Happy Wednesday!

- Quiz 9, Friday, Oct 23th 6am until Oct 24th 11:59am (noon)
  - Decision trees
- Assignment 3 due Mon, Oct 26th, 11:59 pm (midnight)

Coming up soon

- Touch-point 2: deliverables due Mon, Oct 30th, live-event Wed, Nov 2nd
  - Single-slide presentation outlining progress highlights and current challenges
  - Three-minute pre-recorded presentation with your progress and current challenges
- Project midpoint report due Nov 6th 11:59pm (midnight)
  - GitHub page with the results you have achieved utilizing unsupervised learning

CS4641B Machine Learning

Lecture 18: Ensemble learning

Rodrigo Borela ▸ rborelav@gatech.edu

These slides are adapted from Polo Chau and Mahdi Roozbahani

Decision trees so far

- Given N datapoints from training data, each with D features ($\mathbf{X}$) and corresponding target values ($\mathbf{t}$) , construct a sequence of tests (decision tree) to predict the label from the attributes

- Basic strategy for defining the tests ("when to split") → maximize the information gain on the training data set at each node of the tree

- Problems:
  - Computational issues → How expensive is it to compute the IG?
  - The tree will end up being much too big → pruning
  - Evaluating the tree on training data is dangerous → overfitting

Important questions

- How to choose the attribute and value to split on at each level of the tree?
- When to stop splitting? When should a node be declared a leaf?
- If a leaf node is impure, how should the class label be assigned?
- If the tree is too large, how can it be pruned?

What will happen if a tree is too large?

- Overfitting
- High variance
- Instability in predicting test data

How to avoid overfitting?

- Acquire more training data
- Remove irrelevant attributes (manual process – not always possible)
- Grow full tree, then post-prune
- Ensemble learning

Reduced-error pruning

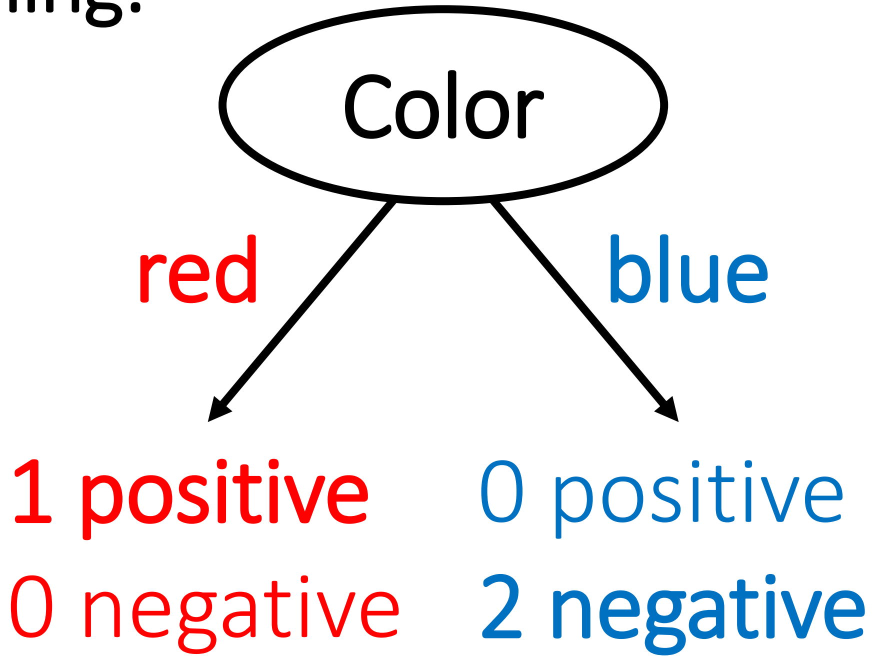
- Split data into training and validation sets
- Grow tree based on training set
- Do until further pruning is harmful:
  - 1. Evaluate impact on validation set of pruning each possible node (plus those below it)
  - 2. Greedily remove the node that most improves validation set accuracy

How to decide to remove it a node using pruning

- Pruning of the decision tree is done by replacing a whole subtree by a leaf node.
- The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf.

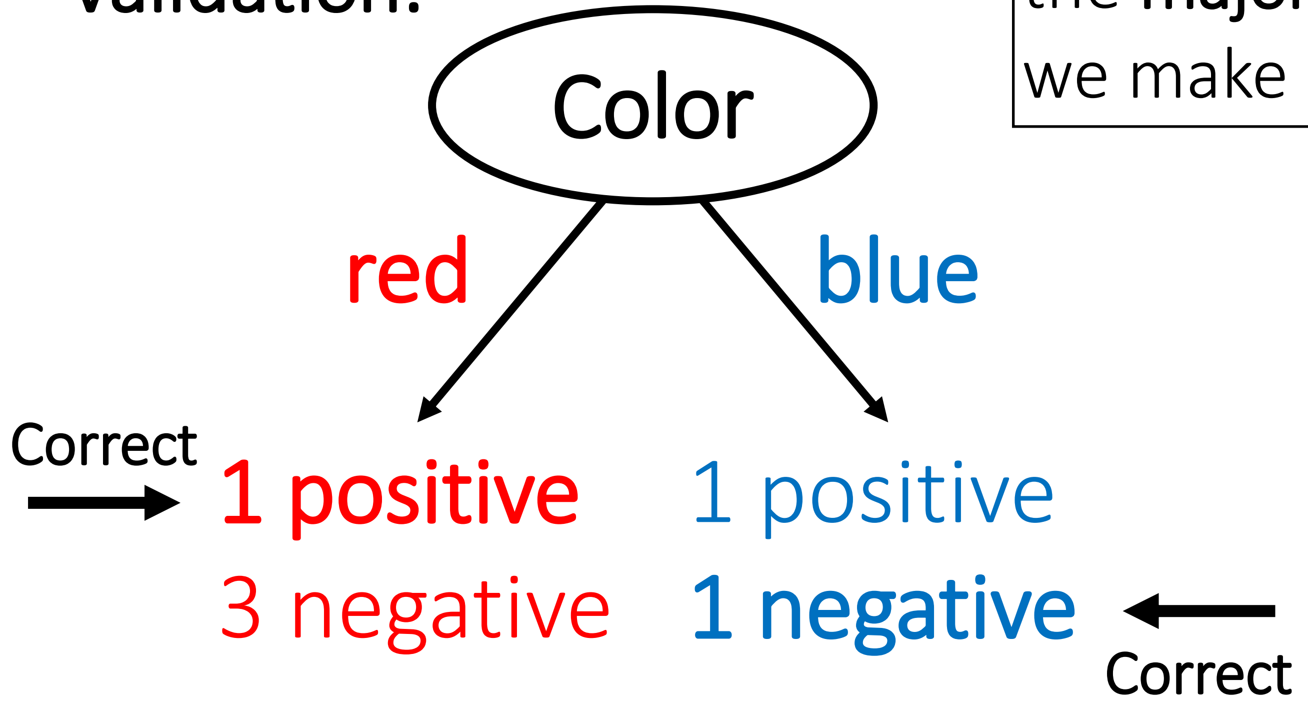
If we had simply predicted the **majority class** (negative), we make 2 errors instead of 4

Training:



3 training data points
Actual label: 1 positive and 2 negative
Predicted label: 1 positive and 2 negative
3 correct and **0** incorrect

Validation:



6 validation data points
Actual label: 2 positive and 4 negative
Predicted label: 4 positive and 2 negative
2 correct and **4** incorrect

Which classifier/model to choose?

- Possible strategies:
  - Go from simplest model to more complex model until you obtain desired accuracy
  - Discover a new model if the existing ones do not work for you
  - Combine all (simple) models

Common Strategy: Bagging (Bootstrap Aggregating)

- Originally designed for combining multiple models, to improve classification "stability" (Leo Breiman, 94)
- Uses random training datasets (sampled from one dataset)
- Consider the data set $S = \{(\mathbf{x}_n, t_n)\}_{n=1,\ldots,N}$
- Pick a sample S^* with replacement of size N (S^* called a "bootstrap sample")

$$S \rightarrow \mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \\ \mathbf{x}_4^T \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 9 & 10 & 11 & 12 \\ 20 & 21 & 22 & 23 \\ 5 & 6 & 7 & 8 \end{bmatrix} \qquad \mathbf{t} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

$$S^* \rightarrow \mathbf{X}^* = \begin{bmatrix} \mathbf{x}_2^T \\ \mathbf{x}_4^T \\ \mathbf{x}_2^T \\ \mathbf{x}_1^T \end{bmatrix} = \begin{bmatrix} \mathbf{9} & \mathbf{10} & \mathbf{11} & \mathbf{12} \\ 5 & 6 & 7 & 8 \\ \mathbf{9} & \mathbf{10} & \mathbf{11} & \mathbf{12} \\ 1 & 2 & 3 & 4 \end{bmatrix} \qquad \mathbf{t}^* = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

Common Strategy: Bagging (Bootstrap Aggregating)

- Consider the data set $S = \{(\mathbf{x}_n, t_n)\}_{n=1,\dots,N}$
- Pick a sample S^* with replacement of size N (S^* called a “bootstrap sample”)
- Train on S^* to get classifier f^*
- Repeat above steps B times get $f_1, f_2, \dots, f_B$
- Final classifier $f(\mathbf{x}) = majority\{f_b(\mathbf{x})\}_{b=1,\dots,B}$

Common Strategy: Bagging (Bootstrap Aggregating)

- Why would bagging work?
  - Combining multiple classifiers reduces the variance of the final classifier

- When would this be useful?
  - When we have a classifier with high variance

Bagging decision trees

- Consider the data set S
- Pick a sample S^* with replacement of size N
- Grow a decision tree T_b
- Repeat B times to get $T_1, \dots, T_B$
- The final classifier will be

$$f(\mathbf{x}) = majority\{f_{T_b}(\mathbf{x})\}_{b=1,\dots,B}$$

Random forests

- Almost identical to bagging decision trees, except we introduce some randomness:
- Randomly pick M of the D available features, at every split when growing the tree (i.e., $D - M$ features ignored)

- Bagged **random** decision trees = **Random forests**

What are our hyperparameters in random forest

- $M =$ Number of randomly chosen attributes
- Usual values for $M = \sqrt{D} \in (1,10)$, where D is number of dimensions, or features, or attributes

- How to optimize M?
  - Cross-validation

- How to optimize B, the number of models or decision trees in random forest?
  - Keep adding trees until training error stabilizes (reaches to a plateau)