakes were made.
Winnow for PSTs

New update: \( \theta_{t+1} = \theta_t + \alpha y_t x_t + \alpha n_t \)

\( n_t \): a “noise” vector that prunes the tree

\[
n_{t,s} = \begin{cases} 
- y_t x_{t,s} & \text{if } |s| > d_t, \ s \text{ suffix} \\
0 & \text{otherwise}
\end{cases}
\]

Let \( P_t = \sum_{i \in J_t} \| n_i \|_\infty = \sum_{i \in J_t} \beta^{d_i+1} \). \( J_t \) is the set of rounds up to \( t \) in which mistakes were made

- \( P_t \) is the effect of noise in the analysis
Winnow for PSTs

New update: \( \theta_{t+1} = \theta_t + \alpha y_t x_t + \alpha n_t \)

\( n_t \): a "noise" vector that prunes the tree

\[
\begin{align*}
n_{t,s} = \begin{cases} 
  -y_t x_{t,s} & \text{if } |s| > d_t, \ s \text{ suffix} \\
  0 & \text{otherwise}
\end{cases}
\end{align*}
\]

Let \( P_t = \sum_{i \in J_t} \|n_i\|_\infty = \sum_{i \in J_t} \beta^{d_i+1} \). \( J_t \) is the set of rounds up to \( t \) in which mistakes were made

- \( P_t \) is the effect of noise in the analysis

\( d_t \) is set to guarantee \( P_t \leq |J_t|^{2/3} \). Suffices to set

\[
d_t = \left\lfloor \log_\beta \left( \sqrt[3]{P_{t-1}^3} + 2P_{t-1}^{3/2} + 1 - P_{t-1} \right) - 1 \right\rfloor
\]
Mistake Bound

Let there be a tree \( u \left( \| u \|_1 = 1, u_i \geq 0 \right) \) which over the input sequence \( y_1, y_2, \ldots, y_T \) attains loss \( L = \sum_{t=1}^{T} \max(0, \delta - y_t \langle u, x_t \rangle) \), then our algorithm’s mistakes \( M_T \) will be at most

\[
\max \left\{ \frac{2L}{\delta} + \frac{8 \log T}{\delta^2}, \frac{64}{\delta^3} \right\}
\]
Mistake Bound and Growth Bound

Let there be a tree $u$ ($\|u\|_1 = 1$, $u_i \geq 0$) which over the input sequence $y_1, y_2, \ldots, y_T$ attains loss $L = \sum_{t=1}^{T} \max(0, \delta - y_t \langle u, x_t \rangle)$, then our algorithm’s mistakes $M_T$ will be at most

$$\max \left\{ \frac{2L}{\delta}, \frac{8 \log T}{\delta^2}, \frac{64}{\delta^3} \right\}$$

Moreover, by setting $\beta = 2^{-1/3}$, the learned tree will have at most $\log_2(M_T) + 4$ levels.
Proof Sketch

Growth bound is straightforward

$$\Phi(w_t) = \sum_{i} u_i \log u_i w_t, \quad i \geq 0$$

Upper bound $\Phi(w_1)$ and lower bound decrease in potential with each mistake:

$$\Delta \Phi = \text{effect of full update} \geq f(\alpha, \delta, \text{loss of } u) - \text{effect of noise}$$
Proof Sketch

Growth bound is straightforward

Mistake bound via potential function

\[ \Phi(w_t) = \sum_{i} u_i \log u_i w_t, \quad i \geq 0 \]

Upper bound \( \Phi(w_1) \) and lower bound decrease in potential with each mistake:

\[ \Delta \Phi = \text{effect of full update} \geq f(\alpha, \delta, \text{loss of } u) - \text{effect of noise} \]
Proof Sketch

Growth bound is straightforward

Mistake bound via potential function

\[ \Phi(w_t) = \sum_i u_i \log \frac{u_i}{w_{t,i}} \geq 0 \]
Proof Sketch

Growth bound is straightforward

Mistake bound via potential function

\[ \Phi(w_t) = \sum_i u_i \log \frac{u_i}{w_{t,i}} \geq 0 \]

Upper bound \( \Phi(w_1) \) and lower bound decrease in potential with each mistake:

\[ \Delta \Phi = \text{effect of full update} - \text{effect of noise} \geq f(\alpha, \delta, \text{loss of } u) \]
Proof Sketch II

Potential:

\[ \Phi(w_1) \]

Noise \( P_t \):

Example Correct Prediction

\[ x_1 \]

Progress due to classic update

Net progress

Effect of noise
Proof Sketch II

Potential: \( \Phi(w_1) \)

Noise \( P_t \):

Example Correct Prediction

\( x_1 \)  \( \times \)

Progress due to classic update

Net progress  Effect of noise
Proof Sketch II

Potential: $\Phi(w_1)$

Noise $P_t$:

Example Correct Prediction

$x_1$

Progress due to classic update

Net progress

Effect of noise
Proof Sketch II

Potential: \( \Phi(w_1) \)

Noise \( P_t \):

Example Correct Prediction

\[ x_1, x_2 \]

Progress due to classic update

Net progress

Effect of noise

Nikos Karampatziakis, Dexter Kozen

Learning Prediction Suffix Trees
Proof Sketch II

Potential: \[ \Phi(w_1) \]

Noise \( P_t \):

- Example Correct Prediction
  - \( x_1 \)
  - \( x_2 \)

Progress due to classic update

Net progress

Effect of noise

Nikos Karampatziakis, Dexter Kozen  Learning Prediction Suffix Trees
Proof Sketch II

Potential: $\Phi(w_1)$

Noise $P_t$:

Example Correct Prediction

$x_1$ $\times$

$x_2$ $\times$

Progress due to classic update

Net progress

Effect of noise
Proof Sketch II

Potential: $\Phi(w_1)$

Noise $P_t$:

Example Correct Prediction

$x_1$  
$x_2$  
$x_3$

Progress due to classic update

Net progress

Effect of noise
Proof Sketch II

Potential: \[ \Phi(w_1) \]

Noise \( P_t \):

Example Correct Prediction

\[ x_1 \] \[ x_2 \] \[ x_3 \]

Progress due to classic update

Net progress

Effect of noise
Proof Sketch II

Potential: \( \Phi(w_1) \)

Noise \( P_t \):

Example Correct Prediction

\( x_1 \) \( x \)
\( x_2 \) \( x \)
\( x_3 \) \( \checkmark \)
\( x_4 \)

Progress due to classic update

Net progress

Effect of noise
Proof Sketch II

Potential: $\Phi(w_1)$

Noise $P_t$:

Example Correct Prediction

$x_1$ $x$

$x_2$ $x$

$x_3$ $\checkmark$

$x_4$ $x$

Progress due to classic update

Net progress

Effect of noise
Proof Sketch II

Potential: \( \Phi(w_1) \)

Noise \( P_t \):

Example Correct Prediction

\[ x_1 \] \[ \times \]
\[ x_2 \] \[ \times \]
\[ x_3 \] \[ \checkmark \]
\[ x_4 \] \[ \times \]

Progress due to classic update

Net progress

Effect of noise
Proof Sketch II

Potential: \[ \Phi(w_1) \]

Noise \( P_t \):

Example Correct Prediction

\[ x_1 \]
\[ x_2 \]
\[ x_3 \]
\[ x_4 \]

Progress due to classic update

Net progress

Effect of noise
Proof Sketch III

\[ \text{length of } \square \leq \Phi(w_1) \]
length of \[ \leq \Phi(w_1) \]

length of \[ - \text{length of} \quad \leq \Phi(w_1) \]
Proof Sketch III

\[
\text{length of } \leq \Phi(w_1)
\]

\[
\text{length of } - \text{length of } \leq \Phi(w_1)
\]

\[
\geq \min \text{ size } \times \text{mistakes}
\]
Proof Sketch III

\[ \text{length of } \leq \Phi(w_1) \]

\[ \text{length of } \geq \text{min size} \times \text{mistakes} - \text{length of } \leq \text{mistakes}^{2/3} \leq \Phi(w_1) \]
Proof Sketch III

\[
\text{length of } \begin{array}{c}
\end{array} \leq \Phi(w_1)
\]

\[
\underbrace{\text{length of } \begin{array}{c}
\end{array} - \text{length of } \begin{array}{c}
\end{array}}_{\geq \text{min size } \times \text{mistakes}} \leq \Phi(w_1)
\]

\[
\begin{array}{c}
\text{min size } \cdot \text{mistakes} - \text{mistakes}^{2/3} \leq \Phi(w_1)
\end{array}
\]
Results

3 programs, 120 sequences of system calls

<table>
<thead>
<tr>
<th>PST Size</th>
<th>Outlook</th>
<th>Excel</th>
<th>Firefox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceptron</td>
<td>41239</td>
<td>24402</td>
<td>21081</td>
</tr>
<tr>
<td>Winnow</td>
<td>25679</td>
<td>15338</td>
<td>12662</td>
</tr>
</tbody>
</table>

Winnow makes fewer mistakes and grows smaller trees for all 120 sequences.
Results

3 programs, 120 sequences of system calls
Ran our winnow variant and [Dekel et al., 2004]'s self bounded perceptron for PSTs
Results

3 programs, 120 sequences of system calls
Ran our \textbf{winnow} variant and \cite{Dekel:2004}'s self bounded \textbf{perceptron} for PSTs

<table>
<thead>
<tr>
<th>% Error</th>
<th>Outlook</th>
<th>Excel</th>
<th>Firefox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceptron</td>
<td>5.1</td>
<td>22.68</td>
<td>14.86</td>
</tr>
<tr>
<td>Winnow</td>
<td>4.43</td>
<td>20.59</td>
<td>13.88</td>
</tr>
</tbody>
</table>
Results

3 programs, 120 sequences of system calls
Ran our winnow variant and [Dekel et al., 2004]’s self bounded perceptron for PSTs

<table>
<thead>
<tr>
<th></th>
<th>Outlook</th>
<th>Excel</th>
<th>Firefox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceptron</td>
<td>5.1</td>
<td>22.68</td>
<td>14.86</td>
</tr>
<tr>
<td>Winnow</td>
<td>4.43</td>
<td>20.59</td>
<td>13.88</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Outlook</th>
<th>Excel</th>
<th>Firefox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceptron</td>
<td>41239</td>
<td>24402</td>
<td>21081</td>
</tr>
<tr>
<td>Winnow</td>
<td>25679</td>
<td>15338</td>
<td>12662</td>
</tr>
</tbody>
</table>
Results

3 programs, 120 sequences of system calls
Ran our **winnow** variant and [Dekel et al., 2004]'s
self bounded **perceptron** for PSTs

<table>
<thead>
<tr>
<th></th>
<th>Outlook</th>
<th>Excel</th>
<th>Firefox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceptron</td>
<td>5.1</td>
<td>22.68</td>
<td>14.86</td>
</tr>
<tr>
<td>Winnow</td>
<td>4.43</td>
<td>20.59</td>
<td>13.88</td>
</tr>
</tbody>
</table>

Winnow makes fewer mistakes and grows smaller trees for **all** 120 sequences
Related Work

Much work on PSTs [Willems et al., 1995], [Ron et al., 1996], [Pereira & Singer, 1999]... but with assumptions on the tree structure e.g. a priori bounds on the tree’s depth.
Related Work

Much work on PSTs [Willems et al., 1995], [Ron et al., 1996], [Pereira & Singer, 1999]. . . but with assumptions on the tree structure e.g. a priori bounds on the tree’s depth

[Dekele et al., 2004] self bounded perceptron.
Similar ideas, but overfits in practice.
Related Work

Much work on PSTs [Willems et al., 1995], [Ron et al., 1996], [Pereira & Singer, 1999]... but with assumptions on the tree structure e.g. a priori bounds on the tree’s depth

[Dekel et al., 2004] self bounded perceptron. Similar ideas, but overfits in practice.

[Shalev-Shwartz & Tewari, 2009] get sparse solutions from any p-norm algorithm
Summary

Introduced an online learning algorithm to learn PSTs
Summary

Introduced an online learning algorithm to learn PSTs

Competitive with best fixed PST in hindsight
Summary

Introduced an online learning algorithm to learn PSTs

Competitive with best fixed PST in hindsight

The resulting trees grow slowly if necessary
Summary

Introduced an online learning algorithm to learn PSTs

  Competitive with best fixed PST in hindsight
  The resulting trees grow slowly if necessary

On our task, it made fewer mistakes and grew smaller trees than other state-of-the-art algorithms.


Shalev-Shwartz, S., & Tewari, A. (2009). Stochastic Methods for $\ell_1$ Regularized Loss Minimization. *Proceedings of the 26th ICML.*
Differences with [Dekel et al., 2004]

Features: \( \beta = 2^{-1/3} \) vs. \( \beta = 2^{-1/2} \)

\[ P_t: \sum \| n_i \|_\infty \text{ vs. } \sum \| n_i \|_2 \]

Tolerance: \( P_t \leq M_t^{2/3} \) vs. \( P_t \leq \frac{1}{2} \sqrt{M_t} \)
Tweaking [Dekel et al., 2004]

Setting $\beta = 2^{-1/3}$: big trees many mistakes (overfit)

Setting $P_t \leq M_t^{2/3}$: small trees many mistakes (underfit)

Doing both: few mistakes, medium sized trees (less overfit)
More Results

Nikos Karampatziakis, Dexter Kozen

Learning Prediction Suffix Trees 24
More Results

Nikos Karampatziakis, Dexter Kozen

Learning Prediction Suffix Trees
More Results

Nikos Karampatziakis, Dexter Kozen

Learning Prediction Suffix Trees 24