# Lecture: Learning Machines - Classification

Jeremy L Wyatt

# Aims

- See another kind of representation suitable for learning

- See another type of learning algorithm

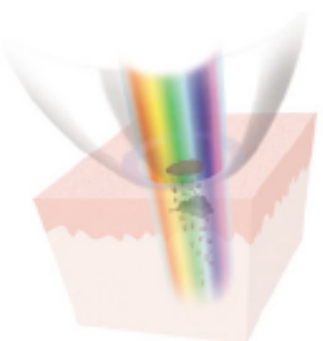
- Know how the decision tree learning algorithm works

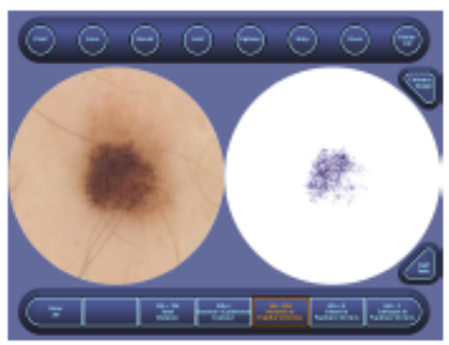
# A Guessing Game

- You think of a thing (animal,vegetable, mineral)
- I ask you some questions about it
- Eventually I guess what it is or I give up
- If I give up or get it wrong you tell me what it is
- If I got it wrong you tell me what makes it different from the sort of thing I thought it was

# A dataset

- We care about solving real world problems
- A cancer dataset
- Generated by a technique invented in this school (SIAscope)





# Dataset Structure

- Mostly a binary feature vector
- BIN_DIAG is ground truth

| Age | sex | Diameter | BIN_DIAG | COLL_BIN | BG_BIN | BL_DISP | BL_BLUS | DER_MEL |
|---|---|---|---|---|---|---|---|---|
| 63 | M | 4 | 1 | 1 | 1 | 1 | 0 | 1 |
| 47 | M | 15 | 1 | 1 | 1 | 0 | 0 | 1 |

- features above are SIAscope + interpretation, those below are normal clinical examination

| size | shape | colour | inflamed | bleed | sens | diam_7 | Total | suspicious |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

# Machine Learning for Cancer Diagnosis

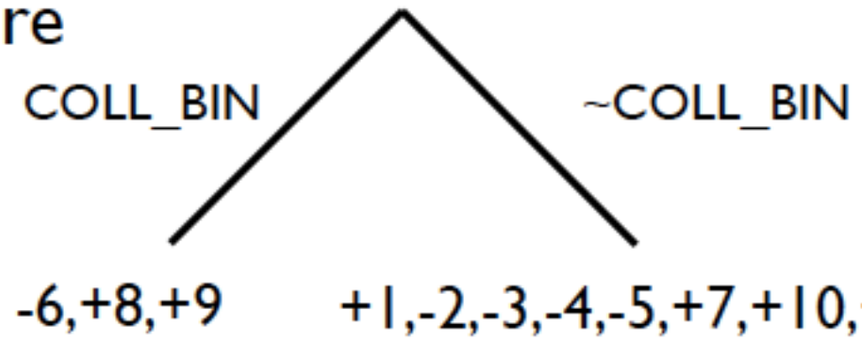
- Clearly we could use the features as inputs to a TLU, and learn the correct classification
- We could also try and learn a decision tree
- Both require that we know the ground truth, i.e. the correct classification. This makes our problem a *supervised learning* problem

# A subset of our data

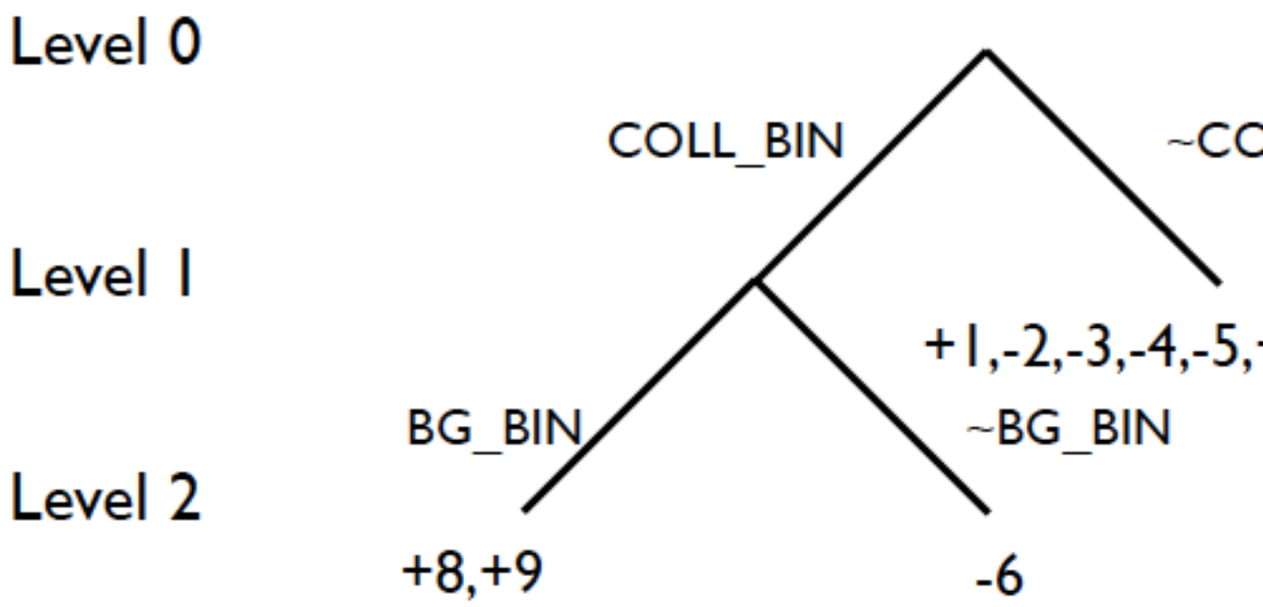
| Example | COLL_BIN | BG_BIN | BL_DISP | BL_BLUS | DER_MEL | Ground Truth |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 1 | 0 | 1 | 0 |
| 5 | 0 | 1 | 0 | 0 | 0 | 0 |
| 6 | 1 | 0 | 1 | 1 | 1 | 0 |
| 7 | 0 | 1 | 0 | 0 | 1 | 1 |
| 8 | 1 | 1 | 0 | 1 | 1 | 1 |
| 9 | 1 | 1 | 1 | 1 | 1 | 1 |
| 10 | 0 | 1 | 0 | 0 | 1 | 1 |
| 11 | 0 | 1 | 1 | 0 | 1 | 1 |

# Learning a Decision Tree

- Select a feature



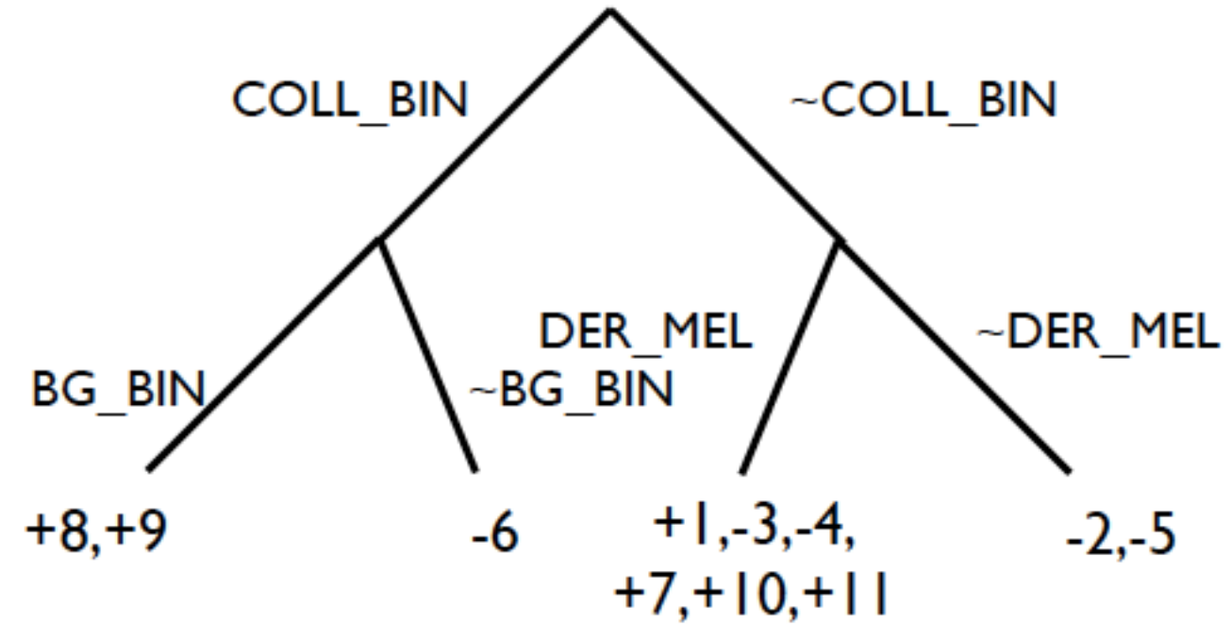

- Now repeat for the left child node, select BG_BIN




- We can now stop splitting at each of those leaf nodes

# Learning a Decision Tree

- Now we can select a feature for the right child at level 1: DER_MEL




- now the leaf classifying examples 2 and 5 is consistent
- So finally we split the remaining inconsistent node by selecting BL_BLUS

# Different features: different trees

- That was quite a small tree. If we had picked different features we could have obtained a much larger one
- Should we prefer smaller or larger trees?
- Why?
- Can we use this insight to build an algorithm for learning good DTs automatically from data?

# A simple DT learner

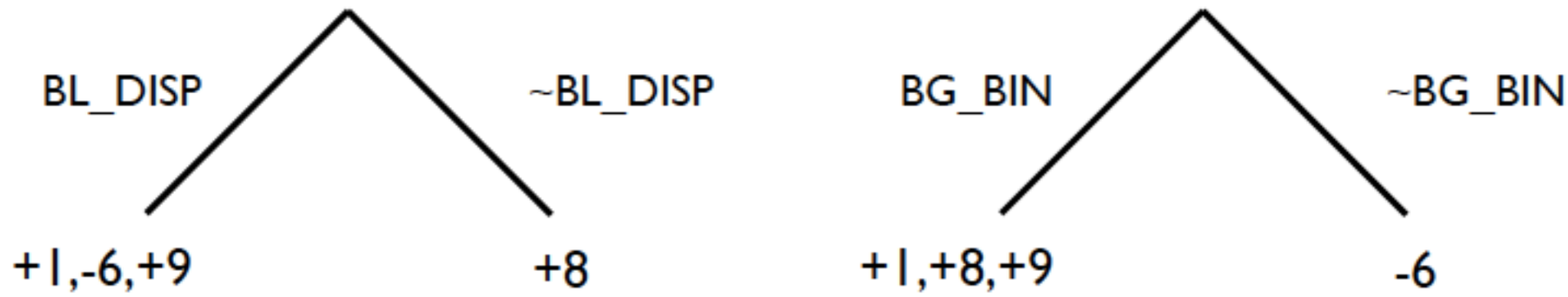
P is a training set of input-output pairs

T = learn_decision_tree(P, root_node)

1. Select a feature F with values ($v_1$, $v_2$ ... $v_n$)
2. Create a tree T with child nodes i=1...n of root node
3. partition P into subsets $P_1$...$P_n$ according the value of F
4. For each $child\_node_i$ = 1 ... n
    1. If Pi is consistent wrt to the classification then return node = (Pi, class) as the result
       else $sub\_tree_i$ = learn_decision_tree($P_i$, $child\_node_i$)
    2. $child\_node_i$ = $sub\_tree_i$
5. return T

# Selecting good features

- Intuition: select a feature which minimises the total disagreement (or uncertainty) at the child nodes about the classification




- Clearly splitting on BG_BIN is better here

- But how can we measure the degree of disagreement/ uncertainty in the resulting split?

# Probability

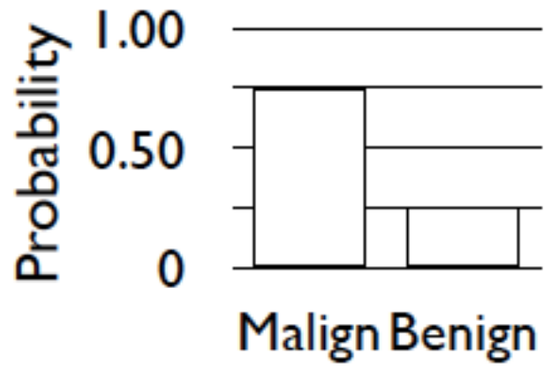
- We can express it using probabilities

- If I have example +1,-6,+8,+9 and I select one at random what is the probability that I would select a malignant example?

$$\Pr(malignant) = \frac{3}{4}$$

- A benign example?

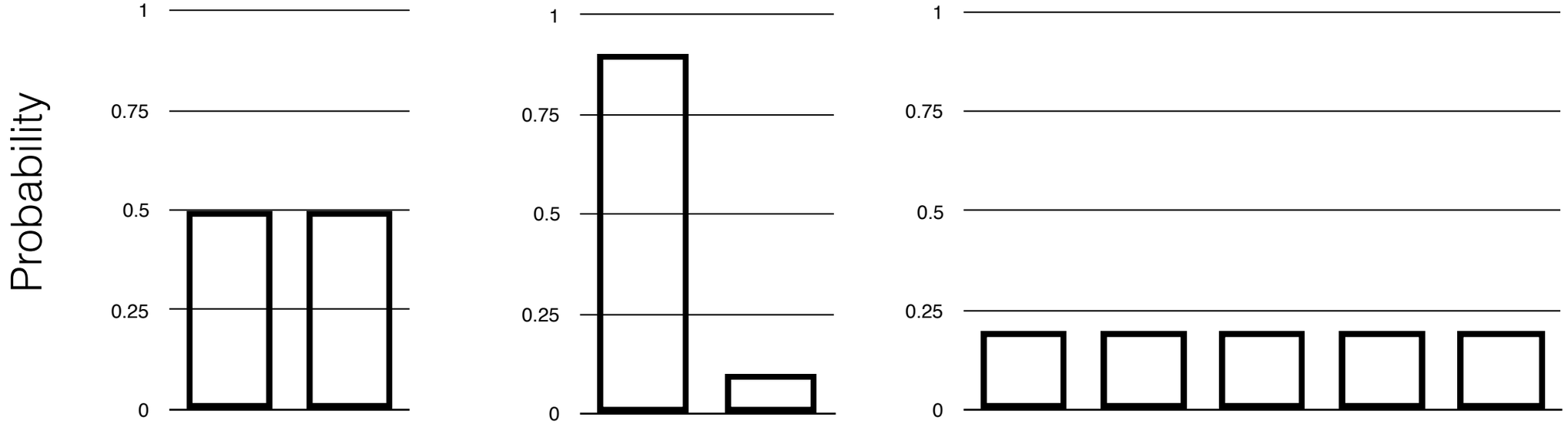
$$\Pr(benign) = \frac{1}{4}$$




- This is a probability distribution over two outcomes

# Entropy

- How do we measure the uncertainty in a probability distribution?
- Rank the three distributions below in terms of how uncertain you are about the outcome.




- Distributions with many equally likely outcomes have high entropy
- Distributions with a few highly likely outcomes have low entropy

# Entropy

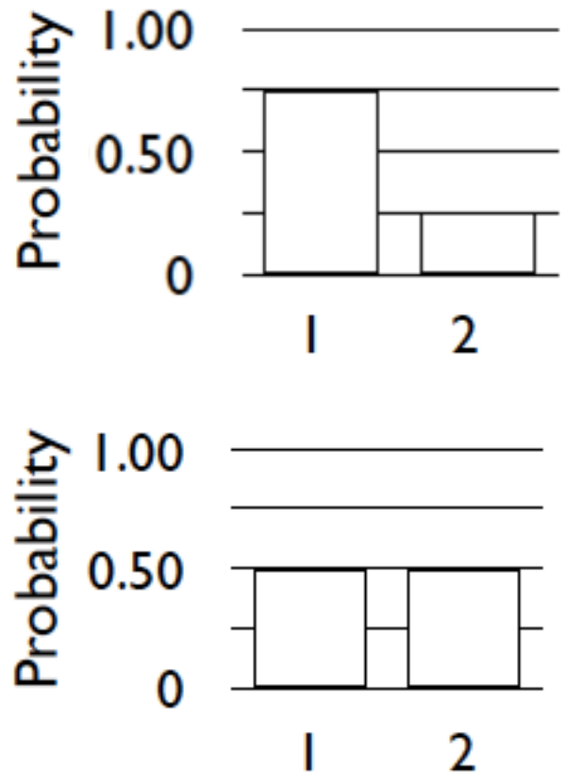
- The entropy of a probability distribution is defined as follows:

$$H = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

- where there are n outcomes and $p_i$ is the probability of the $i^{th}$ outcome

- Due to Claude Shannon, part of Information Theory

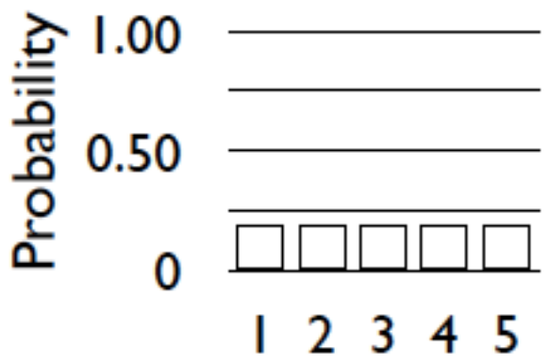
# Entropy




$$H = -\left(\frac{3}{4}\log_2\left(\frac{3}{4}\right) + \frac{1}{4}\log_2\left(\frac{1}{4}\right)\right) \simeq .811$$




$$H = -\left(\frac{1}{2}\log_2\left(\frac{1}{2}\right) + \frac{1}{2}\log_2\left(\frac{1}{2}\right)\right) = 1$$




$$H = -\sum_{i=1}^{5} \frac{1}{5}\log_2\left(\frac{1}{5}\right) \simeq 2.322$$

# Entropy

$$\log_2 \frac{1}{5} \approx -2.322 \qquad \log_2 \frac{1}{4} = -2$$

$$\log_2 \frac{1}{3} \approx -1.585 \qquad \log_2 \frac{1}{2} = -1$$

$$\log_2 \frac{2}{3} \approx -0.585 \qquad \log_2 \frac{3}{4} \approx -0.415$$

# Entropy reduction for Decision Trees

- Let's pick the feature which reduces the entropy most

  +1,-6,+8,+9  entropy=0.811

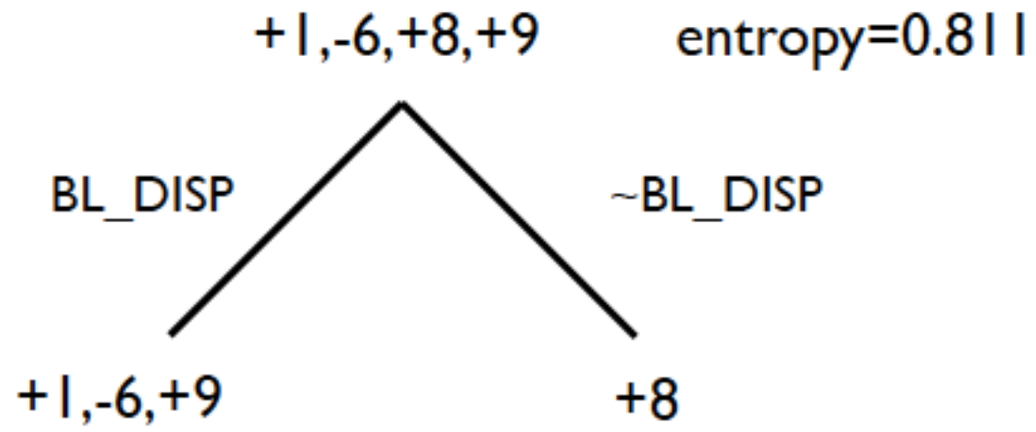
- entropy before splitting is

$$H = -\left(\frac{1}{4}\log_2\frac{1}{4} + \frac{3}{4}\log_2\frac{3}{4}\right) \simeq 0.811$$

- Suppose we split using BL_DISP, what is the total entropy now?

# Entropy reduction for Decision Trees

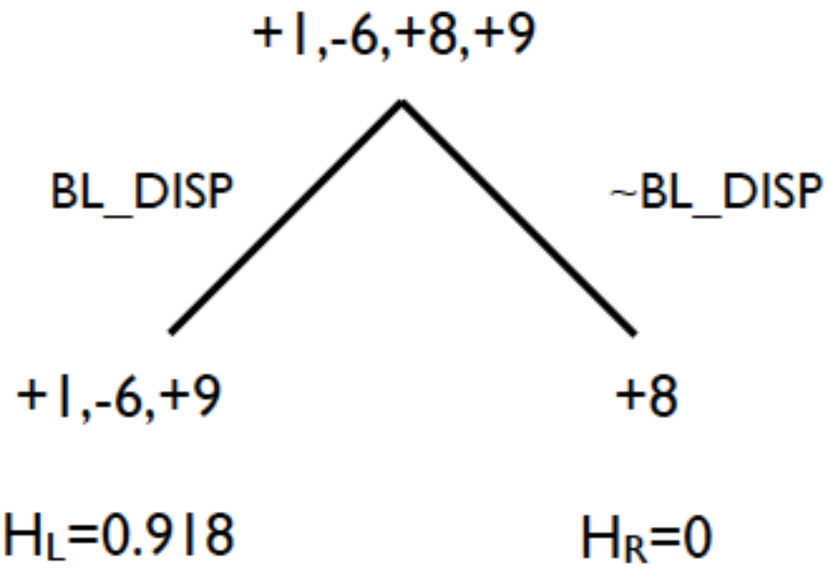
- Suppose we split using BL_DISP, what is the total entropy now?




- Entropy of the left child $H_L = -\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right) \simeq 0.918$

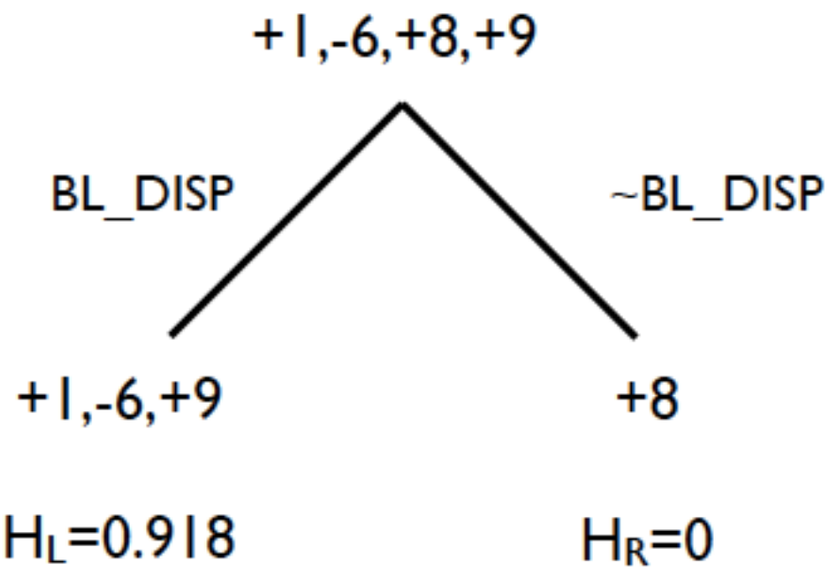
- Entropy of the right child $H_R = -(1\log_2 1 + 0\log_2 0) = 0$

# Expected Entropy




- We want to calculated the total entropy over the examples we have seen at node A given that we have split on BL_DISP
- You add the entropies of the (two) children, each weighted by the proportion of examples from the parent node that end up at that child

# Expected Entropy

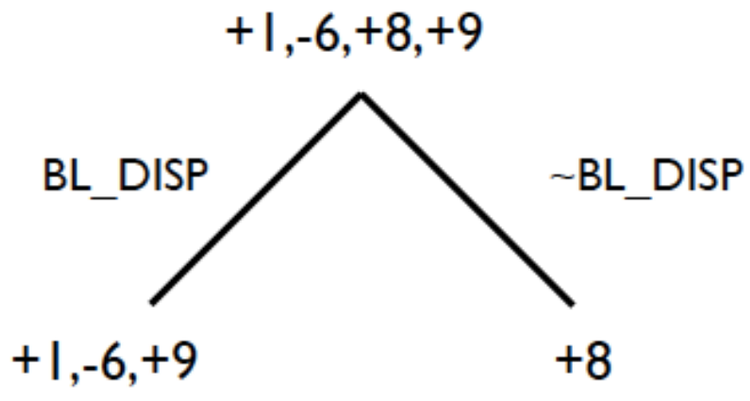



- You add the entropies of the (two) children, each weighted by the proportion of examples from the parent node that end up at that child

$$\text{total entropy} = \frac{3}{4}H_L + \frac{1}{4}H_R$$

$$\simeq \frac{3}{4}\times 0.918 + \frac{1}{4}\times 0 = 0.689$$

# Expected Entropy




$H_L$=0.918 $H_R$=0

$$E\left[H \mid node_j, f\right] = \sum_{i=1}^{n} \Pr(\text{child}_i(\text{node}_j) \mid f) \times H_i$$

$$\text{total entropy} = \frac{3}{4} H_L + \frac{1}{4} H_R$$

- Now calculate the entropy if we split on COLL_BIN

# Entropy Reduction DT learner

P is a training set of input-output pairs

T = learn_decision_tree(P, root_node)

1. Select a feature F with values ($v_1$, $v_2$ ... $v_n$)
2. Create a tree T with child nodes i=1...n of root node
3. partition P into subsets $P_1$...$P_n$ according the value of F
4. For each $child\_node_i$ = 1 ... n
    1. If Pi is consistent wrt to the classification then return
       node = (Pi, class) as the result
       else $sub\_tree_i$ = learn_decision_tree($P_i$, $child\_node_i$)
    2. $child\_node_i$ = $sub\_tree_i$
5. return T

# Entropy Based Decision Trees

- This algorithm can learn compact decision trees that generalise well

- The algorithm is not guaranteed to find the best tree however

- This is because the entropy reduction rule looks only one split ahead. It may be that sequences of locally sub-optimal splits combine to produce a better overall tree.

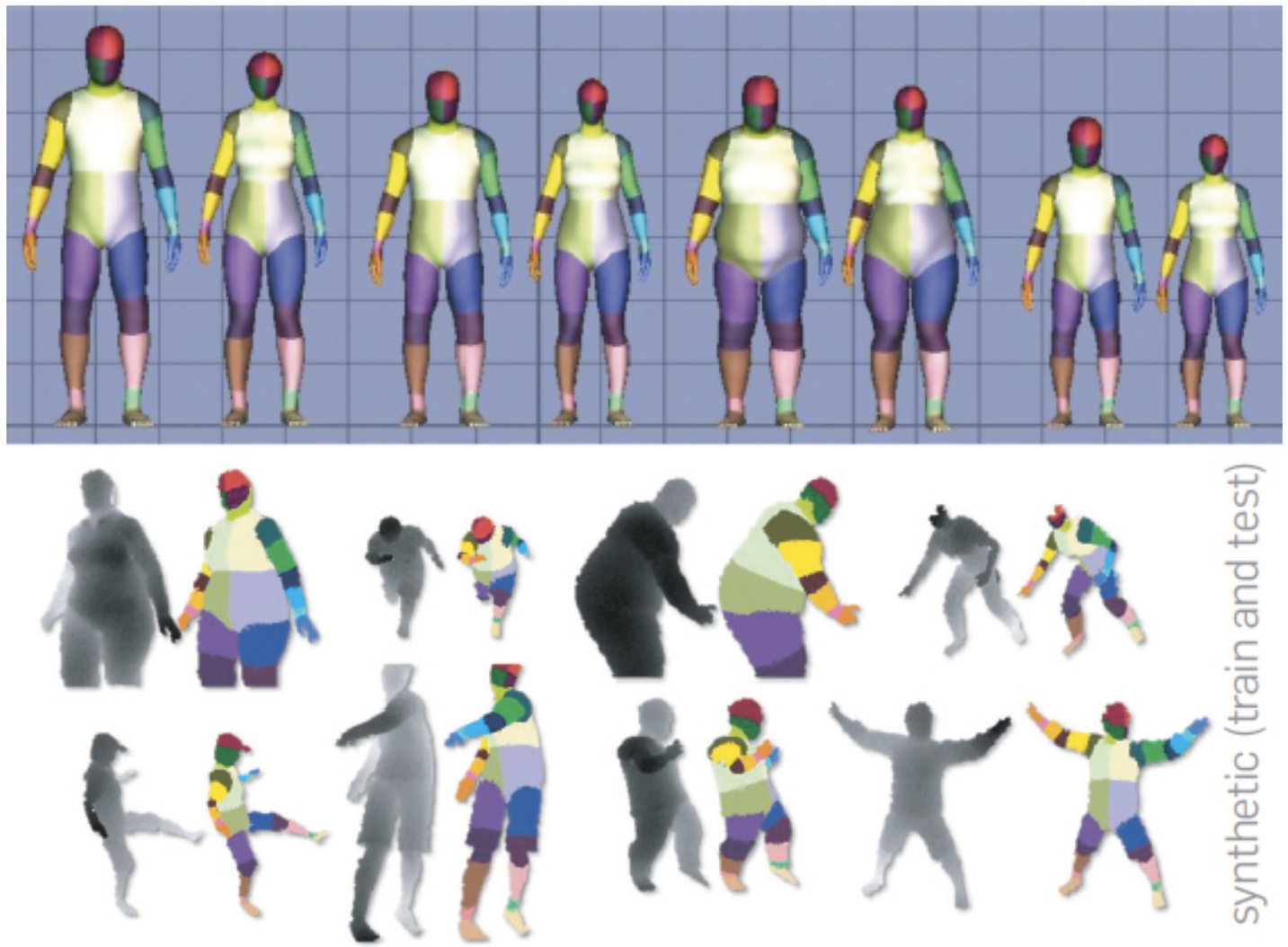
- In that sense the entropy reduction rule as applied here is an example of a greedy algorithm

# Randomised Forests

- We can extend the idea to learning many trees
- Each tree is learned on a randomised subset of the data

1. For some data set $D = \{(\vec{x}^1, t^1), (\vec{x}^2, t^2) \ldots (\vec{x}^k, t^k)\}$
2. For tree $b = 1 : B$
3. Draw a random sample $D^b$ of size $N$ from the training set
4. Build a tree by recursively applying the following steps recursively to every leaf node until some terminal criterion is met
    - 4.1. randomly pick a set of $m$ candidate features
    - 4.2. choose the best feature to split on from these $m$
    - 4.3. split the leaf node into two children
5. Output a committee of trees $\{T^b\}_{1:B}$

# Randomised Forests for Body Part Classification




- Step 1: build synthetic data set for training

# Randomised Forests for Body Part Classification

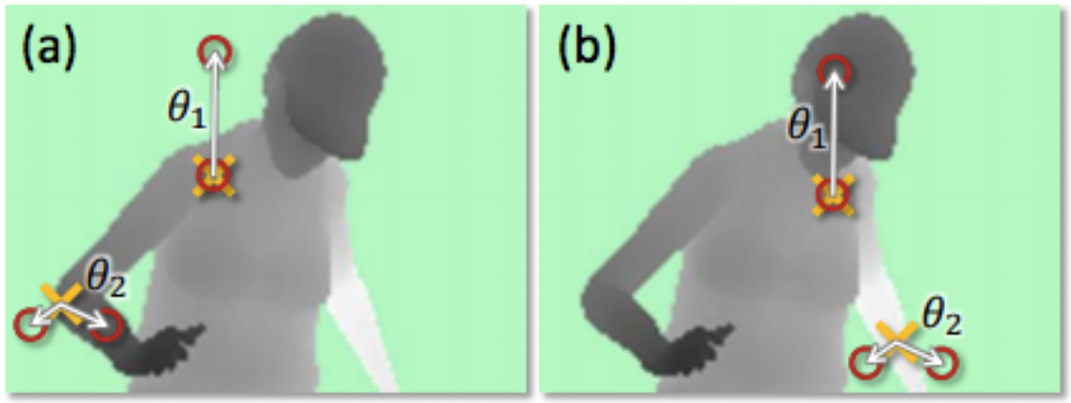



Figure 3. **Depth image features.** The yellow crosses indicates the pixel $\mathbf{x}$ being classified. The red circles indicate the offset pixels as defined in Eq. 1. In (a), the two example features give a large depth difference response. In (b), the same two features at new image locations give a much smaller response.

$\theta = (u, v)$ is a random offset

pixel from training set

$d_I(x)$ is the pixel depth

$$f_\theta(I, \mathbf{x}) = d_I\left(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}\right) - d_I\left(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}\right)$$

- Step 2: define features that have randomised parameters

# Randomised Forests for Body Part Classification

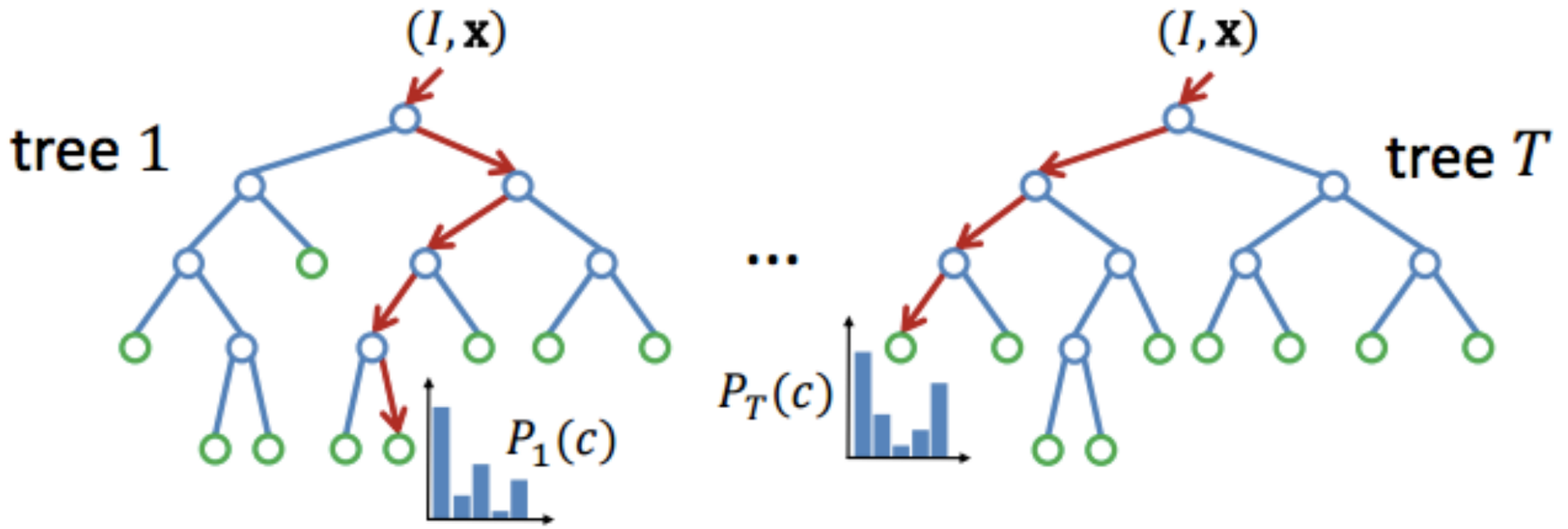
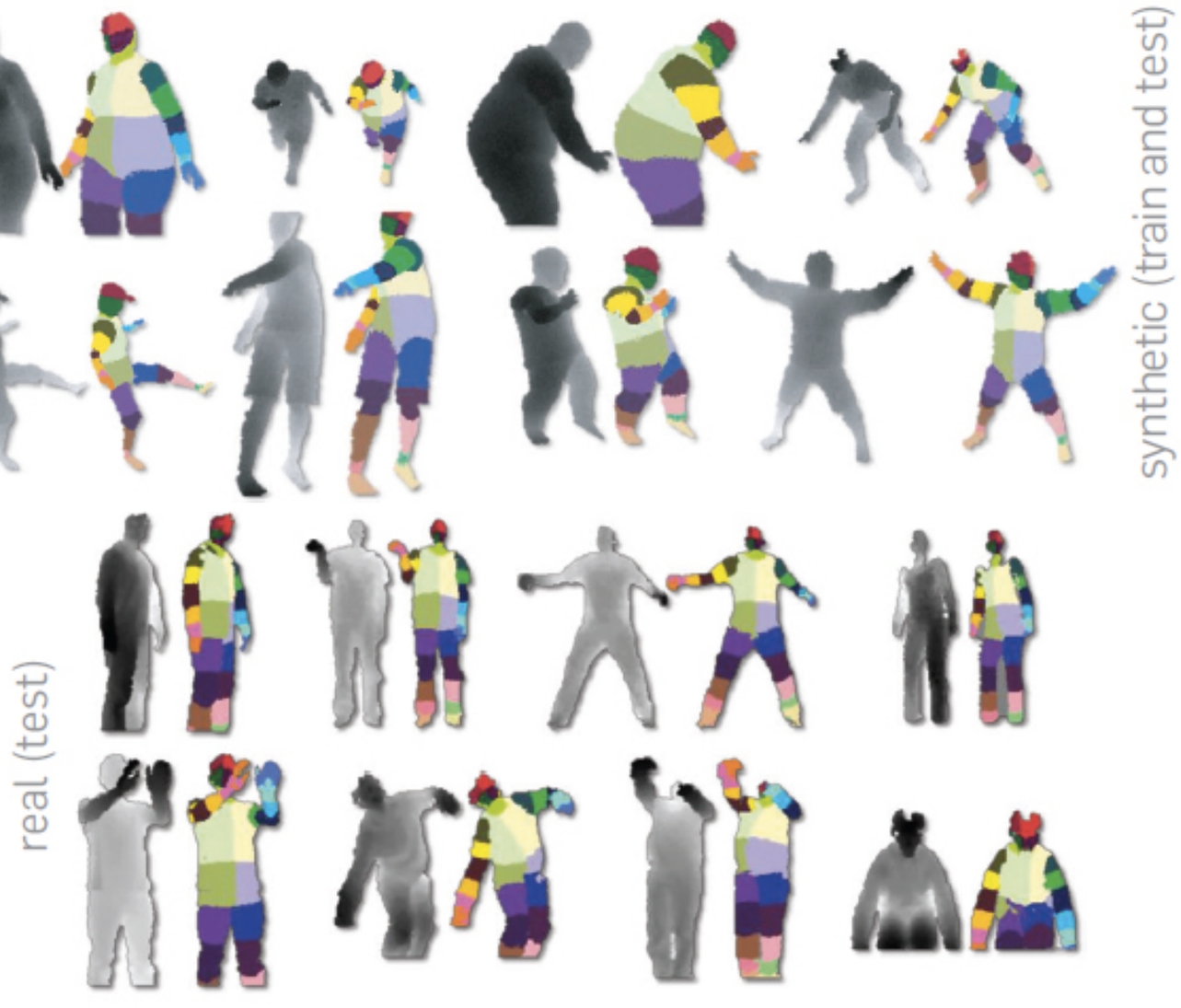



Figure 4. **Randomized Decision Forests.** A forest is an ensemble of trees. Each tree consists of split nodes (blue) and leaf nodes (green). The red arrows indicate the different paths that might be taken by different trees for a particular input.

- Step 3: learn the randomised decision forest

# Randomised Forests for Body Part Classification




- Step 4: segment real data into body parts, and suggest a whole body pose

# Summary

- We've seen a different representation that can be used for classification. There are many others to go.

- We've seen how to learn a decision tree from data using the entropy reduction principle to select the feature to split on at each node

- We've seen that an application of a related method, called randomised decision forests, used the same entropy reduction principle to produce state of the art performance in computer vision.

# Reading

- Russell and Norvig, Chapter 18, Sections 18.1-18.3