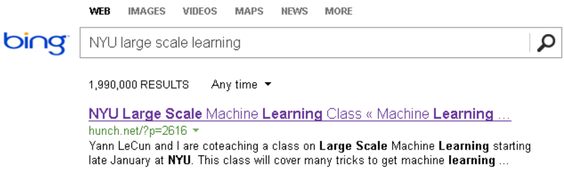
Logarithmic Time Prediction

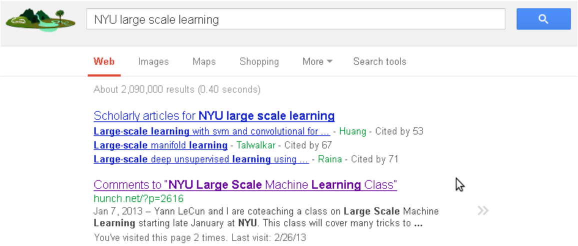
John Langford @ Microsoft Research

Machine Learning the Future, April 24th

A Scenario

You have 10^{10} webpages and want to return the best result in 100ms.





How do you do it?

Who is that?

The Multiclass Prediction Problem

Repeatedly

1. See x
2. Predict $\hat{y} \in \{1, \ldots, K\}$
3. See y

The Multiclass Prediction Problem

Repeatedly

1. See x
2. Predict $\hat{y} \in \{1, \ldots, K\}$
3. See y

Goal: Find $h(x)$ minimizing error rate:

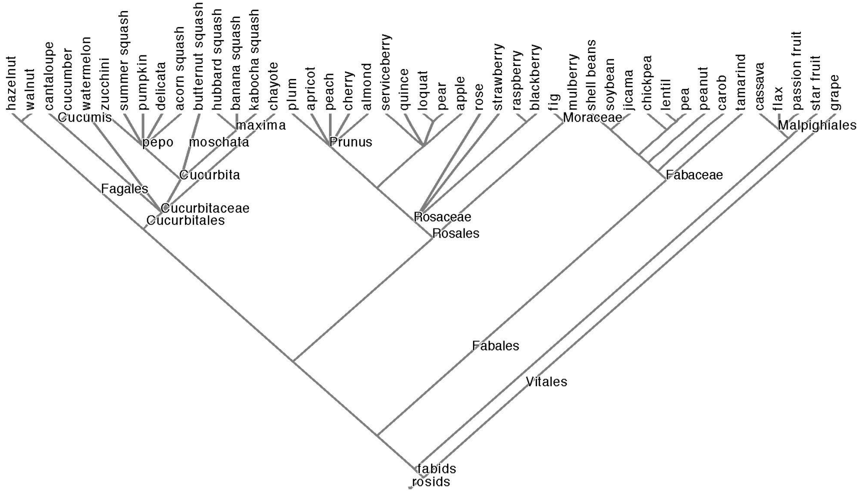
$$\Pr_{(x,y)\sim D}(h(x) \neq y)$$

with $h(x)$ fast.

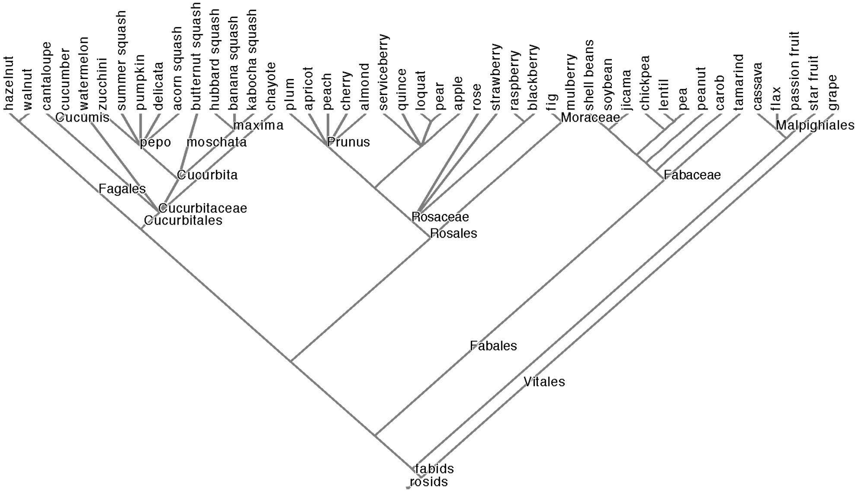
Trick #1

K is small

Trick #2: A hierarchy exists

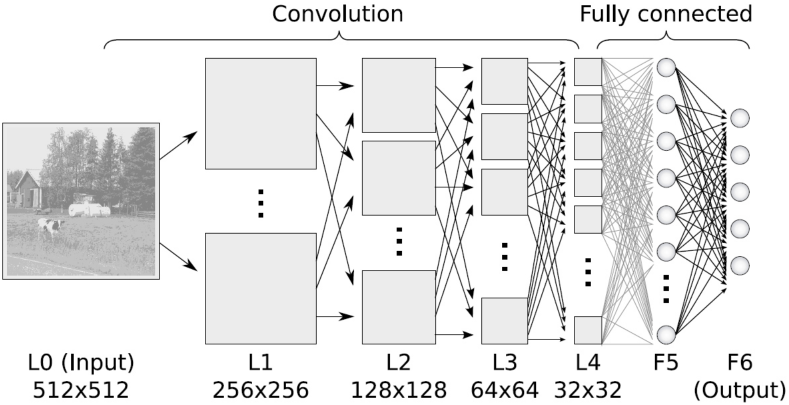


Trick #2: A hierarchy exists

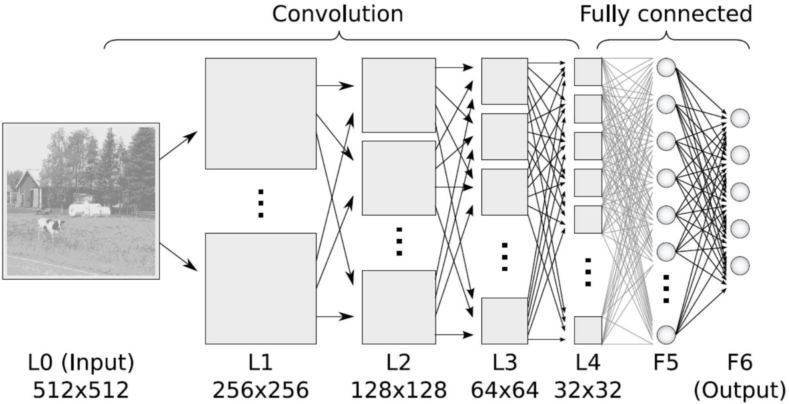


So use Trick #1 repeatedly.

Trick #3: Shared representation

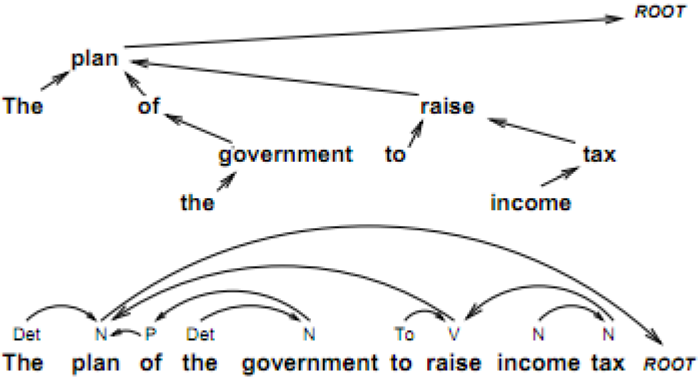


Trick #3: Shared representation

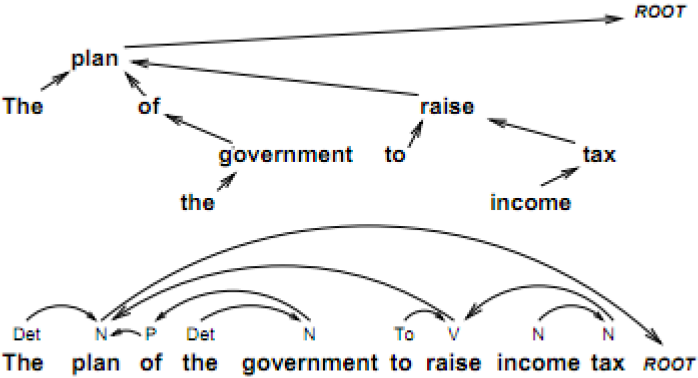


Very helpful... but computation in the last layer can still blow up.

Trick #4: "Structured Prediction"



Trick #4: "Structured Prediction"



But what if the structure is unclear?

Trick #5: GPU

Trick #5: GPU

10 Teraflops is great... yet still burns energy.

Outline

1. Tricks
2. Static Structure
3. Dynamic Structure

How fast can we hope to go?

How fast can we hope to go?

Theorem: There exists multiclass classification problems where achieving 0 error rate requires $\Omega(\log K)$ time to train or test per example.

How fast can we hope to go?

Theorem: There exists multiclass classification problems where achieving 0 error rate requires $\Omega(\log K)$ time to train or test per example.

Proof: Pick $y \sim U(1, ..., K)$

How fast can we hope to go?

Theorem: There exists multiclass classification problems where achieving 0 error rate requires $\Omega(\log K)$ time to train or test per example.

Proof: Pick $y \sim U(1, ..., K)$

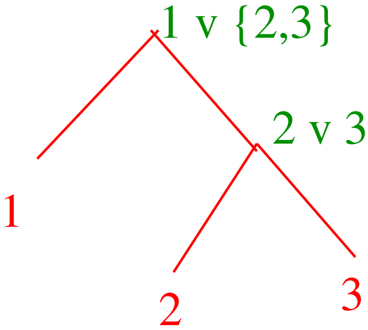
Any prediction algorithm outputting less than $\log_2 K$ bits loses with constant probability.

Any training algorithm reading an example requires $\Omega(\log_2 K)$ time.

Can we predict in time $O(\log_2 K)$?

Can we predict in time $O(\log_2 K)$?

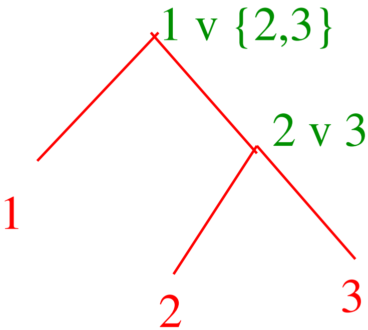
P(y=1) = .4
P(y=2) = .3
P(y=3) = .3



$P(\{2,3\}) > P(1) \Rightarrow$ lose for divide and conquer

Filter Trees [BLR09]

P(y=1) = .4
P(y=2) = .3
P(y=3) = .3



1. Learn 2v3 first
2. Throw away all error examples
3. Learn 1 v Survivors

Theorem: For all multiclass problems, for all binary classifiers, Multiclass Regret $\leq$ Average Binary Regret * $\log(K)$

What about with costs?

Cost-sensitive multi-class classification

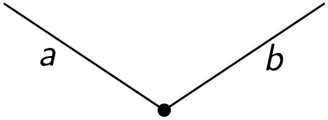
Distribution D over $X \times [0,1]^k$, where a vector in $[0,1]^k$ specifies the cost of each of the k choices.

Find a classifier $h : X \to \{1, \ldots, k\}$ minimizing the expected cost

$$\text{cost}(h, D) = \mathbf{E}_{(x,c)\sim D}[c_{h(x)}].$$

Generalization to the Cost-sensitive Case

To train a non-leaf node on example $(x, c_1, \dots, c_k)$:



$$\text{Let } y = \begin{cases} \text{left} & \text{if } c_a \le c_b \\ \text{right} & \text{otherwise} \end{cases}$$

Train on (x, y) with importance weight $|c_a - c_b|$.

Distribution induced at the node

Draw a cost-sensitive example from D, create an importance weighted sample as above.

Analysis

01 regret:
$\text{reg}_{01}(h, D) = \Pr_D(h(x) \neq y) - \min_{h'} \Pr_D(h'(x) \neq y)$
CS regret:
$\text{reg}_{CS}(h, D) = E_{(x,y)\sim D}[c_{h(x)}] - \min_{h'} E_{(x,y)\sim D}[c_{h'(x)}]$

Theorem

For all CSMC problems and node classifiers,

$$\text{reg}_{CS}(h_{FT}, D) \leq A(\text{reg}_{01}(h, D_{FT}))E_{D_{FT}} \sum_{\text{nodes } n} i_n$$

Analysis

01 regret:
$\mathrm{reg}_{01}(h, D) = \Pr_D(h(x) \neq y) - \min_{h'} \Pr_D(h'(x) \neq y)$
CS regret:
$\mathrm{reg}_{CS}(h, D) = E_{(x,y)\sim D}[c_{h(x)}] - \min_{h'} E_{(x,y)\sim D}[c_{h'(x)}]$

Theorem

For all CSMC problems and node classifiers,

$$\mathrm{reg}_{CS}(h_{FT}, D) \leq A(\mathrm{reg}_{01}(h, D_{FT}))E_{D_{FT}} \sum_{\text{nodes } n} i_n$$

What's multiclass special case?

Analysis

01 regret:
$\text{reg}_{01}(h, D) = \Pr_D(h(x) \neq y) - \min_{h'} \Pr_D(h'(x) \neq y)$
CS regret:
$\text{reg}_{CS}(h, D) = E_{(x,y)\sim D}[c_{h(x)}] - \min_{h'} E_{(x,y)\sim D}[c_{h'(x)}]$

Theorem

For all CSMC problems and node classifiers,

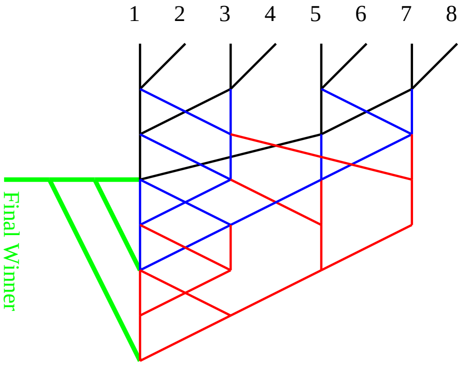
$$\text{reg}_{CS}(h_{FT}, D) \leq A(\text{reg}_{01}(h, D_{FT}))E_{D_{FT}} \sum_{\text{nodes } n} i_n$$

What's multiclass special case? Can you do better?

Error Correcting Tournament

Once an example loses, it moves to the next tournament. Once an example has lost e times, it is eliminated.

The e winners from the first phase compete in the final single elimination tournament. To win in round i, each player must defeat its opponent 2^{i-1} times.



$e = 3$

Summary of Multiclass results

	Filter Tree	Error Correcting Tour.
MC Comp.	$\log k$	$O(\log k)$
MC Regret	$\log k$	5.5
CS Train	k	$O(k \log k)$
CS Test	$\log k$	$O(\log k)$
CS Regret	$\min\{\frac{k}{2}, \sum_n i_n\}$	??

Contextual Bandits in Logarithmic time

Contextual Bandit Classification

Distribution D over $X \times [0,1]^k$, where a vector in $[0,1]^k$ specifies the cost of each of the k choices.

Find a classifier $h : X \to \{1, \ldots, k\}$ minimizing the expected cost

$$\text{cost}(h, D) = \mathbf{E}_{(x,c)\sim D}[c_{h(x)}].$$

given only observations $(x, a, c_a, p_a)^*$.

The Offset Tree for $k = 2$, $p = 0.5$

Suppose $k = 2$ for the moment and let $a \in \{-1, 1\}$. Create binary importance weighted samples according to:

$$\left(x,\ \text{sign}\left(a\left(\frac{1}{2} - c_a\right)\right),\ \left|\frac{1}{2} - c_a\right|\right)$$

x = context
$\text{sign}\left(a\left(\frac{1}{2} - c_a\right)\right)$ = label
$\left|\frac{1}{2} - c_a\right|$ = importance weight

Denoising Binary Importance Weighting

Theorem

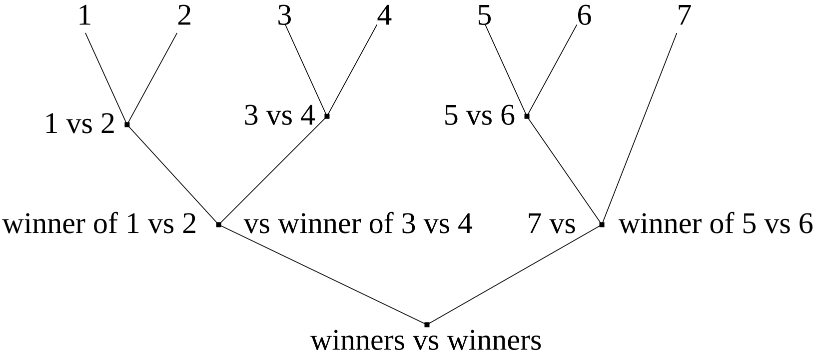
For all Contextual Bandit distributions D with $k = 2$, for all binary classifiers b:

$$\text{policy regret} \leq \text{reg}_{0/1}(b, D_{OT}).$$

Offset reduces noise in induced problem.

$\frac{1}{2}$ = minimax value of the median reward. Plugging in the actual median is always better.

Denoising for $k > 2$ arms



Use the same construction at each node. Internal nodes only get an example if all leaf-wards nodes agree with the label (Filtering trick).

Denoising with k arms

D_{OT} = Induced Binary classification problem
b = the classifier which predicts based on both x and the choice of binary problem according to D_{OT}.

Theorem

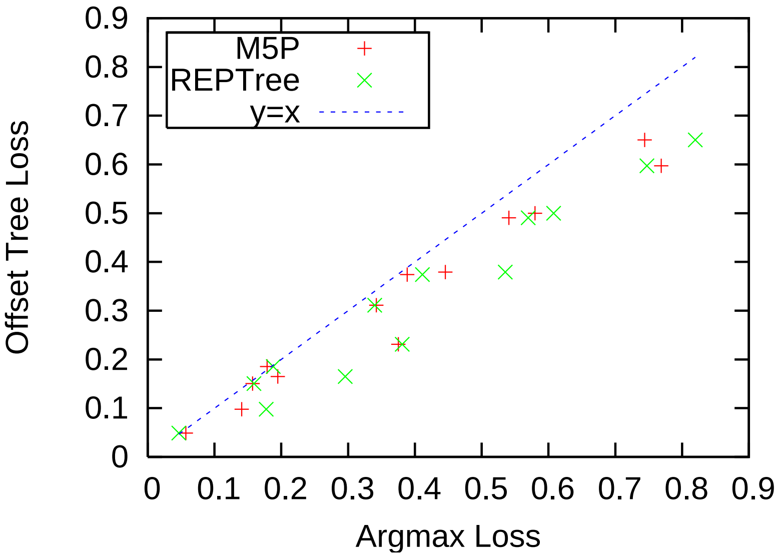
For all k-choice D, binary classifiers b:

$$\text{policy regret} \leq (k-1)\,\text{reg}_{0/1}(b, D_{OT}).$$

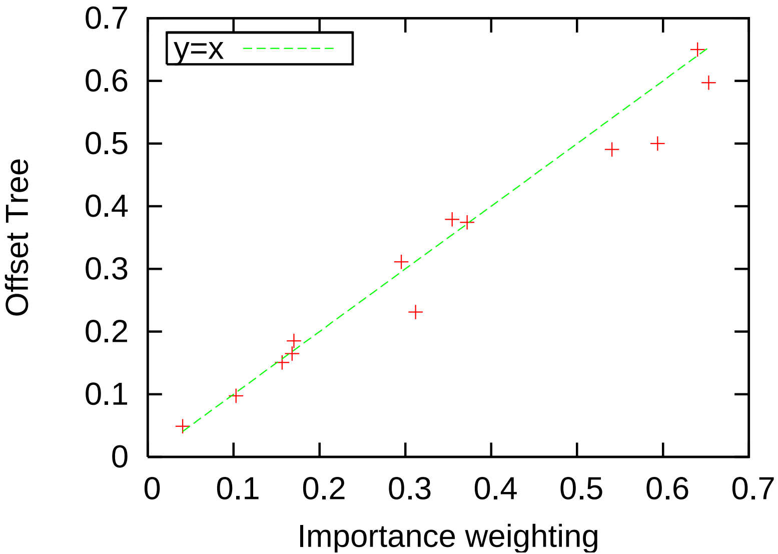
A Comparison of Approaches

Algorithm	Policy Regret Bound
Argmax	$\sqrt{2k \, \mathrm{reg}_{0/1}(s, D_{AR})}$
Importance Weighted	$4k \, \mathrm{reg}_{0/1}(b, D_{IW})$
Offset Tree	$(k-1) \, \mathrm{reg}_{0/1}(b, D_{OT})$

How do you expect things to work, experimentally?

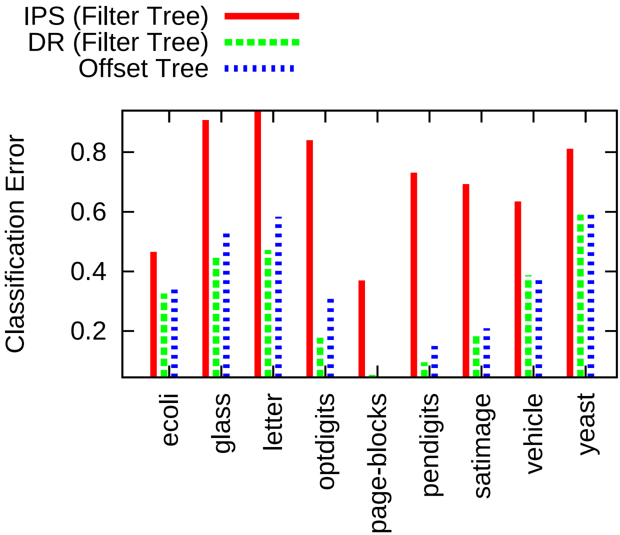


M5P
REPTree
y=x
Offset Tree Loss
Argmax Loss
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9



y=x
Offset Tree
Importance weighting
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7

Compare with Double Robust



DR is exponentially slower, but often a bit better.

Outline

1. Tricks
2. Static Structure
3. Dynamic Structure

How do you learn structure?

Not all partitions are equally difficult.
Compare $\{1,7\}v\{3,8\}$ to $\{1,8\}v\{3,7\}$
What is better?

How do you learn structure?

Not all partitions are equally difficult.
Compare $\{1,7\}v\{3,8\}$ to $\{1,8\}v\{3,7\}$
What is better?

[BWG10]: Better to confuse near leaves than near root.
Intuition: the root predictor tends to be overconstrained while the leafwards predictors are less constrained.

The Partitioning Problem [CL14]

Given a set of n examples each with one of K labels, find a partitioner h that maximizes:

$$E_{x,y} |\Pr(h(x) = 1, y) - \Pr(h(x) = 1) \Pr(y)|$$

The Partitioning Problem [CL14]

Given a set of n examples each with one of K labels, find a partitioner h that maximizes:

$$E_x \sum_y \Pr(y) \left| \Pr(h(x) = 1 | x \in X_y) - \Pr(h(x) = 1) \right|$$

where X_y is the set of x associated with y.

The Partitioning Problem [CL14]

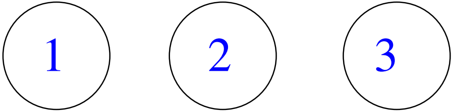
Given a set of n examples each with one of K labels, find a partitioner h that maximizes:

$$E_x \sum_y \Pr(y) \left| \Pr(h(x) = 1 | x \in X_y) - \Pr(h(x) = 1) \right|$$

where X_y is the set of x associated with y.

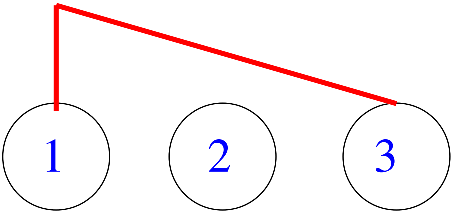
Nonconvex for any symmetric hypothesis class (ouch)

Bottom Up doesn't work



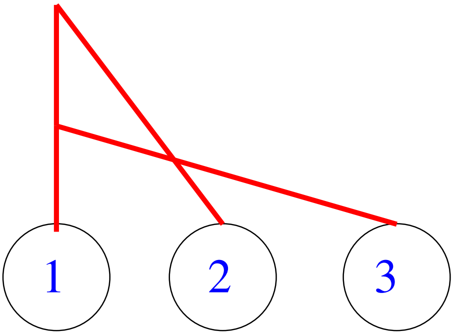
Suppose you use linear representations.

Bottom Up doesn't work



Suppose you use linear representations.
Suppose you first build a 1v3 predictor.

Bottom Up doesn't work



Suppose you use linear representations.
Suppose you first build a 1v3 predictor.
Suppose you then build a 2v{1v3} predictor.
You lose.

How do you train to partition?

Input: Example (x, y); Node n

How do you train to partition?

Input: Example (x, y); Node n
$(\hat{H}_{\mathrm{left}}, \hat{H}'_{\mathrm{left}}) \doteq \mathsf{entropy}(n.\mathrm{left}, y)$
$(\hat{H}_{\mathrm{right}}, \hat{H}'_{\mathrm{right}}) \doteq \mathsf{entropy}(n.\mathrm{right}, y)$
Where entropy = Empirical Shannon entropy without and with y added.

How do you train to partition?

Input: Example (x, y); Node n

$(\hat{H}_{\mathrm{left}}, \hat{H}'_{\mathrm{left}}) \doteq \mathsf{entropy}(n.\mathrm{left}, y)$

$(\hat{H}_{\mathrm{right}}, \hat{H}'_{\mathrm{right}}) \doteq \mathsf{entropy}(n.\mathrm{right}, y)$

Where entropy = Empirical Shannon entropy without and with y added.

$$\hat{H}_{|\mathrm{left}} \doteq \frac{n.\mathrm{left.total}}{n.\mathrm{total}} \hat{H}'_{\mathrm{left}} + \frac{n.\mathrm{right.total}}{n.\mathrm{total}} \hat{H}_{\mathrm{right}}$$

$$\hat{H}_{|\mathrm{right}} \doteq \frac{n.\mathrm{left.total}}{n.\mathrm{total}} \hat{H}_{\mathrm{left}} + \frac{n.\mathrm{right.total}}{n.\mathrm{total}} \hat{H}'_{\mathrm{right}}$$

How do you train to partition?

Input: Example (x, y); Node n

$(\hat{H}_{\mathrm{left}}, \hat{H}'_{\mathrm{left}}) \doteq \text{entropy}(n.\mathrm{left}, y)$

$(\hat{H}_{\mathrm{right}}, \hat{H}'_{\mathrm{right}}) \doteq \text{entropy}(n.\mathrm{right}, y)$

Where entropy = Empirical Shannon entropy without and with y added.

$$\hat{H}_{|\mathrm{left}} \doteq \frac{n.\mathrm{left.total}}{n.\mathrm{total}} \hat{H}'_{\mathrm{left}} + \frac{n.\mathrm{right.total}}{n.\mathrm{total}} \hat{H}_{\mathrm{right}}$$

$$\hat{H}_{|\mathrm{right}} \doteq \frac{n.\mathrm{left.total}}{n.\mathrm{total}} \hat{H}_{\mathrm{left}} + \frac{n.\mathrm{right.total}}{n.\mathrm{total}} \hat{H}'_{\mathrm{right}}$$

$$\widehat{\Delta H}_{\mathrm{post}} \leftarrow \hat{H}_{|\mathrm{left}} - \hat{H}_{|\mathrm{right}}$$

How do you train to partition?

Input: Example (x, y); Node n

$(\hat{H}_{\mathrm{left}}, \hat{H}'_{\mathrm{left}}) \doteq \mathrm{entropy}(n.\mathrm{left}, y)$

$(\hat{H}_{\mathrm{right}}, \hat{H}'_{\mathrm{right}}) \doteq \mathrm{entropy}(n.\mathrm{right}, y)$

Where $\mathrm{entropy}$ = Empirical Shannon entropy without and with y added.

$\hat{H}_{|\mathrm{left}} \doteq \frac{n.\mathrm{left.total}}{n.\mathrm{total}} \hat{H}'_{\mathrm{left}} + \frac{n.\mathrm{right.total}}{n.\mathrm{total}} \hat{H}_{\mathrm{right}}$

$\hat{H}_{|\mathrm{right}} \doteq \frac{n.\mathrm{left.total}}{n.\mathrm{total}} \hat{H}_{\mathrm{left}} + \frac{n.\mathrm{right.total}}{n.\mathrm{total}} \hat{H}'_{\mathrm{right}}$

$\widehat{\Delta H}_{\mathrm{post}} \leftarrow \hat{H}_{|\mathrm{left}} - \hat{H}_{|\mathrm{right}}$

$\mathrm{Learn}_n(x, |\widehat{\Delta H}_{\mathrm{post}}|, \mathrm{sign}(\widehat{\Delta H}_{\mathrm{post}}))$

Important Optimizations

1. Do not descend to a pure leaf. Halt early and train one-against-some classifiers for each label.
2. Use large deviation bound on recall to control halting.
3. Add node id features as you descend the tree.

A boosting theorem

γ-Weak Learning Assumption

For all distributions $D(x, y)$ a learning algorithm using examples $(x, y)^*$ IID from D finds a binary classifier $c : X \rightarrow \{l, r\}$ satisfying

$$p_l H_l + p_r H_r \leq H_n - \gamma$$

A boosting theorem

γ-Weak Learning Assumption

For all distributions $D(x, y)$ a learning algorithm using examples $(x, y)^*$ IID from D finds a binary classifier $c : X \rightarrow \{l, r\}$ satisfying

$$p_l H_l + p_r H_r \leq H_n - \gamma$$

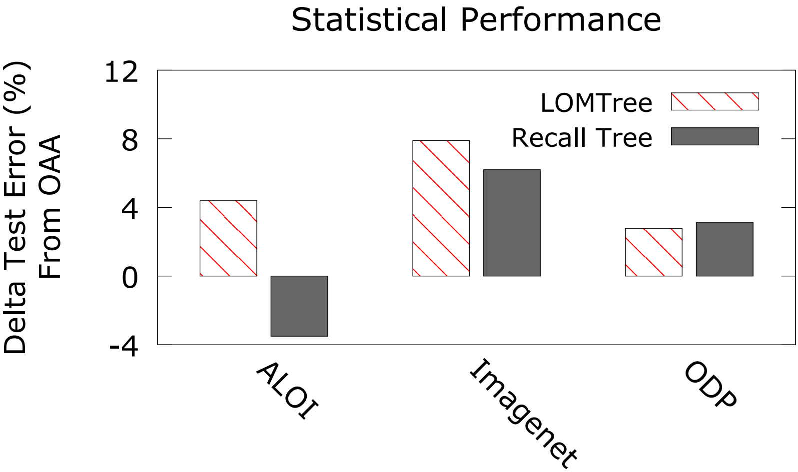
Theorem

If γ-Weak Learning holds, then after t splits the multiclass error rate ϵ of the tree is bounded by:

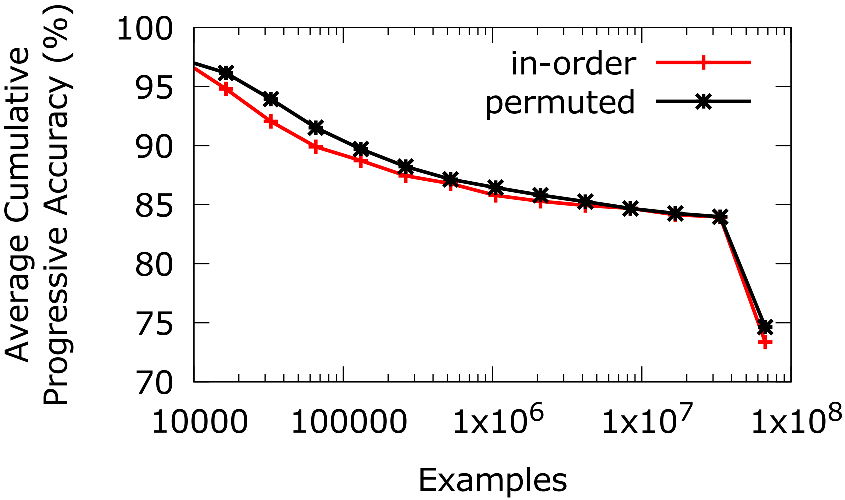
$$\epsilon \leq H_1 - \gamma \ln(t + 1)$$

where H_1 is the class label distribution entropy.

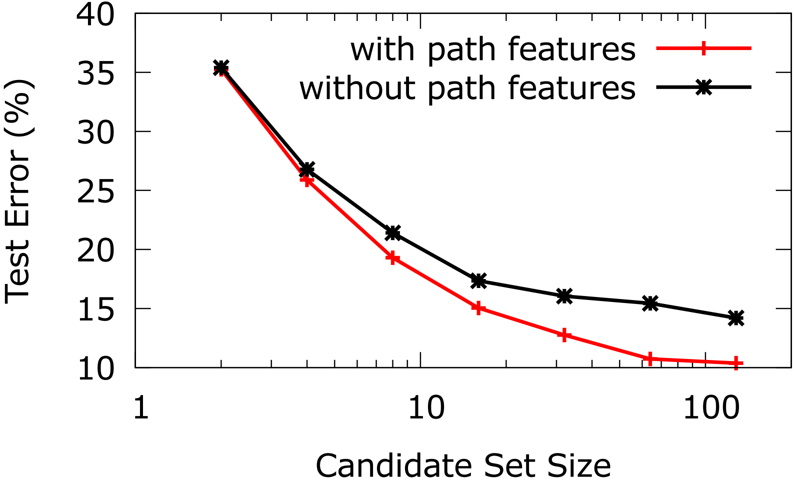
Accuracy vs. LOMtree, OAA



Online performance (LTCB)



With/without path features (ALOI)



Can we predict in time $O(\log_2 K)$?

Can we predict in time $O(\log_2 K)$?

What is the right way to achieve consistency and dynamic partition?

Can we predict in time $O(\log_2 K)$?

What is the right way to achieve consistency and dynamic partition?

How can you balance representation complexity and sample complexity?

Bibliography: Static

Alina Beygelzimer, John Langford, Pradeep Ravikumar, Error-Correcting Tournaments, ALT 2009.

Alina Beygelzimer, John Langford, Error-Correcting Tournaments, KDD 2009.

Bibliography: Dynamic

Samy Bengio, Jason Weston, David Grangier, Label embedding trees for large multi-class tasks, NIPS 2010.

Anna Choromanska, John Langford, Logarithmic Time Online Multiclass prediction, NIPS 2015.

Hal Daumé III, Nikos Karampatziakis, John Langford, Paul Mineiro, Logarithmic Time One-Against-Some, https://arxiv.org/abs/1606.04988