Lazy Decision Trees

Ronny Kohavi

Data Mining and Visualization Group

Silicon Graphics, Inc.

Joint work with

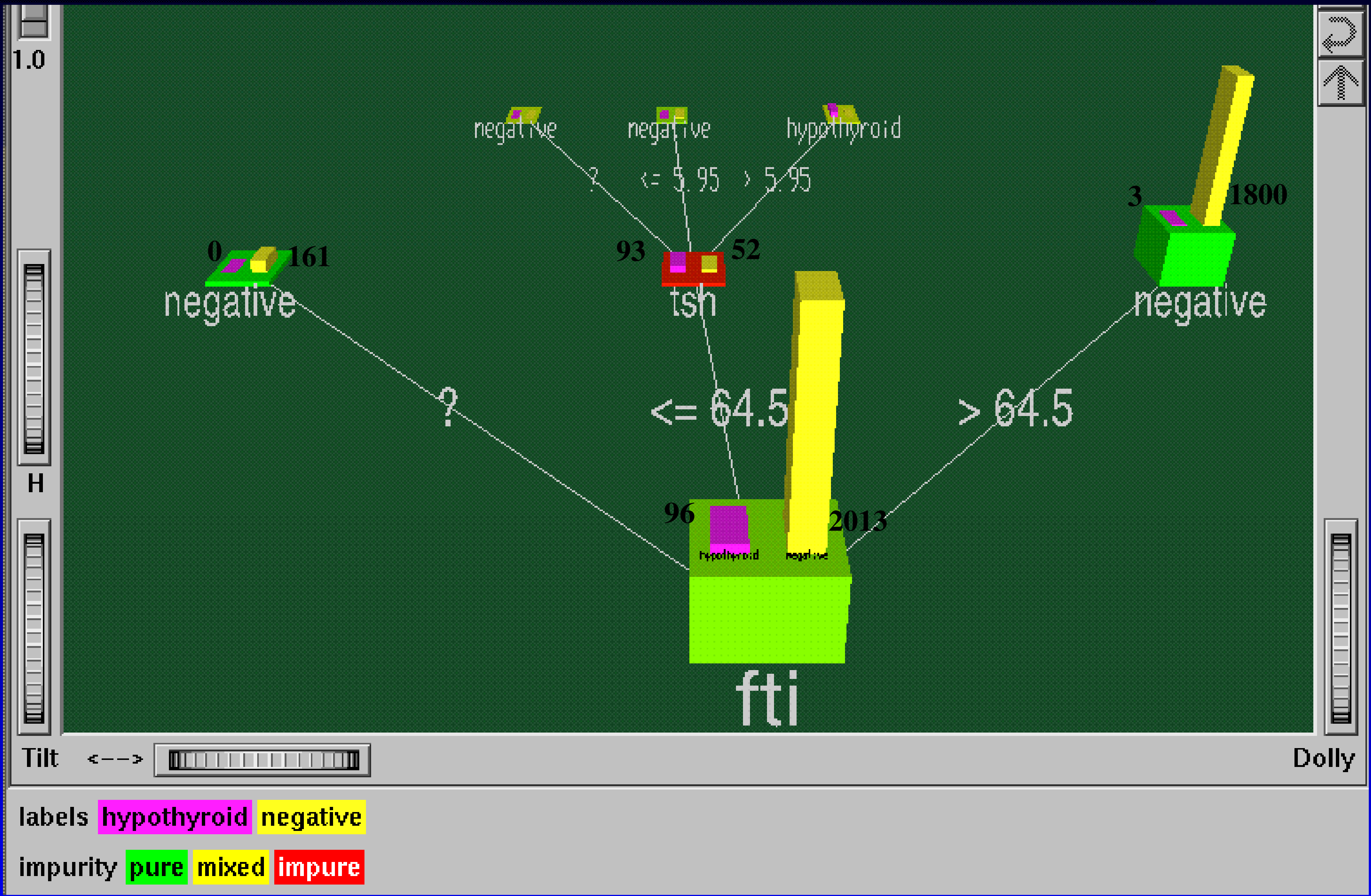
Jerry Friedman

and

Yeogirl Yun

Stanford University

Motivation: Average Impurity ≠ interesting impurity



Eager and Lazy Learning

- **Eager** decision–tree algorithms (e.g., C4.5, CART, ID3) create a single decision tree for classification.
  The inductive leap is attributed to the building of this decision tree.

- **Lazy** learning algorithms (e.g., nearest –neighbors, and this paper) do not build a concise representation of the classifier and wait for the test instance to be given.
  The inductive leap is attributed to the classifier; little (if any) is done during the training phase.

Problems with Eager Decision

- **Replication and fragmentation**: As a tree is built, the number of instances in every node decreases.
  If many features are relevant, we may not have enough data to make the number of splits needed.
- **Unknown values:** Complex methods are usually employed. C4.5 penalizes attributes using induction and does multi–way splits during classification; CART finds surrogate splits.

Lazy DTs: Basic Observation

- In theory, we would like to select the best decision tree for each test instance, i.e., pick the best tree from all possible trees.
- Observation: only the path the test instance takes really matters.

  We don’t need to search or build all possible trees, but at possible paths.

The LazyDT Algorithm *(recursive)*

- Input: training set T of labelled instances. Instance I to classify.
- Output: a label for instance I.

stopping criterion

1. If T is pure (all instances have same label L), return label L.
2. If all instances have the same feature values, return the majority class in T.

recursive step

3. Select a test X and let x be the value of the test on instance I. Assign the test of instances with X=x to T and apply the algorithm to T.

The Split Measure Isn't Obvious

- The "obvious" measure, the difference in entropies between the parent and the child node (into which the test instance trickles), is not a good idea:
  - The difference in entropies may be negative. In fact, if A is dominant, for B to be dominant we may need to increase the entropy first.
  - 80/20 and 20/80 have the same entropy, but they are very different.

Our Choice of Splitting Criteria

- We chose to reweight the instances at every node so that all classes have equal probability.
- The entropy for each child is computed using the weighted instances.
- This method ensures that:
  - The difference in entropy is always non–negative.
  - Changes from 80/20 to 20/80 are very signfiicant.

The LazyDT Implementation

- As with standard decision trees, we chose to limit ourselves to univariate splits (single attr).
- We allow splits on single values to fine–tune the parititions and avoid fragmentation.
- To speed the classification, we:
  - Discretize the data (global discretization).
  - Cache the impurity measures as we compute them. Because only a few attributes get chosen at every node, the cache was very effective.

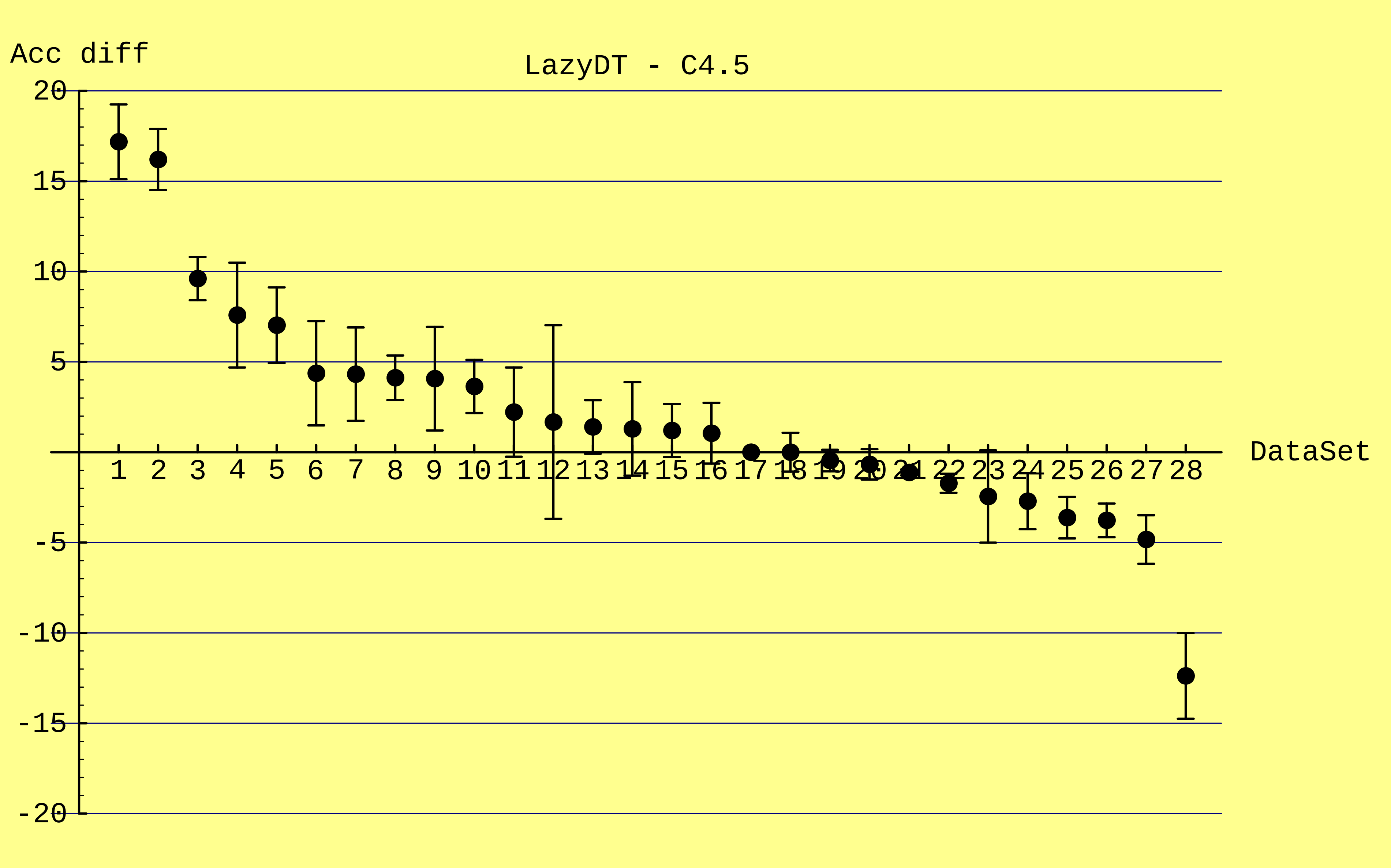
Missing Values

- LazyDT never considers a split on an attribute whose value is unknown.

Contrast with

- C4.5 penalizes attributes with missing values based on the ratio of missing values.
  An attribute, such as tested–for–AIDS,
  may be missing from most instance and never chosen by C4.5 because of that. However,
  if the test instance has a value, it might be extremely useful and LazyDT will use it.
- CART computes surrogates to use instead.

Experiments



Interesting Observation

- For the Anneal dataset, ID3 outperformed both LazyDT and C4.5 (0% error versus 5.9% and 8.4%).

  Reason: unknown handling. Our ID3 considered unknowns as a separate value.

  Xref: Schaffer's paper showing how NN beat C4.5 (encoding for NN was as separate value).

  It's all in the representation.
- Changing the "?" to Unknown reduced the C4.5 error from 8.4% to 1.3%

Other Interesting Differences

- C4.5 outperformed LazyDT on audiology. Reason: 69 features, 24 classes, 226 instances. LazyDT clearly overfits (variance problem). Note that LazyDT as implemented does no pruning (not obvious how to do it).

- LazyDT significantly outperformed C4.5 on tic–tac–toe. Concept is whether X won in an end–game. LazyDT can split on squares that have X's (or at least are non–blank) while decision trees need to pick the squares in advance.

Related Work

- Lazy learning issue (special issue of AI review to appear).
- Friedman, Flexible metric nearest–neighbor.
- Hastie and Tibshirani, Discriminant adaptive nearest neighbor classification.
- Holte, Acker & Porter: small disjuncts (could LazyDT help?); Quinlan, improved estimates for small disjuncts.

Future Work

LazyDT is far from perfect:

- **There is no regularization (pruning). We proceed until the leaf is pure.**
- **Data is discretized in advance. That's very eager and local interactions are lost. (without discretization caching won't work well and classification would be very slow).**
- **Compare dynamic complexity (Holte), i.e., the number of splits until a decision is made.**

Summary

- LazyDT creates a path in a tree that would be "best" for a given test instance.
- The small single–attribute splits coupled with the choice of path reduce fragmentation and allow handling problems with many relevant attributes.
- Missing values are naturally handled by avoiding splits on such values.
- Disadvantages: no pruning, pre–discretization.