Online Learning of Prediction Suffix Trees

Nikos Karampatziakis Dexter Kozen

Cornell University

April 24, 2009

To appear in ICML 2009

Introduction

Sequential prediction

Introduction

Sequential prediction

Prediction Suffix Trees

Introduction

Sequential prediction

Prediction Suffix Trees

Learning algorithm

Motivation

Monitoring of applications in a computer system

Motivation

Monitoring of applications in a computer system

System calls

Motivation

Monitoring of applications in a computer system

System calls

Prediction model

Motivation

Monitoring of applications in a computer system

System calls

Prediction model

Some assumptions

Outline

Prediction Suffix Trees

Algorithm and Properties

Results

Outline

Prediction Suffix Trees

Algorithm and Properties

Results

What is the next item in the sequence?

open(),read(),read(),...
$G, A, T, T, A, C, \ldots$
$+1, -1, +1, -1, +1, \ldots$
all,your,base,...

What is the next item in the sequence?

open(),read(),read(),...
$G, A, T, T, A, C, \ldots$
$+1, -1, +1, -1, +1, \ldots$
all,your,base,...

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

What is the next item in the sequence?

open(),read(),read(),...
$G, A, T, T, A, C, \ldots$
$+1, -1, +1, -1, +1, \ldots$
all,your,base,...

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

$y_t = h(y_1^{t-1}) \qquad y_i^j := y_i, \ldots, y_j$

What is the next item in the sequence?

open(),read(),read(),...
$G, A, T, T, A, C, \ldots$
$+1, -1, +1, -1, +1, \ldots$
all,your,base,...

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

$y_t = h(y_1^{t-1}) \qquad y_i^j := y_i, \ldots, y_j$

More general problem: $p(y_t | y_1^{t-1})$

What is the next item in the sequence?

open(),read(),read(),. . .
$G, A, T, T, A, C, \ldots$
$+1, -1, +1, -1, +1, \ldots$
all,your,base,. . .

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

$y_t = h(y_1^{t-1}) \qquad y_i^j := y_i, \ldots, y_j$

More general problem: $p(y_t | y_1^{t-1})$

Markovian Assumption

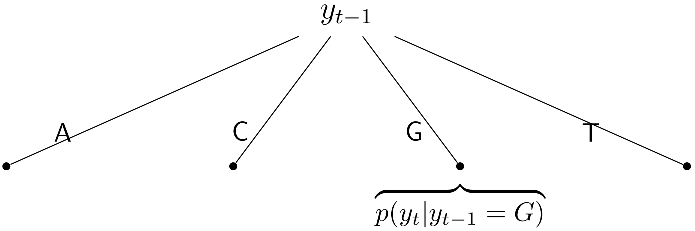
Markov Models

Zero order: $p(y_t|y_1^{t-1}) = p(y_t)$

Markov Models

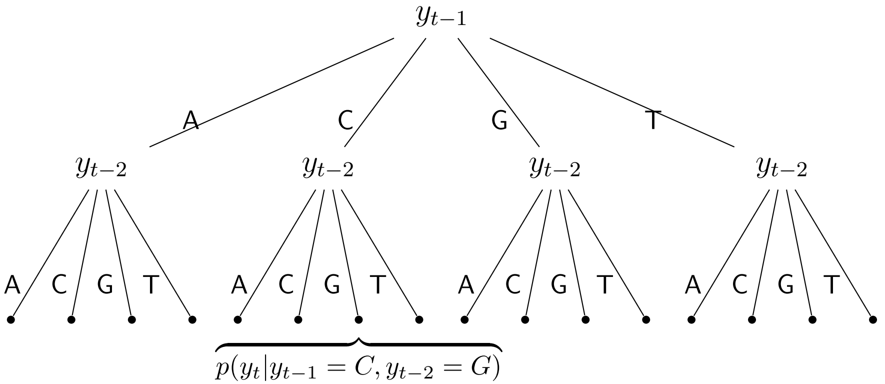
Zero order: $p(y_t|y_1^{t-1}) = p(y_t)$

First order:



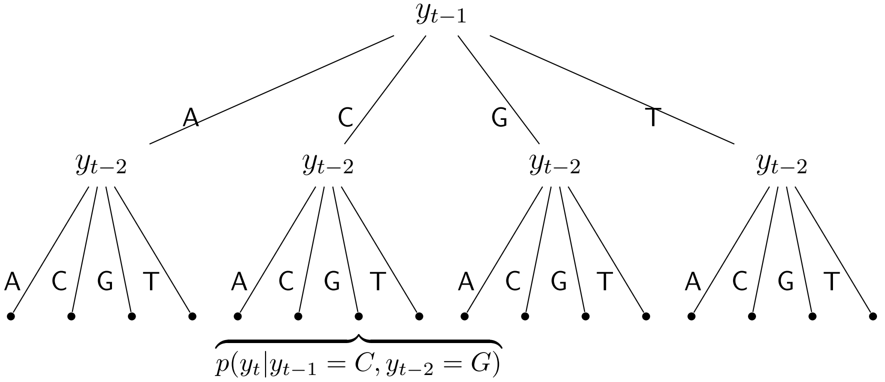
Markov Models

Second order:



Markov Models

Second order:



Third order, k-th order etc.

Markov Models — Problems

How to select the best k?

Markov Models — Problems

How to select the best k?

Number of model parameters exponential in k.

Markov Models — Problems

How to select the best k?

Number of model parameters exponential in k.

Poor family of models. What if I want a model with 100 parameters?

Markov Models — Problems

How to select the best k?

Number of model parameters exponential in k.

Poor family of models. What if I want a model with 100 parameters?

What if y_t does not depend on y_{t-2} if $y_{t-1} = A$ but it depends on it if $y_{t-1} = T$?

Markov Models — Problems

How to select the best k?

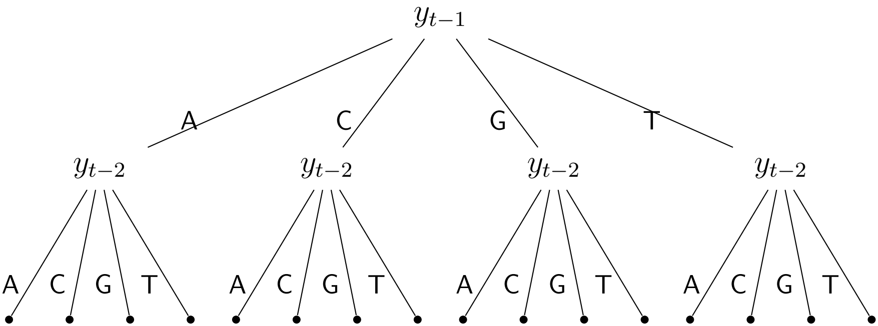
Number of model parameters exponential in k.

Poor family of models. What if I want a model with 100 parameters?

What if y_t does not depend on y_{t-2} if $y_{t-1} = A$ but it depends on it if $y_{t-1} = T$?

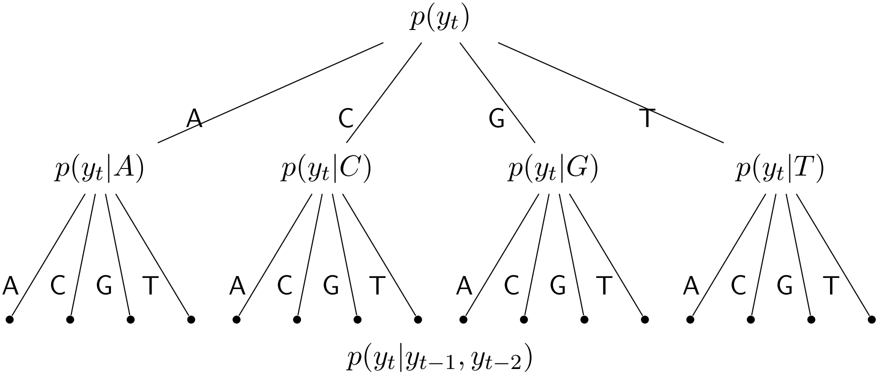
Prediction Suffix Trees address these problems.
[Willems et al., 1995, Ron et al., 1996, Helmbold & Schapire, 1997, Pereira & Singer, 1999]

Prediction Suffix Trees



Prediction Suffix Trees

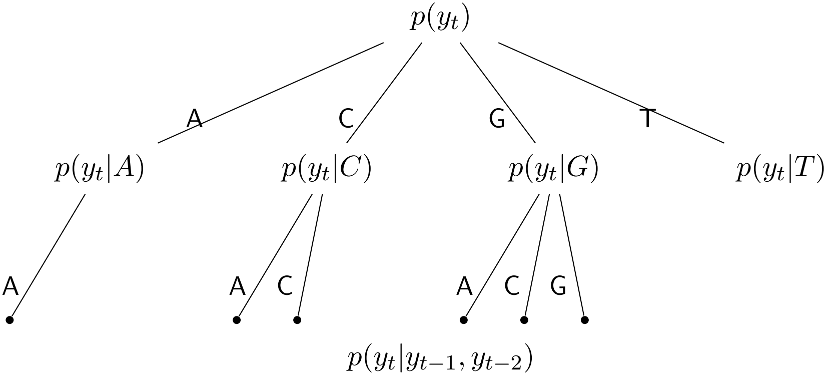
Keep a model for each k. Use a weighted sum.



Prediction Suffix Trees

Keep a model for each k. Use a weighted sum.

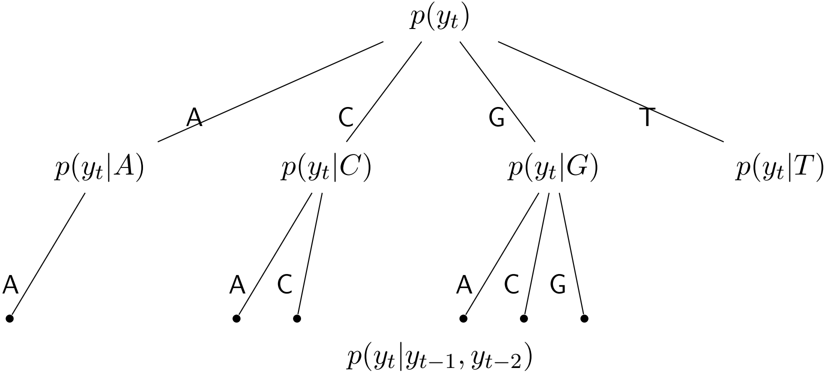
Prune useless branches and subtrees.



Prediction Suffix Trees

Keep a model for each k. Use a weighted sum.

Prune useless branches and subtrees.



Assumptions: $p(y_t|y_{t-1} = T, y_{t-2}) = p(y_t|y_{t-1} = T)$,
$p(y_t|y_{t-1} = A, y_{t-2} = \neg A) = p(y_t|y_{t-1} = A)$.

Moving Away from Probabilistic Modeling

We just want to predict y_t from y_1^{t-1}

Moving Away from Probabilistic Modeling

We just want to predict y_t from y_1^{t-1}

Assume $y_t \in \{-1, +1\}$

Moving Away from Probabilistic Modeling

We just want to predict y_t from y_1^{t-1}

Assume $y_t \in \{-1, +1\}$

Store a weight instead of a probability

Moving Away from Probabilistic Modeling

We just want to predict y_t from y_1^{t-1}

Assume $y_t \in \{-1, +1\}$

Store a weight instead of a probability

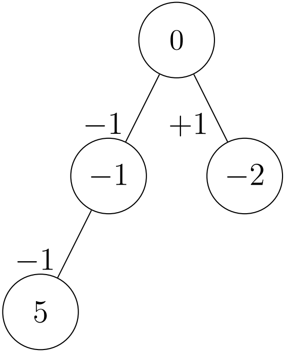
Use the sign of a weighted sum of the values in the appropriate path

Example

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

Discounting: A node at depth d is discounted by 2^{-d}

Tree:



Input Sequence:

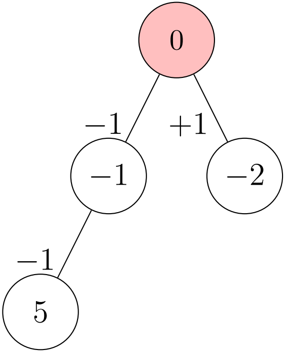
Decision:

Example

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

Discounting: A node at depth d is discounted by 2^{-d}

Tree:



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

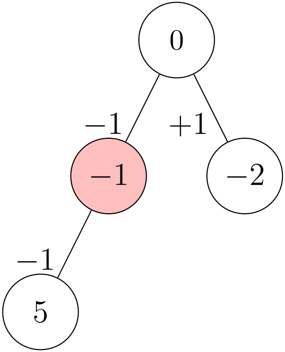
Decision: 0

Example

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

Discounting: A node at depth d is discounted by 2^{-d}

Tree:



Input Sequence: $\ldots, -1, \mathbf{-1}$

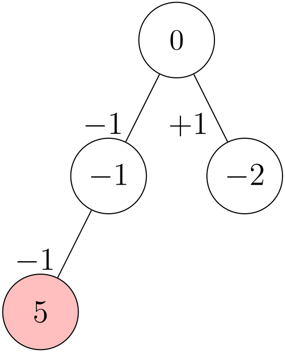
Decision: $0 - \frac{1}{2} \cdot 1$

Example

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

Discounting: A node at depth d is discounted by 2^{-d}

Tree:



Input Sequence: $\ldots, \mathbf{-1}, -1$

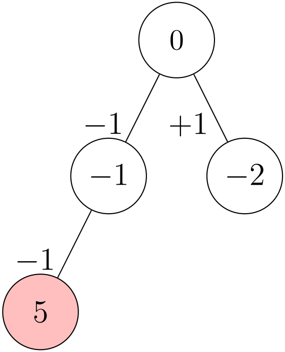
Decision: $0 - \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 5$

Example

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

Discounting: A node at depth d is discounted by 2^{-d}

Tree:



Input Sequence: $\ldots, \mathbf{-1}, -1$

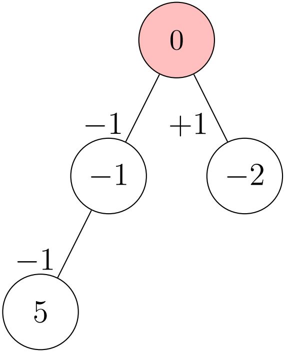
Decision: $0 - \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 5 = \frac{3}{4} \xrightarrow{sign} +1$

Example

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

Discounting: A node at depth d is discounted by 2^{-d}

Tree:



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

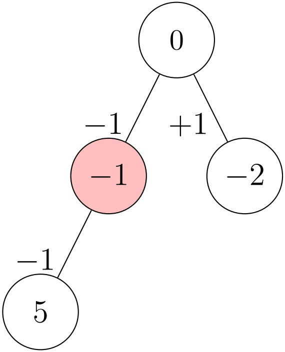
Decision: 0

Example

True Sequence: $-1, -1, +1, -1, -1, +1, \dots$

Discounting: A node at depth d is discounted by 2^{-d}

Tree:



Input Sequence: $\dots, +1, \mathbf{-1}$

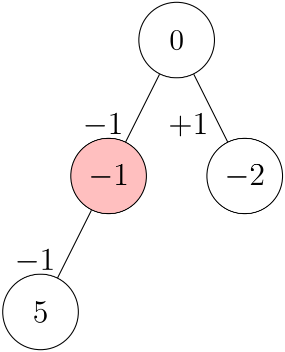
Decision: $0 - \frac{1}{2} \cdot 1$

Example

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

Discounting: A node at depth d is discounted by 2^{-d}

Tree:



Input Sequence: $\ldots, +1, \mathbf{-1}$

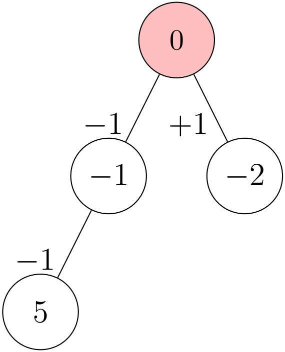
Decision: $0 - \frac{1}{2} \cdot 1 = -\frac{1}{2} \stackrel{sign}{\longrightarrow} -1$

Example

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

Discounting: A node at depth d is discounted by 2^{-d}

Tree:



Input Sequence: $\ldots, +1$

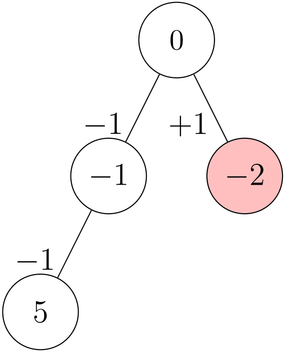
Decision: 0

Example

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

Discounting: A node at depth d is discounted by 2^{-d}

Tree:



Input Sequence: $\ldots, \mathbf{+1}$

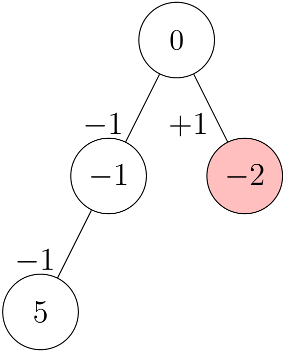
Decision: $0 - \frac{1}{2} \cdot 2$

Example

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

Discounting: A node at depth d is discounted by 2^{-d}

Tree:



Input Sequence: $\ldots, +\mathbf{1}$

Decision: $0 - \frac{1}{2} \cdot 2 = -1 \overset{sign}{\longrightarrow} -1$

Remarks

Notation for the node values: $g_{t,s}$

Remarks

Notation for the node values: $g_{t,s}$

Discounting scheme in this work:

$$x^+_{t,s} = \begin{cases} (1-\epsilon)^{|s|} & \text{if} \quad s = y^{t-1}_{t-i} \qquad i = 1, \ldots, t-1 \\ 0 & \text{otherwise} \end{cases}$$

The best value of ϵ will be determined (much) later.

Remarks

Notation for the node values: $g_{t,s}$

Discounting scheme in this work:

$$x_{t,s}^{+} = \begin{cases} (1-\epsilon)^{|s|} & \text{if} \quad s = y_{t-i}^{t-1} \qquad i = 1, \ldots, t-1 \\ 0 & \text{otherwise} \end{cases}$$

The best value of ϵ will be determined (much) later.

Decision at time t: $\operatorname{sign}\left(\sum_s g_{t,s} x_{t,s}^{+}\right) = \operatorname{sign}(\langle g_t, x_t^{+} \rangle)$

Outline

Prediction Suffix Trees

Algorithm and Properties

Results

Outline

Prediction Suffix Trees

Algorithm and Properties

Results

Algorithms for Linear Prediction

SVMs, logistic regression,. . .

Algorithms for Linear Prediction

SVMs, logistic regression,. . .

Focus on online setting

Algorithms for Linear Prediction

SVMs, logistic regression,. . .

Focus on online setting

Algorithm maintains hypothesis w_t. In each round:

Algorithms for Linear Prediction

SVMs, logistic regression,...

Focus on online setting

Algorithm maintains hypothesis w_t. In each round:

The algorithm receives information x_t.

Algorithms for Linear Prediction

SVMs, logistic regression,...

Focus on online setting

Algorithm maintains hypothesis w_t. In each round:

The algorithm receives information x_t.
It outputs $\hat{y}_t = \mathrm{sign}(\langle w_t, x_t \rangle)$.

Algorithms for Linear Prediction

SVMs, logistic regression,...

Focus on online setting

Algorithm maintains hypothesis w_t. In each round:

The algorithm receives information x_t.
It outputs $\hat{y}_t = \mathrm{sign}(\langle w_t, x_t \rangle)$.
It receives the correct output y_t.

Algorithms for Linear Prediction

SVMs, logistic regression,...

Focus on online setting

Algorithm maintains hypothesis w_t. In each round:

The algorithm receives information x_t.
It outputs $\hat{y}_t = \mathrm{sign}(\langle w_t, x_t \rangle)$.
It receives the correct output y_t.
It updates: $w_{t+1} \leftarrow f(w_t)$.

Algorithms for Linear Prediction

SVMs, logistic regression,. . .

Focus on online setting

Algorithm maintains hypothesis w_t. In each round:

The algorithm receives information x_t.
It outputs $\hat{y}_t = \mathrm{sign}(\langle w_t, x_t \rangle)$.
It receives the correct output y_t.
It updates: $w_{t+1} \leftarrow f(w_t)$.

A mistake is made when $y_t \langle w_t, x_t \rangle \leq 0$.

Perceptron

If a mistake is made at round t: $w_{t+1} = w_t + \alpha y_t x_t$

Perceptron

If a mistake is made at round t: $w_{t+1} = w_t + \alpha y_t x_t$

α is a learning rate.

Perceptron

If a mistake is made at round t: $w_{t+1} = w_t + \alpha y_t x_t$

α is a learning rate.

It is selected so that it optimizes various tradeoffs.

Balanced Winnow

Balanced Winnow [Littlestone, 1989]	Perceptron [Rosenblatt, 1958]
$\theta_1 \leftarrow 0$	$\theta_1 \leftarrow 0$
for $t = 1, 2, \ldots, T$ **do**	**for** $t = 1, 2, \ldots, T$ **do**
$\quad w_{t,i} \leftarrow \frac{e^{\theta_{t,i}}}{\sum_j e^{\theta_{t,j}}}$	$\quad w_t \leftarrow \theta_t$
$\quad \hat{y}_t \leftarrow \langle w_t, x_t \rangle$	$\quad \hat{y}_t \leftarrow \langle w_t, x_t \rangle$
$\quad$ **if** $y_t \hat{y}_t \leq 0$	$\quad$ **if** $y_t \hat{y}_t \leq 0$
$\quad\quad \theta_{t+1} \leftarrow \theta_t + \alpha y_t x_t$	$\quad\quad \theta_{t+1} \leftarrow \theta_t + \alpha y_t x_t$
$\quad$ **else**	$\quad$ **else**
$\quad\quad \theta_{t+1} \leftarrow \theta_t$	$\quad\quad \theta_{t+1} \leftarrow \theta_t$

Balanced Winnow

Balanced Winnow [Littlestone, 1989]	Perceptron [Rosenblatt, 1958]
$\theta_1 \leftarrow 0$	$\theta_1 \leftarrow 0$
for $t = 1, 2, \ldots, T$ **do**	**for** $t = 1, 2, \ldots, T$ **do**
$w_{t,i} \leftarrow \frac{e^{\theta_{t,i}}}{\sum_j e^{\theta_{t,j}}}$	$w_t \leftarrow \theta_t$
$\hat{y}_t \leftarrow \langle w_t, x_t \rangle$	$\hat{y}_t \leftarrow \langle w_t, x_t \rangle$
if $y_t \hat{y}_t \leq 0$	**if** $y_t \hat{y}_t \leq 0$
$\theta_{t+1} \leftarrow \theta_t + \alpha y_t x_t$	$\theta_{t+1} \leftarrow \theta_t + \alpha y_t x_t$
else	**else**
$\theta_{t+1} \leftarrow \theta_t$	$\theta_{t+1} \leftarrow \theta_t$

Important for Balanced Winnow:

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

$$x_{t,s}^+ = \begin{cases} (1-\epsilon)^{|s|} & \text{if} \quad s = y_{t-i}^{t-1} \qquad i = 1, \ldots, t-1 \\ 0 & \text{otherwise} \end{cases}$$

Advantages

Multiplicative updates — fast convergence.

Advantages

Multiplicative updates — fast convergence.

Can cope with many features when few of them are relevant.

Advantages

Multiplicative updates — fast convergence.

Can cope with many features when few of them are relevant.

For our application it can track changes better.

Where's the catch?

Perceptron and Winnow don't care about the sparsity of their hypothesis.

Where's the catch?

Perceptron and Winnow don't care about the sparsity of their hypothesis.

Eventually, we have a feature for every substring of y_1^T.

Where's the catch?

Perceptron and Winnow don't care about the sparsity of their hypothesis.

Eventually, we have a feature for every substring of y_1^T.

Implications:

Need to learn $O(T^2)$ weights in T rounds ☹

Where's the catch?

Perceptron and Winnow don't care about the sparsity of their hypothesis.

Eventually, we have a feature for every substring of y_1^T.

Implications:

Need to learn $O(T^2)$ weights in T rounds ☹

Need to store $O(T^2)$ numbers ☹

Naively, all weights affect the decision and must be stored.

Does the decision depend on every feature?

Recall the specific form of features: $x_t = [x_t^+, x_t^-] = [x_t^+, -x_t^+]$.

Does the decision depend on every feature?

Recall the specific form of features: $x_t = [x_t^+, x_t^-] = [x_t^+, -x_t^+]$.

Notation: $w_t = [w_t^+, w_t^-] \quad \theta_t = [\theta_t^+, \theta_t^-]$. Recall $w_{t,i} = \dfrac{e^{\theta_{t,i}}}{\sum_j e^{\theta_{t,j}}}$

Does the decision depend on every feature?

Recall the specific form of features: $x_t = [x_t^+, x_t^-] = [x_t^+, -x_t^+]$.

Notation: $w_t = [w_t^+, w_t^-] \quad \theta_t = [\theta_t^+, \theta_t^-]$. Recall $w_{t,i} = \dfrac{e^{\theta_{t,i}}}{\sum_j e^{\theta_{t,j}}}$

Easy inductive argument can show that $\theta_t^- = -\theta_t^+$

Does the decision depend on every feature?

Recall the specific form of features: $x_t = [x_t^+, x_t^-] = [x_t^+, -x_t^+]$.

Notation: $w_t = [w_t^+, w_t^-] \quad \theta_t = [\theta_t^+, \theta_t^-]$. Recall $w_{t,i} = \frac{e^{\theta_{t,i}}}{\sum_j e^{\theta_{t,j}}}$

Easy inductive argument can show that $\theta_t^- = -\theta_t^+$

Decision $\hat{y}_t = \langle w_t, x_t \rangle = \langle w_t^+ - w_t^-, x_t^+ \rangle =$[1]

$$= \sum_{i=1}^{d} \frac{\sinh(\theta_{t,i}^+)}{\sum_{j=1}^{d} \cosh(\theta_{t,j}^+)} x_{t,i}^+ \propto \sum_{i=1}^{d} \sinh(\theta_{t,i}^+) x_{t,i}^+$$

[1] $\sinh(x) = \frac{e^x - e^{-x}}{2}$ and $\cosh(x) = \frac{e^x + e^{-x}}{2}$

Does the decision depend on every feature?

Recall the specific form of features: $x_t = [x_t^+, x_t^-] = [x_t^+, -x_t^+]$.

Notation: $w_t = [w_t^+, w_t^-] \quad \theta_t = [\theta_t^+, \theta_t^-]$. Recall $w_{t,i} = \frac{e^{\theta_{t,i}}}{\sum_j e^{\theta_{t,j}}}$

Easy inductive argument can show that $\theta_t^- = -\theta_t^+$

Decision $\hat{y}_t = \langle w_t, x_t \rangle = \langle w_t^+ - w_t^-, x_t^+ \rangle =$[1]

$$= \sum_{i=1}^{d} \frac{\sinh(\theta_{t,i}^+)}{\sum_{j=1}^{d} \cosh(\theta_{t,j}^+)} x_{t,i}^+ \propto \sum_{i=1}^{d} \sinh(\theta_{t,i}^+) x_{t,i}^+$$

Iff $\theta_{t,i}^+ = 0$ the decision does not depend on feature i.

[1] $\sinh(x) = \frac{e^x - e^{-x}}{2}$ and $\cosh(x) = \frac{e^x + e^{-x}}{2}$

Observations

If $\theta_{t,s} = 0$ we don't have to store it.

Observations

If $\theta_{t,s} = 0$ we don't have to store it.

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

Observations

If $\theta_{t,s} = 0$ we don't have to store it.

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

As mistakes are made, the tree grows.

Observations

If $\theta_{t,s} = 0$ we don't have to store it.

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

As mistakes are made, the tree grows.

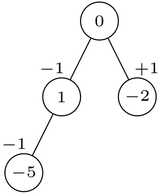
Classic Winnow/Perceptron update quickly destroys sparsity.

Illustration

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

$$x^+_{t,s} = \begin{cases} 2^{-|s|} & \text{if} \quad s = y^{t-1}_{t-i} \qquad i = 1, \ldots, t-1 \\ 0 & \text{otherwise} \end{cases}$$

Tree:



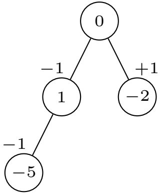
Decision:

Illustration

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

$$x^{+}_{t,s} = \begin{cases} 2^{-|s|} & \text{if} \quad s = y_{t-i}^{t-1} \qquad i = 1, \ldots, t-1 \\ 0 & \text{otherwise} \end{cases}$$

Tree:



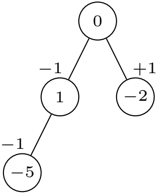
Decision: $\frac{1}{2} \cdot \sinh(1)$

Illustration

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

$$x^+_{t,s} = \begin{cases} 2^{-|s|} & \text{if} \quad s = y_{t-i}^{t-1} \qquad i = 1, \ldots, t-1 \\ 0 & \text{otherwise} \end{cases}$$

Tree:



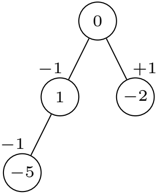
Decision: $\frac{1}{2} \cdot \sinh(1) > 0$

Illustration

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

$$x^+_{t,s} = \begin{cases} 2^{-|s|} & \text{if} \quad s = y^{t-1}_{t-i} \qquad i = 1, \ldots, t-1 \\ 0 & \text{otherwise} \end{cases}$$

Tree:



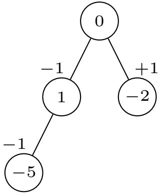
Decision: $\frac{1}{2} \cdot \sinh(1) > 0 \xrightarrow{sign} +1$

Illustration

Input Sequence: $\ldots, -1, +1, -1, +1, -1, \color{red}{-1}$

$$x^+_{t,s} = \begin{cases} 2^{-|s|} & \text{if} \quad s = y^{t-1}_{t-i} \qquad i = 1, \ldots, t-1 \\ 0 & \text{otherwise} \end{cases}$$

Tree:



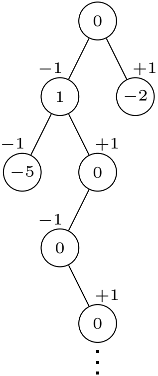
Decision: $\frac{1}{2} \cdot \sinh(1) > 0 \xrightarrow{sign} +1$

Illustration

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

$$x^+_{t,s} = \begin{cases} 2^{-|s|} & \text{if} \quad s = y^{t-1}_{t-i} \qquad i = 1, \ldots, t-1 \\ 0 & \text{otherwise} \end{cases}$$

Tree:



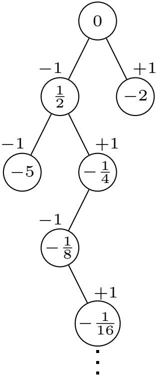
Decision: $\frac{1}{2} \cdot \sinh(1) > 0 \stackrel{sign}{\rightarrow} +1$

Illustration

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

$$x^+_{t,s} = \begin{cases} 2^{-|s|} & \text{if} \quad s = y_{t-i}^{t-1} \qquad i = 1, \ldots, t-1 \\ 0 & \text{otherwise} \end{cases}$$

Tree:



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

Summary of the Problem

Mistake at time t: $O(t)$ nodes are inserted in the tree.

Summary of the Problem

Mistake at time t: $O(t)$ nodes are inserted in the tree.

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

Summary of the Problem

Mistake at time t: $O(t)$ nodes are inserted in the tree.

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

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

Summary of the Problem

Mistake at time t: $O(t)$ nodes are inserted in the tree.

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

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

Need to learn a good θ while keeping it sparse.

Summary of the Problem

Mistake at time t: $O(t)$ nodes are inserted in the tree.

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

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

Need to learn a good θ while keeping it sparse.

Not all sparse vectors are equal.

Better Update Rule

Adaptive bound d_t on the length of the suffixes.

Better Update Rule

Adaptive bound d_t on the length of the suffixes.

Same as the depth up to which the tree will grow on round t.

Better Update Rule

Adaptive bound d_t on the length of the suffixes.

Same as the depth up to which the tree will grow on round t.

d_t will be growing slowly if necessary.

Better Update Rule

Adaptive bound d_t on the length of the suffixes.

Same as the depth up to which the tree will grow on round t.

d_t will be growing slowly if necessary.

New update:

$$\theta_{t+1,s} = \begin{cases} \theta_{t,s} + \alpha y_t x_{t,s} & \text{if } |s| \leq d_t \\ \theta_{t,s} & \text{otherwise} \end{cases}$$

Better Update Rule

Adaptive bound d_t on the length of the suffixes.

Same as the depth up to which the tree will grow on round t.

d_t will be growing slowly if necessary.

New update:

$$\theta_{t+1,s} = \begin{cases} \theta_{t,s} + \alpha y_t x_{t,s} & \text{if } |s| \leq d_t \\ \theta_{t,s} & \text{otherwise} \end{cases}$$

Equivalently:

$$\theta_{t+1} = \theta_t + \alpha y_t x_t + \alpha n_t$$

where n_t is a noise vector that cancels part of the update:

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

Setting d_t

$h_t =$ the length of the path from the root using $y_{t-1}, y_{t-2}, \ldots$.

Setting d_t

$h_t =$ the length of the path from the root using $y_{t-1}, y_{t-2}, \ldots$.

J_t is the subset of rounds $1, \ldots, t$ in which a mistake was made.

Setting d_t

$h_t =$ the length of the path from the root using $y_{t-1}, y_{t-2}, \ldots$.

J_t is the subset of rounds $1, \ldots, t$ in which a mistake was made.

$P_t = \sum_{i \in J_t} ||n_i||_\infty = \sum_{i \in J_t} (1-\epsilon)^{(d_i+1)}$.

Setting d_t

$h_t =$ the length of the path from the root using $y_{t-1}, y_{t-2}, \ldots$.

J_t is the subset of rounds $1, \ldots, t$ in which a mistake was made.

$P_t = \sum_{i \in J_t} ||n_i||_\infty = \sum_{i \in J_t} (1-\epsilon)^{(d_i+1)}$.

d_t will be the smallest integer such that $P_t \leq |J_t|^{2/3}$.

Setting d_t

$h_t =$ the length of the path from the root using $y_{t-1}, y_{t-2}, \ldots$.

J_t is the subset of rounds $1, \ldots, t$ in which a mistake was made.

$P_t = \sum_{i \in J_t} ||n_i||_\infty = \sum_{i \in J_t} (1-\epsilon)^{(d_i+1)}$.

d_t will be the smallest integer such that $P_t \leq |J_t|^{2/3}$.

To guarantee that we can set

$$d_t = \max\left\{h_t, \left\lceil \log_{1-\epsilon}\left(\sqrt[3]{P_{t-1}^3 + 2P_{t-1}^{3/2} + 1} - P_{t-1}\right) - 1\right\rceil\right\}$$

Algorithm Properties — Learning

Mistake Bound

If there exists a tree which over the input sequence $y_1, y_2, \ldots, y_T$ correctly predicts all items with confidence $\geq \delta$, our algorithm's mistakes will be at most

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

Algorithm Properties — Learning

Mistake Bound

If there exists a tree which over the input sequence $y_1, y_2, \ldots, y_T$ correctly predicts all items with confidence $\geq \delta$, our algorithm's mistakes will be at most

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

Instead of assuming a perfect tree, assume a tree u that attains a cumulative δ-hinge loss $L > 0$. The mistake bound becomes:

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

Algorithm Properties — Learning

Mistake Bound

If there exists a tree which over the input sequence $y_1, y_2, \ldots, y_T$ correctly predicts all items with confidence $\geq \delta$, our algorithm's mistakes will be at most

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

Instead of assuming a perfect tree, assume a tree u that attains a cumulative δ-hinge loss $L > 0$. The mistake bound becomes:

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

$L = \sum_{t=1}^{T} \ell_t(u)$ where $\ell_t(u) = \max(0, \delta - y_t\langle u, x_t\rangle)$

Algorithm Properties — Resources

Let M_t be the number of mistakes up to round t

Algorithm Properties — Resources

Let M_t be the number of mistakes up to round t

Set ϵ so that

$$x_{t,s}^{+} = \begin{cases} 2^{-|s|/3} & \text{if} \quad s = y_{t-i}^{t-1} \qquad i = 1, \ldots, t-1 \\ 0 & \text{otherwise} \end{cases}$$

Algorithm Properties — Resources

Let M_t be the number of mistakes up to round t

Set ϵ so that

$$x^+_{t,s} = \begin{cases} 2^{-|s|/3} & \text{if} \quad s = y^{t-1}_{t-i} \qquad i = 1, \ldots, t-1 \\ 0 & \text{otherwise} \end{cases}$$

Growth Bound

Our algorithm will not grow a tree deeper than $\log_2 M_{T-1} + 4$

How to go about proving the properties

Bounding how fast the tree grows is straightforward.

How to go about proving the properties

Bounding how fast the tree grows is straightforward.

Mistake bound: show that each update contributes towards a goal.

How to go about proving the properties

Bounding how fast the tree grows is straightforward.

Mistake bound: show that each update contributes towards a goal.

We need a goal and measure of progress.

Measuring Progress

Goal: A fixed tree with good performance.

Measuring Progress

Goal: A fixed tree with good performance.

For example, a vector u such that $y_t\langle u, x_t\rangle \geq \delta$.

Measuring Progress

Goal: A fixed tree with good performance.

For example, a vector u such that $y_t\langle u, x_t\rangle \geq \delta$.

Our adaptive tree is represented by w_t. Remember $\sum_i w_{t,i} = 1$.

Measuring Progress

Goal: A fixed tree with good performance.

For example, a vector u such that $y_t \langle u, x_t \rangle \geq \delta$.

Our adaptive tree is represented by w_t. Remember $\sum_i w_{t,i} = 1$.

To be fair assume $u_i \geq 0$ and $\sum_i u_i = 1$

Measuring Progress

Goal: A fixed tree with good performance.

For example, a vector u such that $y_t\langle u, x_t\rangle \geq \delta$.

Our adaptive tree is represented by w_t. Remember $\sum_i w_{t,i} = 1$.

To be fair assume $u_i \geq 0$ and $\sum_i u_i = 1$

Measure of progress: Relative entropy between u and w_t

$$D(u||w_t) = \sum_i u_i \log \frac{u_i}{w_{t,i}}$$

Measuring Progress

Goal: A fixed tree with good performance.

For example, a vector u such that $y_t\langle u, x_t\rangle \geq \delta$.

Our adaptive tree is represented by w_t. Remember $\sum_i w_{t,i} = 1$.

To be fair assume $u_i \geq 0$ and $\sum_i u_i = 1$

Measure of progress: Relative entropy between u and w_t

$$D(u||w_t) = \sum_i u_i \log \frac{u_i}{w_{t,i}}$$

Potential function: $\Phi(w_t) = D(u||w_t) \geq 0$

Proof Technique

Upper bound the initial potential.

Proof Technique

Upper bound the initial potential.

Lower bound the change in the potential with each update.

Proof Technique

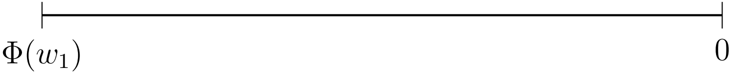
Upper bound the initial potential.

Lower bound the change in the potential with each update.

Keep the total effect of noise bounded. [Dekel et al., 2004]

Pictorially

Potential:



Noise P_t :

Example	Correct Prediction
x_1	

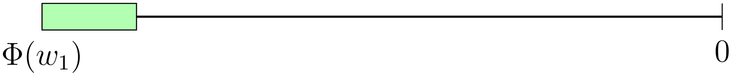
Progress due to classic update

Effect of noise

Net progress

Pictorially

Potential:



Noise P_t :

Example	Correct Prediction
x_1	✗

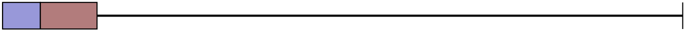
Progress due to classic update

Effect of noise

Net progress

Pictorially

Potential:



$\Phi(w_1)$ 0

Noise P_t :

Example	Correct Prediction
x_1	✗

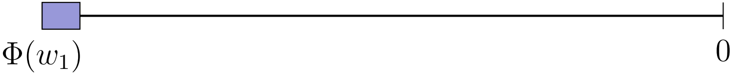
Progress due to classic update

Effect of noise

Net progress

Pictorially

Potential:



Noise P_t :

Example	Correct Prediction
x_1	✗
x_2	

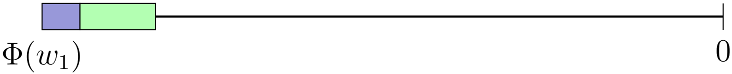
Progress due to classic update

Effect of noise

Net progress

Pictorially

Potential:



Noise P_t :

Example	Correct Prediction
x_1	✗
x_2	✗

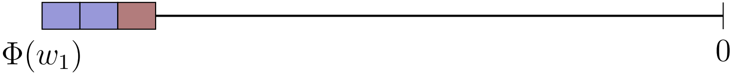
Progress due to classic update

Effect of noise

Net progress

Pictorially

Potential:



Noise P_t :

Example	Correct Prediction
x_1	✗
x_2	✗

Progress due to classic update

Effect of noise

Net progress

Pictorially

Potential:

Noise P_t :

Example	Correct Prediction
x_1	✗
x_2	✗
x_3	

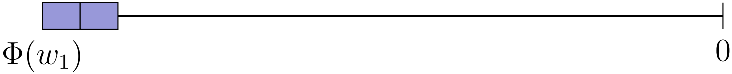
Progress due to classic update

Effect of noise

Net progress

Pictorially

Potential:



Noise P_t :

Example	Correct Prediction
x_1	✗
x_2	✗
x_3	✓

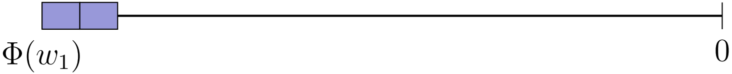
Progress due to classic update

Effect of noise

Net progress

Pictorially

Potential:



Noise P_t :

Example	Correct Prediction
x_1	✗
x_2	✗
x_3	✓
x_4	

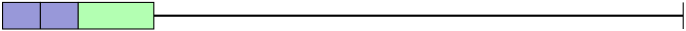
Progress due to classic update

Effect of noise

Net progress

Pictorially

Potential:



$\Phi(w_1)$ $\qquad$ 0

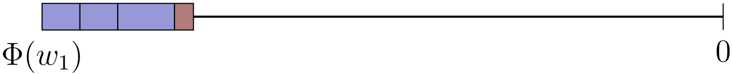
Noise P_t :

Example	Correct Prediction
x_1	✗
x_2	✗
x_3	✓
x_4	✗

- Progress due to classic update
- Effect of noise
- Net progress

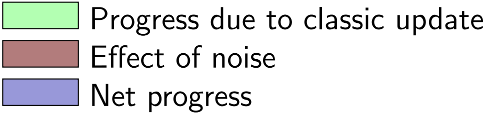
Pictorially

Potential:



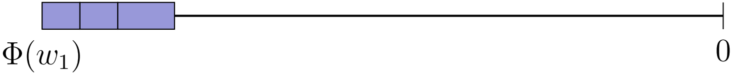
Noise P_t :

Example	Correct Prediction
x_1	✗
x_2	✗
x_3	✓
x_4	✗



Pictorially

Potential:



Noise P_t :

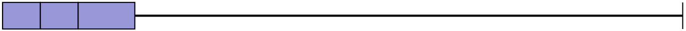
Example	Correct Prediction
x_1	✗
x_2	✗
x_3	✓
x_4	✗
x_5	

Progress due to classic update

Effect of noise

Net progress

Pictorially

Potential: 

$\Phi(w_1)$ 0

Noise P_t : 

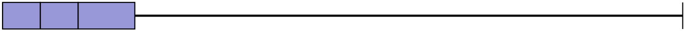
Example	Correct Prediction
x_1	✗
x_2	✗
x_3	✓
x_4	✗
x_5	✓

Progress due to classic update

Effect of noise

Net progress

Pictorially

Potential: 

$\Phi(w_1)$ 0

Noise P_t : 

Example	Correct Prediction
x_1	✗
x_2	✗
x_3	✓
x_4	✗
x_5	✓
x_6	

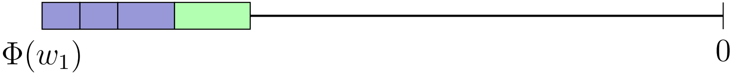
Progress due to classic update

Effect of noise

Net progress

Pictorially

Potential:



Noise P_t :

Example	Correct Prediction
x_1	✗
x_2	✗
x_3	✓
x_4	✗
x_5	✓
x_6	✗

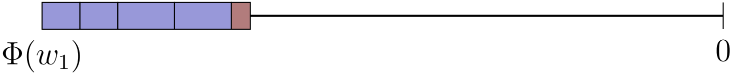
Progress due to classic update

Effect of noise

Net progress

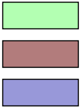
Pictorially

Potential:



Noise P_t :

Example	Correct Prediction
x_1	✗
x_2	✗
x_3	✓
x_4	✗
x_5	✓
x_6	✗

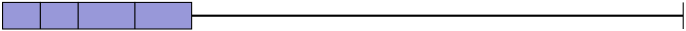


Progress due to classic update

Effect of noise

Net progress

Pictorially

Potential: 

$\Phi(w_1)$ 0

Noise P_t : 

Example	Correct Prediction
x_1	✗
x_2	✗
x_3	✓
x_4	✗
x_5	✓
x_6	✗

Progress due to classic update

Effect of noise

Net progress

Proof

length of $\boxed{} \leq \Phi(w_1)$

Proof

length of  $\leq \Phi(w_1)$

length of  $-$length of  $\leq \Phi(w_1)$

Proof

$$\text{length of } \boxed{} \leq \Phi(w_1)$$

$$\underbrace{\text{length of } \boxed{}}_{\geq \min \boxed{} \text{ size} \times \text{mistakes}} - \text{length of } \boxed{} \leq \Phi(w_1)$$

Proof

$$\text{length of } \boxed{} \leq \Phi(w_1)$$

$$\underbrace{\text{length of } \boxed{}}_{\geq \min \boxed{}\ \text{size} \times \text{mistakes}} - \underbrace{\text{length of } \boxed{}}_{\leq \text{mistakes}^{2/3}} \leq \Phi(w_1)$$

Proof

$$\text{length of } \leq \Phi(w_1)$$

$$\underbrace{\text{length of }}_{\geq \text{min } \text{size} \times \text{mistakes}} - \underbrace{\text{length of }}_{\leq \text{mistakes}^{2/3}} \leq \Phi(w_1)$$

$$\text{min } \text{size} \cdot \text{mistakes} - \text{mistakes}^{2/3} \leq \Phi(w_1)$$

Multiclass extension

We maintain weights $w^{(1)}, w^{(2)}, \dots, w^{(k)}$

Predict $\hat{y}_t = \mathrm{argmax}_i \langle w^{(i)}, x_t \rangle$

In case of a mistake

$$\begin{aligned} \theta^{(\hat{y}_t)}_{t+1,s} &= \theta^{(\hat{y}_t)}_{t,s} - \alpha x_{t,s} \\ \theta^{(y_t)}_{t+1,s} &= \theta^{(y_t)}_{t,s} + \alpha x_{t,s} \end{aligned}$$

for all s such that $|s| \leq d_t$.

Outline

Prediction Suffix Trees

Algorithm and Properties

Results

Outline

Prediction Suffix Trees

Algorithm and Properties

Results

Data

120 sequences of system calls from Outlook, Excel and Firefox (40 each).

The monitoring program records 23 different system calls.

A typical sequence contains hundreds of thousands of system calls.

Results

Averages over the 40 sequences

Outlook	Perceptron	Winnow
% Error	5.1	4.43
PST Size	41239	25679

"Perceptron" is the PST learning algorithm of [Dekel et al., 2004].

Results

Averages over the 40 sequences

Outlook	Perceptron	Winnow
% Error	5.1	4.43
PST Size	41239	25679

Firefox	Perceptron	Winnow
% Error	14.86	13.88
PST Size	21081	12662

"Perceptron" is the PST learning algorithm of [Dekel et al., 2004].

Results

Averages over the 40 sequences

Outlook	Perceptron	Winnow
% Error	5.1	4.43
PST Size	41239	25679

Firefox	Perceptron	Winnow
% Error	14.86	13.88
PST Size	21081	12662

Excel	Perceptron	Winnow
% Error	22.68	20.59
PST Size	24402	15338

"Perceptron" is the PST learning algorithm of [Dekel et al., 2004].

Results

Averages over the 40 sequences

Outlook	Perceptron	Winnow
% Error	5.1	4.43
PST Size	41239	25679

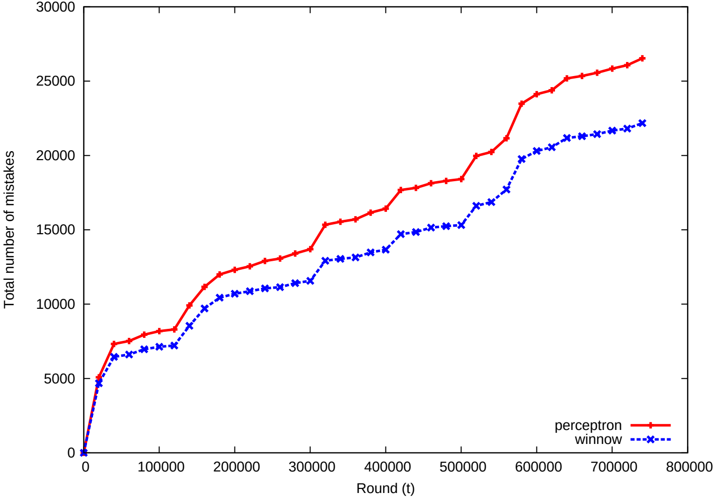
Firefox	Perceptron	Winnow
% Error	14.86	13.88
PST Size	21081	12662

Excel	Perceptron	Winnow
% Error	22.68	20.59
PST Size	24402	15338

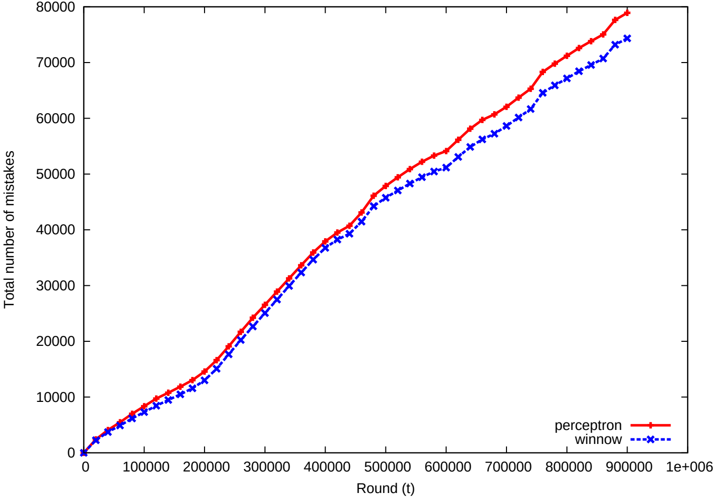
Moreover, Winnow made less mistakes and grew smaller trees for all 120 sequences

"Perceptron" is the PST learning algorithm of [Dekel et al., 2004].

Results

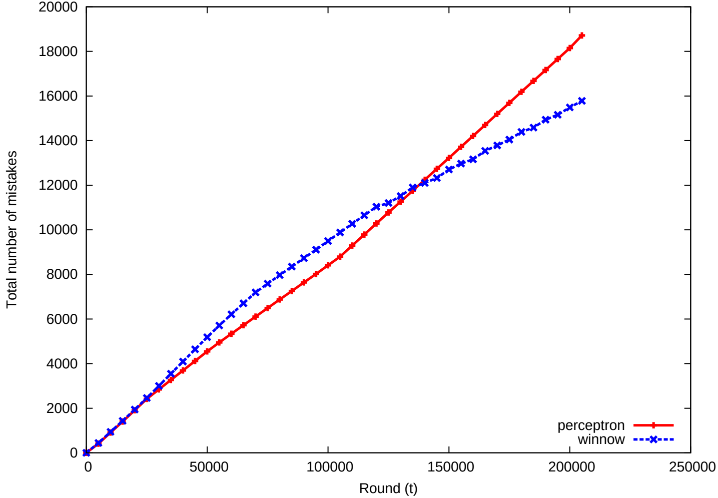

Total number of mistakes
30000
25000
20000
15000
10000
5000
0
0
100000
200000
300000
400000
500000
600000
700000
800000
Round (t)
perceptron
winnow

Results



80000
70000
60000
50000
40000
30000
20000
10000
0
Total number of mistakes
0
100000
200000
300000
400000
500000
600000
700000
800000
900000
1e+006
Round (t)
perceptron
winnow

Results



Total number of mistakes
20000
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
0
50000
100000
150000
200000
250000
Round (t)
perceptron
winnow

Related Work

Many learning algorithms assume an a priori bound on the tree's depth [Willems et al., 1995], [Ron et al., 1996], [Pereira & Singer, 1999]...

[Dekel et al., 2004] present a perceptron algorithm similar to ours.

[Kivinen & Warmuth, 1997] show how to compete against vectors u with $||u||_1 \leq U$

Sparsifying Winnow is popular (e.g. [Blum, 1997]) but no guarantees.

Summary

Outlined the benefits of prediction suffix trees.

Summary

Outlined the benefits of prediction suffix trees.

Introduced an online learning algorithm to learn PSTs.

Summary

Outlined the benefits of prediction suffix trees.

Introduced an online learning algorithm to learn PSTs.

It is competitive with the best fixed PST in hindsight.

Summary

Outlined the benefits of prediction suffix trees.

Introduced an online learning algorithm to learn PSTs.

- It is competitive with the best fixed PST in hindsight.
- The resulting trees grow slowly if necessary

Summary

Outlined the benefits of prediction suffix trees.

Introduced an online learning algorithm to learn PSTs.

- It is competitive with the best fixed PST in hindsight.
- The resulting trees grow slowly if necessary

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

References I

Blum, A. (1997).
Empirical Support for Winnow and Weighted-Majority Algorithms: Results on a Calendar Scheduling Domain.
Machine Learning, *26*, 5–23.

Dekel, O., Shalev-Shwartz, S., & Singer, Y. (2004).
The power of selective memory: Self-bounded learning of prediction suffix trees.
Advances in Neural Information Processing Systems, *17*.

Helmbold, D., & Schapire, R. (1997).
Predicting Nearly As Well As the Best Pruning of a Decision Tree.
Machine Learning, *27*, 51–68.

References II

Kivinen, J., & Warmuth, M. (1997).
Exponentiated Gradient versus Gradient Descent for Linear Predictors.
Information and Computation, *132*, 1–63.

Littlestone, N. (1989).
Mistake bounds and logarithmic linear-threshold learning algorithms.
Doctoral dissertation, Santa Cruz, CA, USA.

Pereira, F., & Singer, Y. (1999).
An Efficient Extension to Mixture Techniques for Prediction and Decision Trees.
Machine Learning, *36*, 183–199.

References III

Ron, D., Singer, Y., & Tishby, N. (1996).
The Power of Amnesia: Learning Probabilistic Automata with Variable Memory Length.
Machine Learning, *25*, 117–149.

Rosenblatt, F. (1958).
The perceptron: A probabilistic model for information storage and organization in the brain.
Psychological Review, *65*, 386–408.

Willems, F., Shtarkov, Y., & Tjalkens, T. (1995).
The context-tree weighting method: basic properties.
IEEE Transactions on Information Theory, *41*, 653–664.