

A large tree with vibrant red autumn leaves against a clear blue sky. The tree's branches are dense with bright red foliage, and the sky is a uniform, clear blue. The overall scene is bright and clear, suggesting a sunny day in autumn.

# Decision Trees

Machine Learning  
CS5824/ECE5424  
Bert Huang  
Virginia Tech

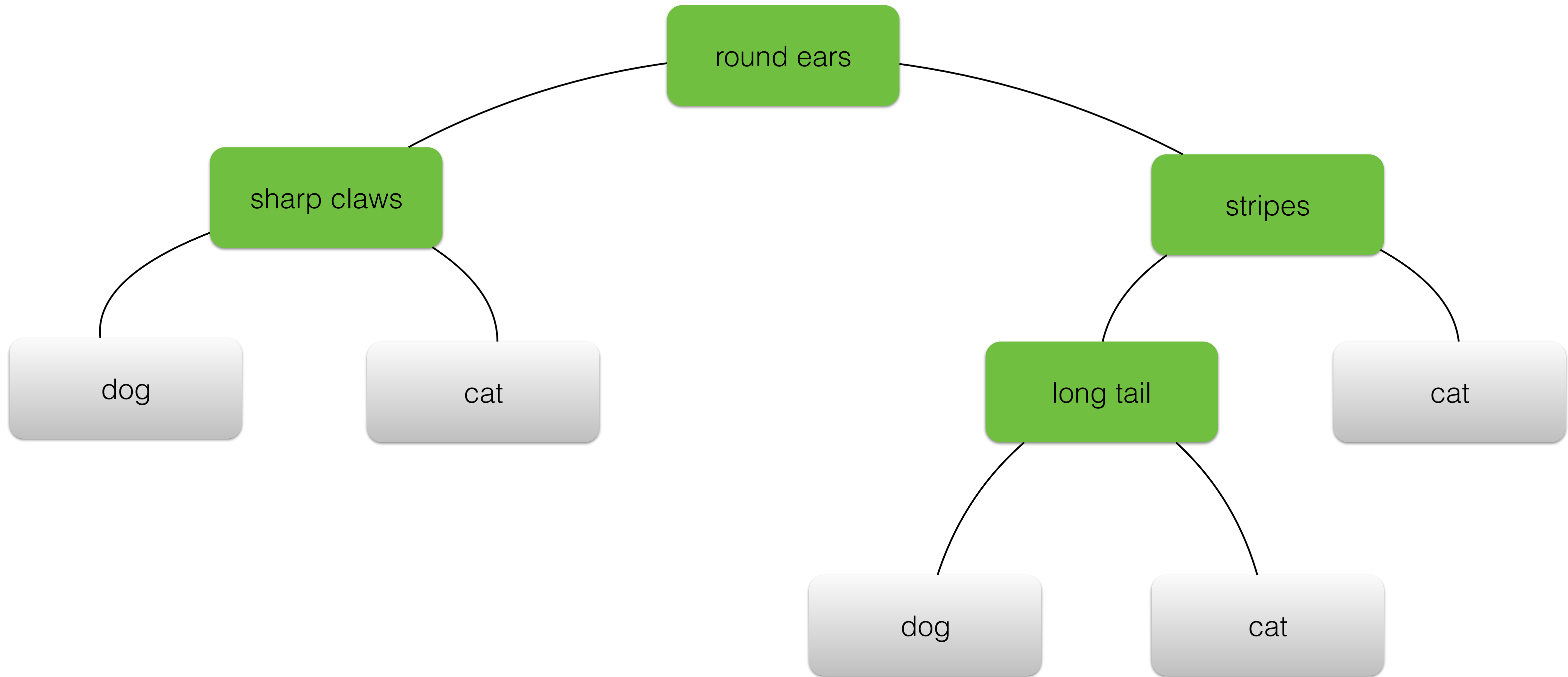


# Nearest Neighbor Classification

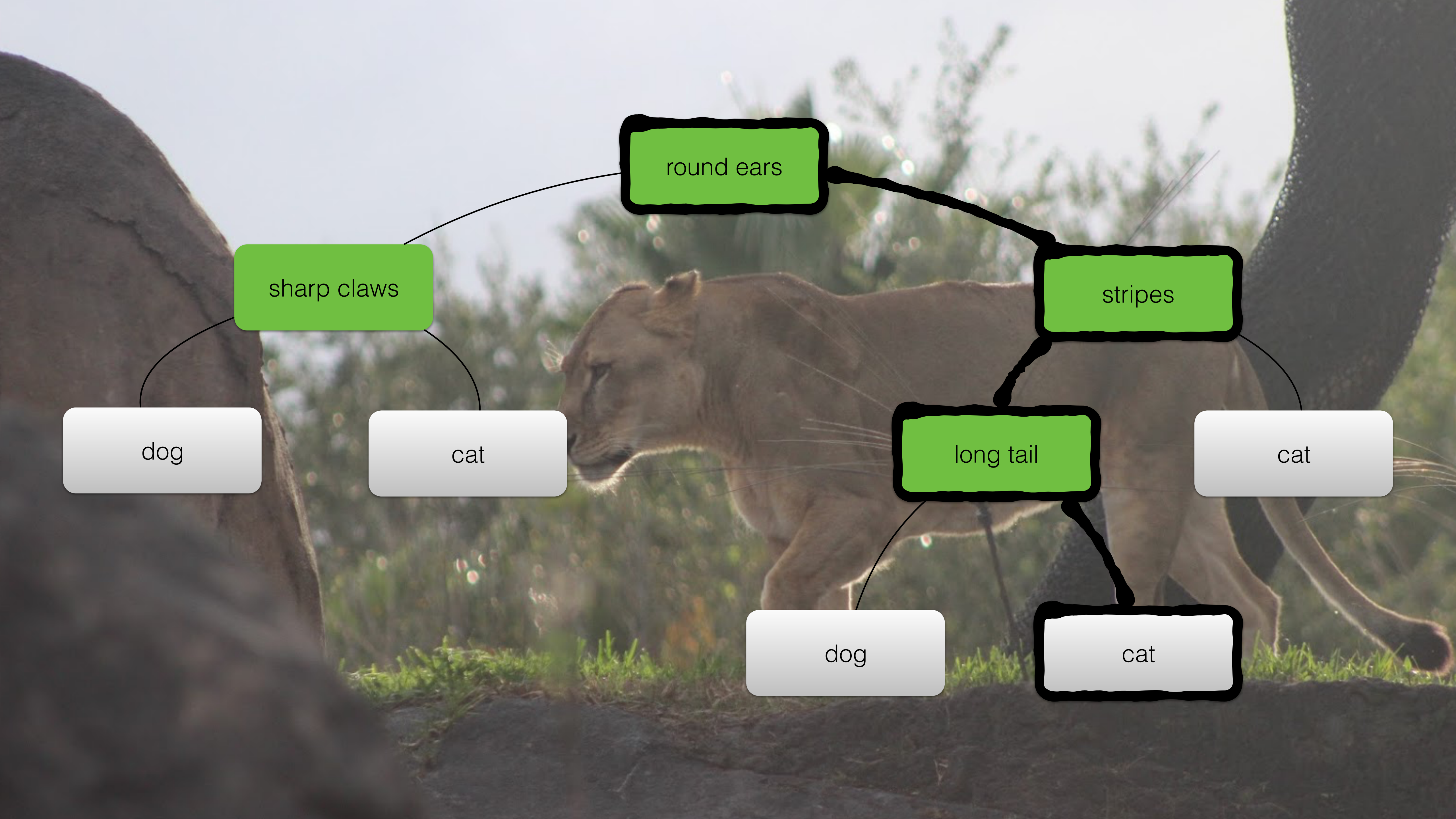
- $D = \{(x_1, y_1), \dots, (x_m, y_m)\}$
- $h(x) = y_k$ , where  $k = \operatorname{argmin}_i d(x, x_i)$
- How much does this cost?
  - How to make it cheaper?
- How to improve this to avoid mistakes?

# Outline

- Learning decision trees
- Extensions: random forests







round ears

sharp claws

stripes

dog

cat

long tail

cat

dog

cat



# Decision Tree Learning

- Greedily choose best decision rule
- Recursively train decision tree for each resulting subset

```
function fitTree(D, depth)
  if D is all one class or depth >= maxDepth
    node.prediction = most common class in D
    return node
  rule = BestDecisionRule(D)
  dataLeft = {(x, y) from D where rule(D) is true}
  dataRight = {(x, y) from D where rule(D) is false}
  node.left = fitTree(dataLeft, depth+1)
  node.right = fitTree(dataRight, depth+1)
  return node
```

```
function fitTree(D, depth)
  if D is all one class or depth >= maxDepth
    node.prediction = most common class in D
  return node
  rule = BestDecisionRule(D)
  dataLeft = {(x, y) from D where rule(D) is true}
  dataRight = {(x, y) from D where rule(D) is false}
  node.left = fitTree(dataLeft, depth+1)
  node.right = fitTree(dataRight, depth+1)
  return node
```

# Choosing Decision Rules

- Define a cost function **cost(D)**
  - Misclassification rate (training error)
  - Entropy or information gain
  - Gini index (smoothed training error)



# Misclassification Rate

$$\hat{\pi}_c := \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \mathbb{I}(y_i = c)$$

class proportion  
(estimated probability)

$$\hat{y} := \operatorname{argmax}_c \hat{\pi}_c$$

best prediction

$$\operatorname{cost}(\mathcal{D}) := \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \mathbb{I}(y_i \neq \hat{y}) = 1 - \hat{\pi}_{\hat{y}}$$

error rate

$$\operatorname{cost}(\mathcal{D}) - \left( \frac{|\mathcal{D}_L|}{|\mathcal{D}|} \operatorname{cost}(\mathcal{D}_L) + \frac{|\mathcal{D}_R|}{|\mathcal{D}|} \operatorname{cost}(\mathcal{D}_R) \right)$$

cost reduction



# Entropy and Information Gain

$$\hat{\pi}_c := \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \mathbb{I}(y_i = c)$$

$$\text{cost}(\mathcal{D}) - \left( \frac{|\mathcal{D}_L|}{|\mathcal{D}|} \text{cost}(\mathcal{D}_L) + \frac{|\mathcal{D}_R|}{|\mathcal{D}|} \text{cost}(\mathcal{D}_R) \right)$$

$$H(\hat{\pi}) := - \sum_{c=1}^C \hat{\pi}_c \log \hat{\pi}_c$$

$$\text{infoGain}(j) = H(Y) - H(Y|X_j)$$

$$= - \sum_y \Pr(Y = y) \log \Pr(Y = y) +$$

$$\sum_{x_j} \Pr(X_j = x_j) \sum_y \Pr(Y = y|X_j = x_j) \log \Pr(Y = y|X_j = x_j).$$



# Information Gain

$$\begin{aligned}\text{infoGain}(j) &= H(Y) - H(Y|X_j) \\ &= - \sum_y \Pr(Y = y) \log \Pr(Y = y) + \\ &\quad \sum_{x_j} \Pr(X_j = x_j) \sum_y \Pr(Y = y|X_j = x_j) \log \Pr(Y = y|X_j = x_j).\end{aligned}$$

$$X_j = Y$$

$$X_j \perp Y$$

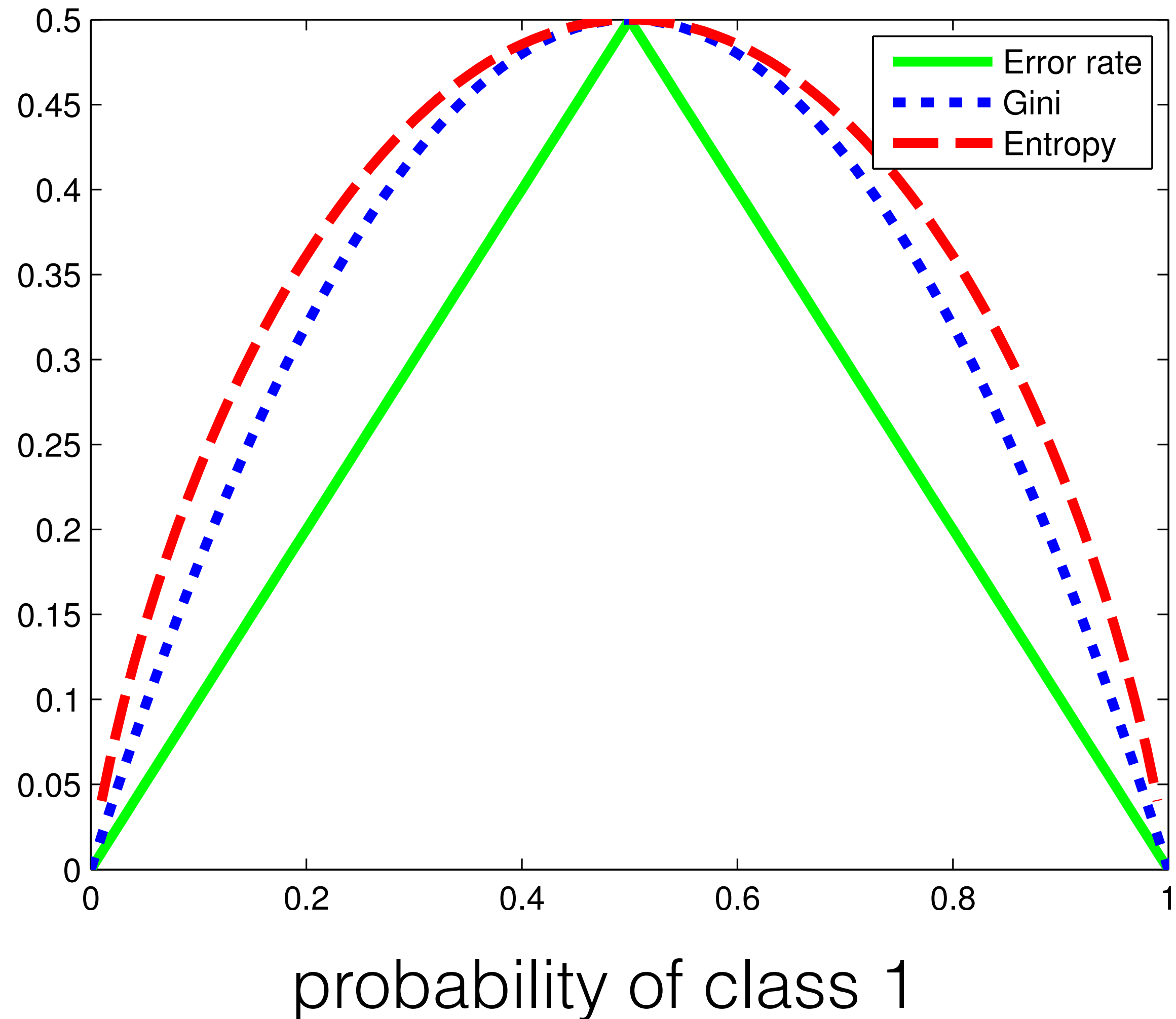


# Gini Index

$$\sum_{c=1}^C \hat{\pi}_c(1 - \hat{\pi}_c) = \sum_c \hat{\pi}_c - \sum_c \hat{\pi}_c^2 = 1 - \sum_c \hat{\pi}_c^2$$

like misclassification rate, but accounts for uncertainty

# Comparing the Metrics



% Fig 9.3 from Hastie book

```
p=0:0.01:1;  
gini = 2*p.*(1-p);  
entropy = -p.*log(p) - (1-p).*log(1-p);  
err = 1-max(p,1-p);
```

```
% scale to pass through (0.5, 0.5)  
entropy = entropy./max(entropy) * 0.5;
```

```
figure;  
plot(p, err, 'g-', p, gini, 'b:', p, ...  
      entropy, 'r--', 'linewidth', 3);  
legend('Error rate', 'Gini', 'Entropy')
```



# Overfitting

- A decision tree can achieve 100% training accuracy when each example is unique
- Limit depth of tree
- Strategy: train very deep tree
  - Adaptively prune

# Validation

- Take training data and split into **training set** and **validation set**

- $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$

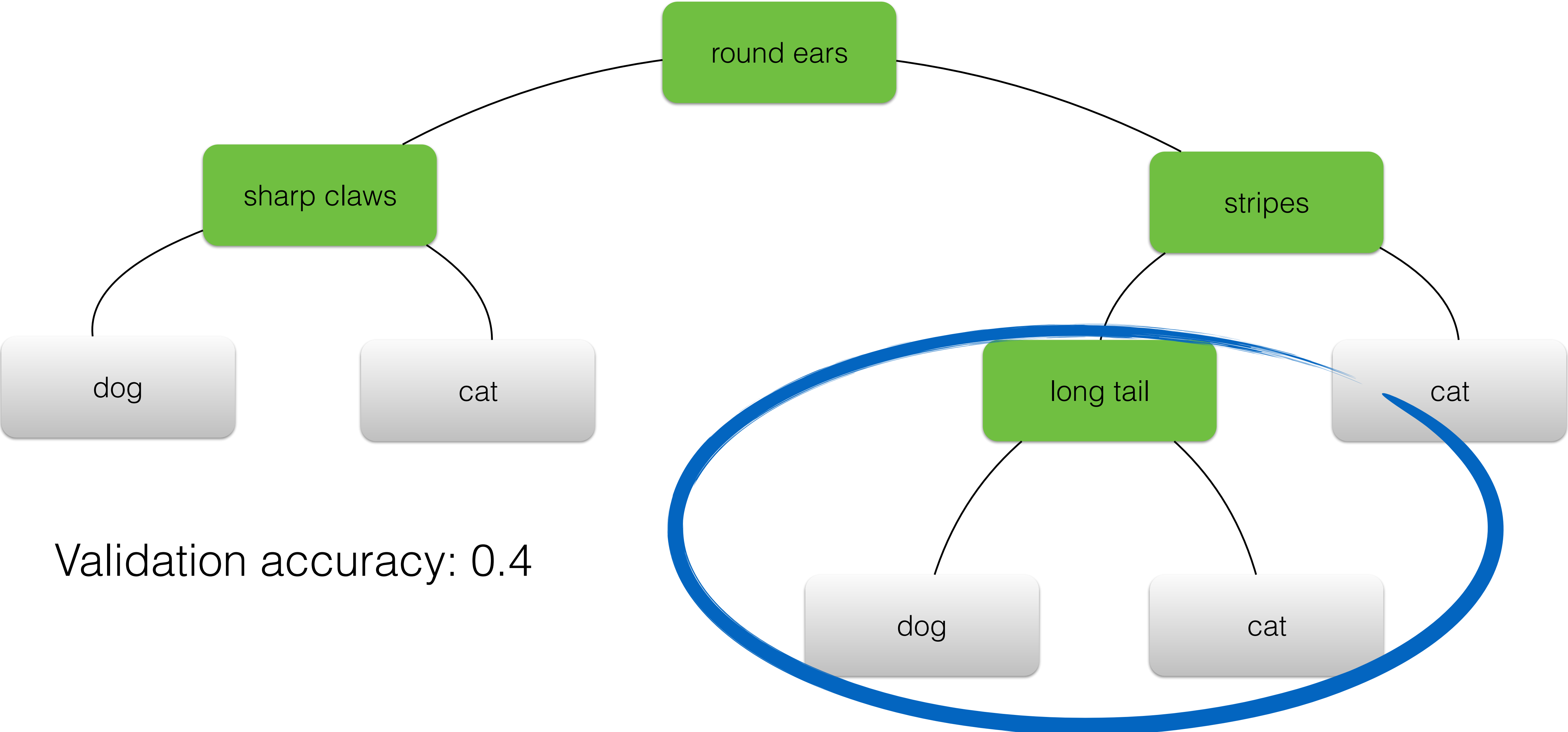
$$D_t = \{(x_1, y_1), \dots, (x_m, y_m)\}$$

$$D_v = \{(x_{m+1}, y_{m+1}), \dots, (x_n, y_n)\}$$

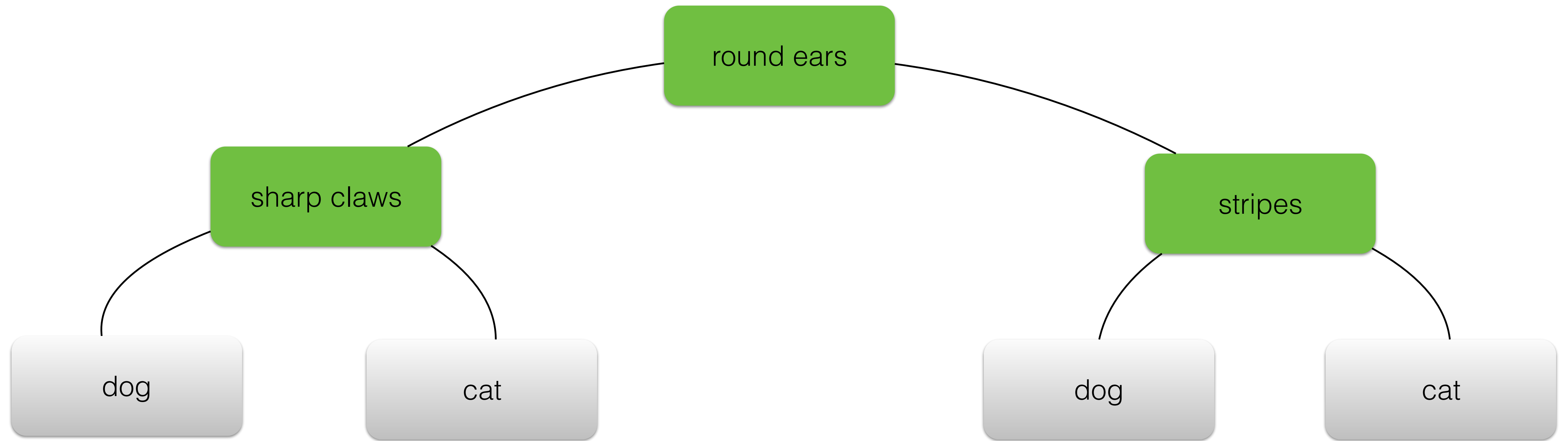
- Only allow learning algorithm to look at  $D_t$



# Pruning with Validation Set



# Pruning with Validation Set



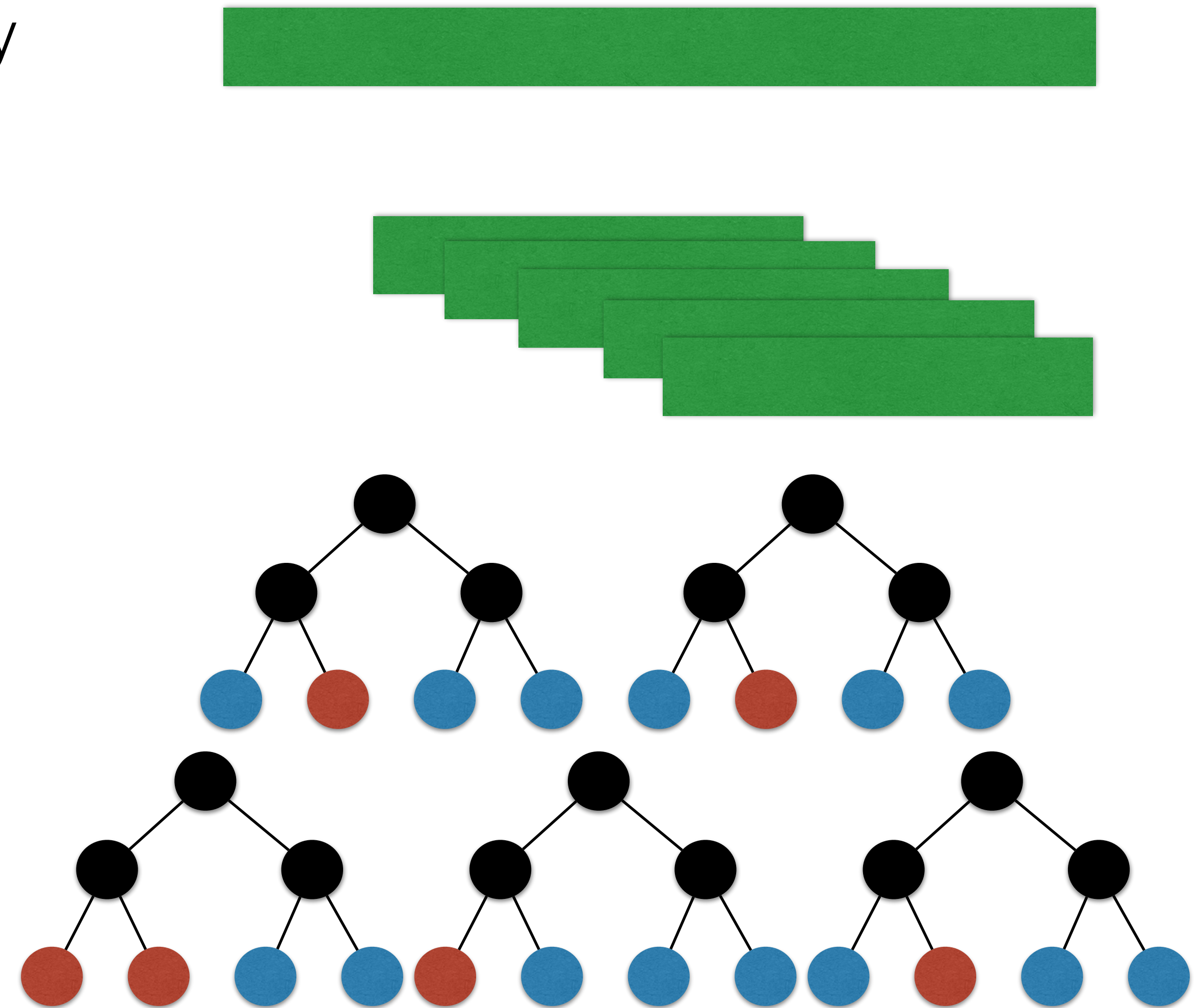
Validation accuracy: 0.4

new validation accuracy: 0.41



# Random Forests

- Use **bootstrap aggregation** to train many decision trees
- Randomly subsample **n** examples
- Train decision tree on subsample
- Use average or majority vote among learned trees as prediction
- Also randomly subsample features
- Reduces *variance* without changing *bias*



# Summary

- Training decision trees
- Cost functions
  - Misclassification
  - Entropy and information gain
  - Gini index (expected error)
- Pruning
- Random forests (bagging)