2.4
Random Forrests

SS 2019

Prof. Dr. Rainer Lienhart

www.multimedia-computing.{de,org}

References

- L. Breiman. *Random forests*. Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

- Antonio Criminisi, Jamie Shotton, Ender Konukoglu . *Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning*. In Foundations and Trends® in Computer Graphics and Vision, Vol. 7: No 2-3, pp 81-227, 2011.

➔ Figures and notation are taken from this reference

Definitions

- A *forest* is an ensemble of decision trees
- A random forest is an ensemble of random decision trees

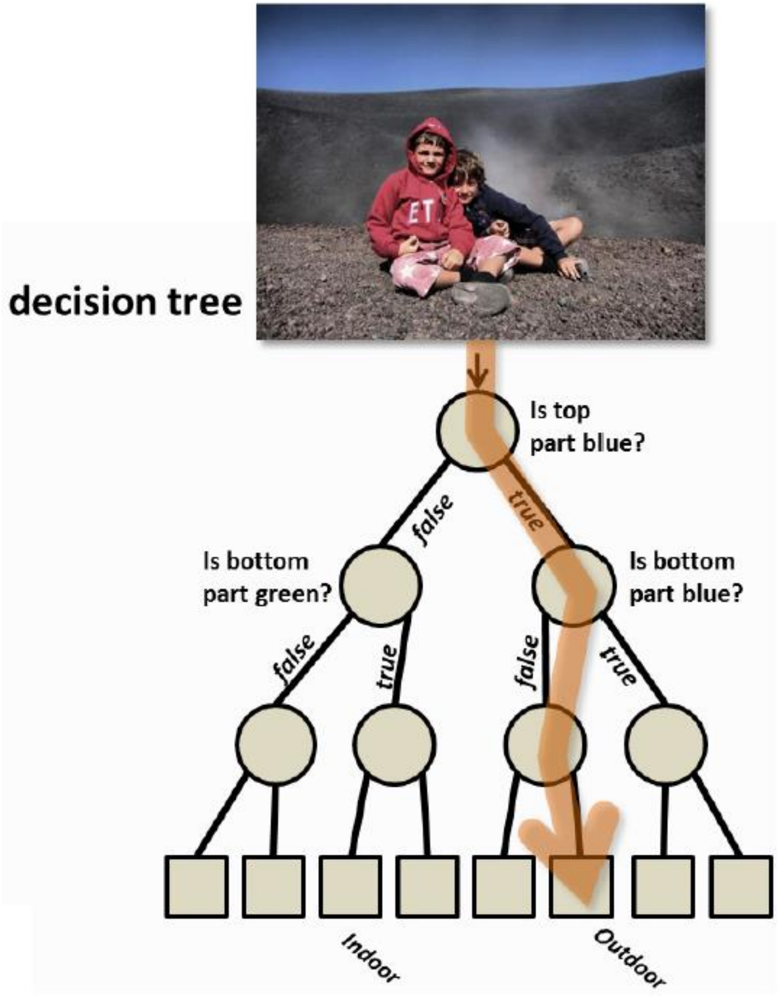
- What is a random decision tree $h(x)$?
  - E.g., Select at random at each node a small group of input variables to split on. Grow the tree with our ID3 algorithm.

- Motivation: Observation that ensembles of slightly different trees tend to produce higher accuray on previously unseen data
  ➔ better generalization capability

ReCAP – Decision Tree (1)

A decision tree is a tree where each internal node stores a split (or test) function to be applied to the incoming data.

Each leaf stores the final answer (predictor).



ReCAP – Decision Tree (2)

- A decision tree is a set of questions organized in a hierarchical manner and represented graphically as a tree.
- For a given input object, a decision tree estimates an unknown property of the object by asking successive questions about its known properties.
- Which question to ask next depends on the answer of the previous question and this relationship is represented graphically as a path through the tree which the object follows.
- The decision is then made based on the terminal node on the path.

WLOG (*without loss of generality*), we make the following assumptions for the rest of this discussion:

- Binary trees only, i.e., trees where each internal node has exactly two outgoing edges

- Instance space X equals $\mathbb{R}^n$, i.e., each instance $x \in \mathbb{R}^n$
- We call an instance also a data point.

- y represents a generic known label.

Random Decision Tree

SS 2019

Prof. Dr. Rainer Lienhart

www.multimedia-computing.{de,org}

Formal Model: Data

- Each instance $\boldsymbol{x} \in \mathbb{R}^n$. n can be very large, possibly ∞.
- $x_i, i \in \{1, \ldots, n\}$ are the features of the data point / instance $\boldsymbol{x}$. The individual features of an instance might be computed on demand (e.g., for $n = \infty$).

- For each decision in a decision node, only a subset n' of features are used
$$\Phi(\boldsymbol{x}) = \left(x_{\Phi_1}, x_{\Phi_2}, \ldots, x_{\Phi_{n'}}\right)$$
where n' is the subspace dimension and $\Phi_i \in \{1, \ldots, n\}$ denote the selected dimensions.
- Usually, $n' \ll n$, e.g., $n' = 1$ or $n' = 2$.
- $\Phi(\boldsymbol{x}) \in \mathbb{R}^{n'}$ is called the *selector.*

Formal Model: Split Functions (1)

- Terms *split function*, *test function*, and *weak learner* are used interchangeably
- Each node has associated a different test function. We formulate a test function at a split node j as a function with binary outputs:

$$h(\boldsymbol{x}, \theta_j): \mathbb{R}^n \times \boldsymbol{T} \rightarrow \{0,1\}$$

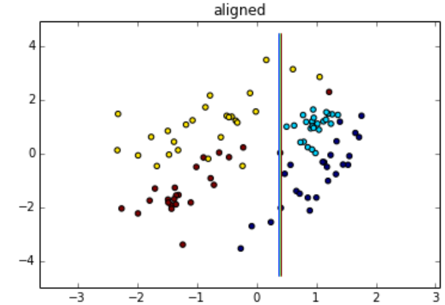
- where
  - $\boldsymbol{T}$ is the space of decision parameters
  - $\theta_j \in \boldsymbol{T}$ denote the parameters of the test function at the jth split node
  - 0 and 1 can be interpreted as “false” and “true”

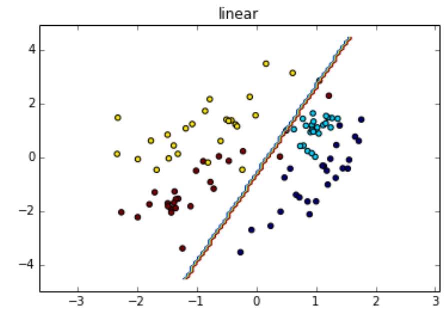
Formal Model: Split Functions (2)

Example: Threshold classifier (= default choice)

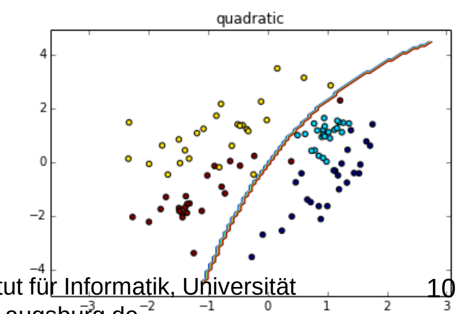
- The space of decision parameters consists of three component
  $\theta_j = (\Phi, \psi, \tau)$
  $= (selector, geometric\ primitve, threshold)$
- The *selector picks* $\Phi(\boldsymbol{x})$ a subset n' of features e.g., $n' = 1$ or $n' = 2$.
- The *geometric primitive* picks how to use the subset of features $\Phi(\boldsymbol{x})$ to compute a value that is thresholded by the threshold classifier $h(\boldsymbol{x}, \theta_j)$.
- The *threshold* τ is the threshold used by the threshold classifier $h\big(\boldsymbol{x}, \theta_j\big)$.
- $[\cdot]$ is the indicator function

$$h(\mathbf{v}, \theta) = [\tau > \Phi(\mathbf{v}) \cdot \psi]$$





$$h(\mathbf{v}, \theta) = [\tau > \Phi^T(\mathbf{v}) \cdot \psi \cdot \Phi(\mathbf{v})]$$



Formal Model: Split Functions (3)

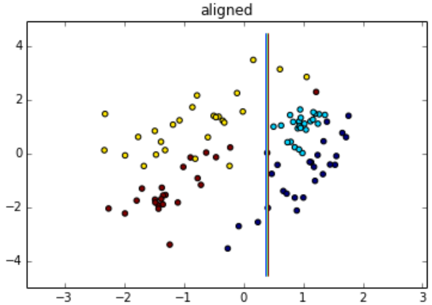
In the chapter „Decision Tree“ we use in each node j

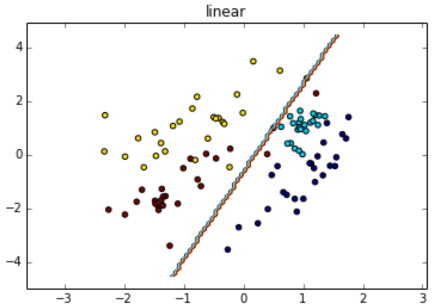
$$\theta_j = (\Phi, \psi, \tau)$$

mit

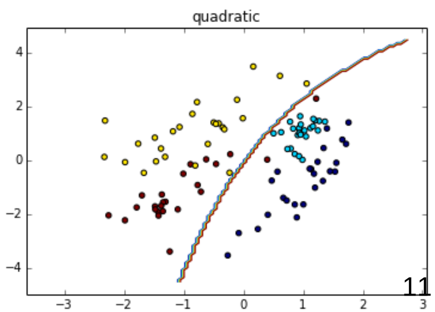
- Φ = the full feature space, i.e. $\Phi(\boldsymbol{x}) = x$
- ψ = axis-aligned hyperplane (= stump), i.e., $\psi \in \{e_1, \dots, e_n\}$ with $e_i = i$th unit vector.
- τ = the optimized threshold + direction of the unequality

$$h(\mathbf{v}, \theta) = [\tau > \Phi(\mathbf{v}) \cdot \psi]$$





$$h(\mathbf{v}, \theta) = [\tau > \Phi^T(\mathbf{v}) \cdot \psi \cdot \Phi(\mathbf{v})$$



Formal Model: Trainings Set

- Let $S_0 = \{(\boldsymbol{x}, y)\}$ denote the entire trainings set.
- S_j denote the training set reaching note j
- S_j^L, S_j^R denote the subsets going to the left and to the right children of node j, respectively

- The following equations hold:

$$S_j = S_j^L \cup S_j^R$$

$$S_j^L \cap S_j^R = \emptyset$$

$$S_j^L(S_j, \theta_j) = \{(\boldsymbol{x}, y) \in S_j \,|\, h(\boldsymbol{x}, \theta_j) = 0\}$$

$$S_j^R(S_j, \theta_j) = \{(\boldsymbol{x}, y) \in S_j \,|\, h(\boldsymbol{x}, \theta_j) = 1\}$$

Formal Model: Training (1)

An objective function I must be selected for optimizing deciders

$$\theta_j = arg \max_{\theta \in T} I(S_j, \theta)$$

A standard choice is the *information gain*

$$I = H(S) - \sum_{i \in \{L,R\}} \frac{|S^i|}{|S|} H(S^i)$$

with Shannon entropy

$$H(S) = -\sum_{c \in C} p(c) \cdot \log(p(c))$$

Formal Model: Training (2)

➔ We pick the function $h(\boldsymbol{x}, \theta_j)$ that best splits S_j in S_j^L and S_j^R.

At the end of the training phase we obtain:

- the (greedily) optimum weak learners associated with each node,
- a learned tree structure, and
- (a different set of training points at each leaf

Note: $p(c) \coloneqq p(c|S)$ is calculated as the normalized empirical histogram of the label corresponding to the training point in S.

Formal Model: Leaf Predicition

For each leaf node j, a model must be create from the training set S_j arriving at the node.

The usual choise is to model $p(c|\boldsymbol{x})$ for discrete, $p(y|\boldsymbol{x})$ for continuous variables

Alternatively, one could model

$$c^* = arg \max_c p(c|\boldsymbol{x})$$

or

$$y^* = arg \max_y p(y|\boldsymbol{x})$$

to get the MAP estimates.

Formal Model: Randomness (1)

Two popular ways to inject randomness during training:

- Random training set sampling
- Radomized node optimization

➔ We focus on the latter

Radomized node optimization:

$$\theta_j = arg \max_{\theta \in T_j \subset T} I(S_j, \theta)$$

where $T_j \subset T$. T_j is randomly picked. Thus $|T| = \infty$ is now possible.

Measure of degree of determinism as $\frac{|T_j|}{|T|} \in (0; 1]$

Formal Model: Randomness (2)

We define $\rho = |T_j|$.

ρ is usually set the same for all nodes of a tree.

When $\rho = 1$, each split node takes only a single randomly chosen set of values for ist parameter θ_j. There is no optimization at all.

When $\rho = |T|$, we have full optimization and no randomness at all.

Example with $\rho = 2$ and stump-based threshold classifier:

$T_j = \emptyset$

for i in range(2):

Return a random integer N such that 1 <= N <= n

$\Phi_i =$ random.randint(1,n)

$\psi = 1$

Return a random floating point number N such that a <= N <= b for a <= b and b <= N <= a for b < a.

a=min(x_{Φ_i}), b=max(x_{Φ_i}) over training data.

τ = random.uniform(min(x_{Φ_i}), max(x_{Φ_i}))

$T_j = T_j \cup (\Phi_i, \psi, \tau)$

Random Decision Forest

SS 2019

Prof. Dr. Rainer Lienhart

www.multimedia-computing.{de,org}

Decision Forest (1)

A *random decision forest* is an ensemble of randomly trained decision trees

- Key aspect: Its component trees are randomly different from each other

➢ Leads to decorrelation between the individual tree predictions

➢ Improves generalization and robustness

The value of the randomness parameter $\rho = |T_j|$ is set a priori before training and the same for all trees in the forest.

Decision Forest - Example

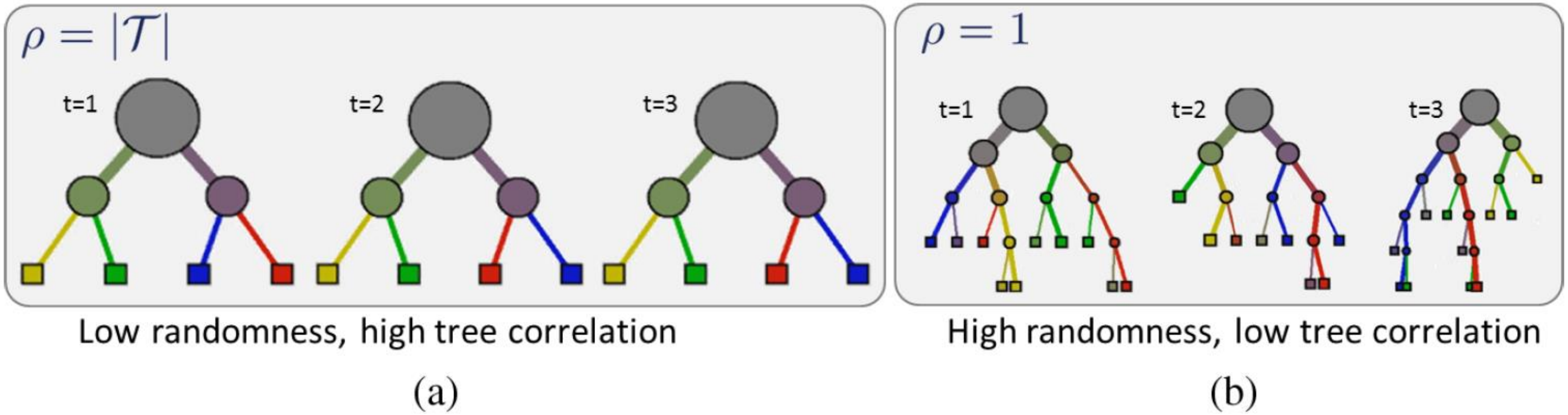


Fig. 2.7 Controlling the amount of randomness and tree correlation. (a) Large values of ρ correspond to little randomness and thus large tree correlation. In this case the forest behaves very much as if it was made of a single tree. (b) Small values of ρ correspond to large randomness in the training process. Thus the forest component trees are all very different from one another.

Decision Forest (2)

- In a forest with T trees we use the variable $t \in \{1, \dots, T\}$ to index each component tree.
- Training:
  - All trees are trained independently (→ trivial to parallelize)
- Testing:
  - Each test instance $\boldsymbol{x}$ is pushed through all trees (→ trivial to parallelize).
  - All tree predictions can be combined into a single forest prediction by simple averaging:

$$p(c|\boldsymbol{x}) = \frac{1}{T}\sum_{t=1}^{T} p_t(c|\boldsymbol{x})$$

  - Where $p_t(c|\boldsymbol{x})$ denotes the posterior distribution obtained by the tth tree.

Decision Forest – Key Parameters

- The maximum allowed tree depth D
- The amount of randomness (controlled by ρ) and its type
- The forest size (number of trees) T
- The choice of weak learner model
- The training objective function
- The choice of features in practical applications

Those choices directly affect the forest predictive accuracy, the *accuracy of its confidence*, its generalization and its computational efficiency.

Decision Forest – Observations

Several papers have pointed out

- how the testing accuracy increases monotonically with the forest size T
- how learning very deep trees can lead to overfitting
- the importance of using very large amounts of training data
- the importance of randomness and its effect on tree decorrelation
- how the choice of randomness model directly influences a classification forest's generalization

Example 1

Training objective function: Information gain

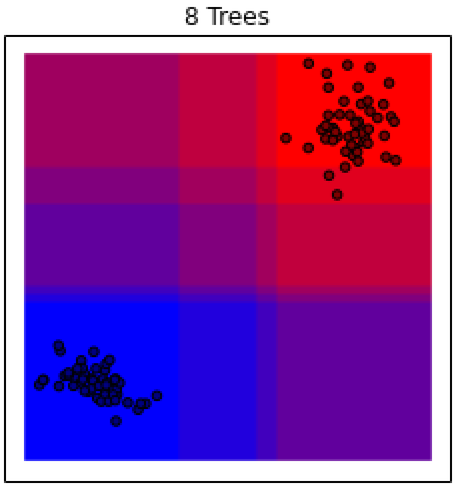
Leaf model: $p(c|\boldsymbol{x})$ by averaging

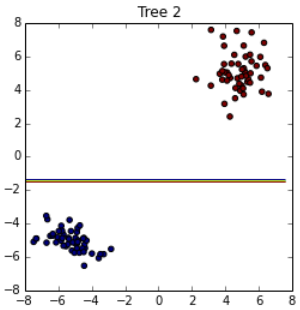
Decider: linear, aligned

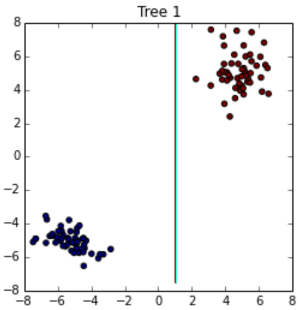
Depth: 1

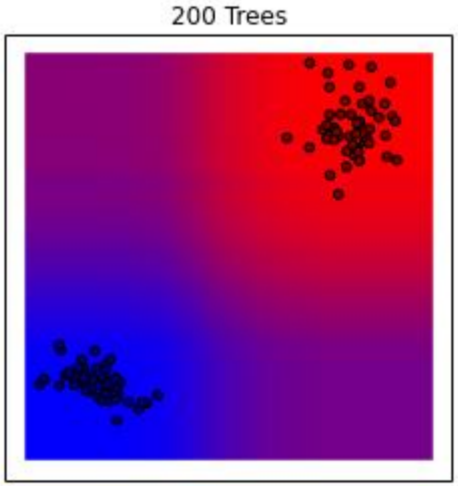
Randomized optimization: $\rho = 4$

→ 2 axes, 2 thresholds per split









Example 2

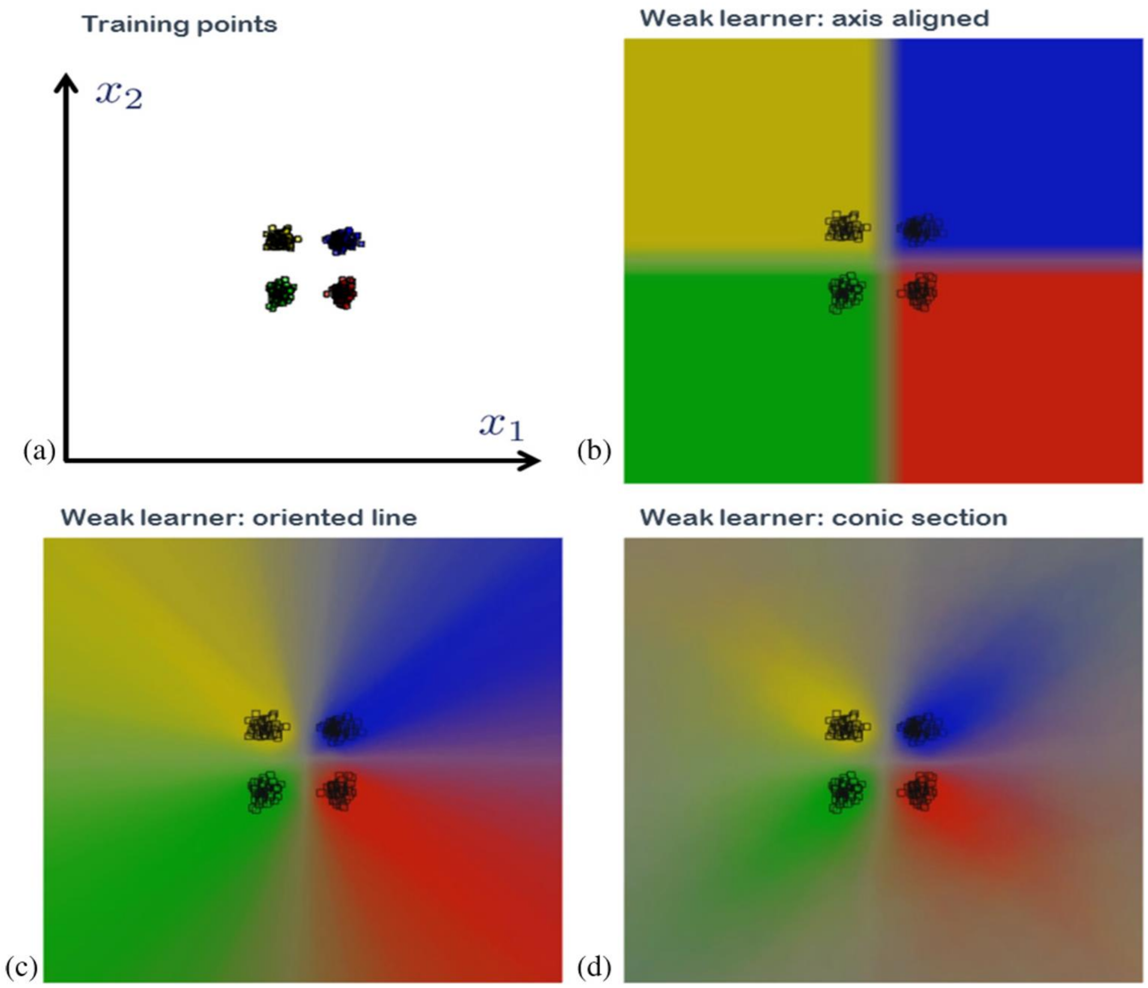
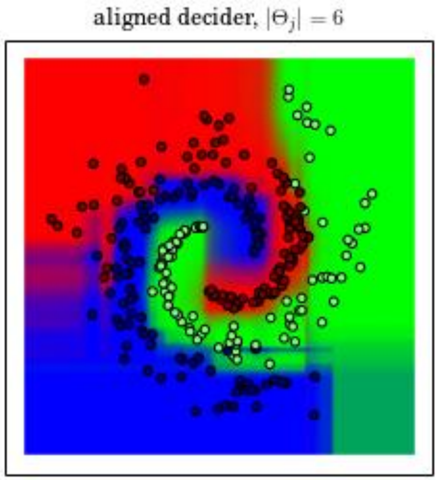


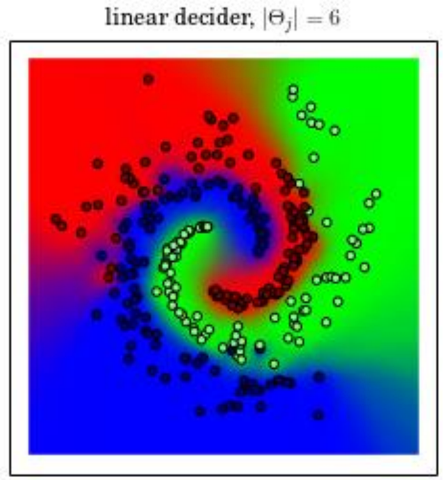
Fig. 3.6 The effect of the weak learner model. (a) A four-class training set. (b) The testing posterior for a forest with axis-aligned weak learners. In regions far from the training points the posterior is overconfident. (c) The testing posterior for a forest with oriented line weak learners. (d) The testing posterior for a forest with conic section weak learners. In (c) and (d) the uncertainty of class prediction increases with distance from the training data points. Here we use D = 3 and T = 200 for all examples.

Example 3

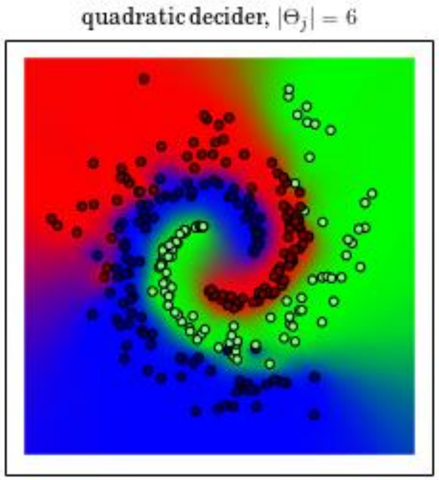
$|theta| = 6$

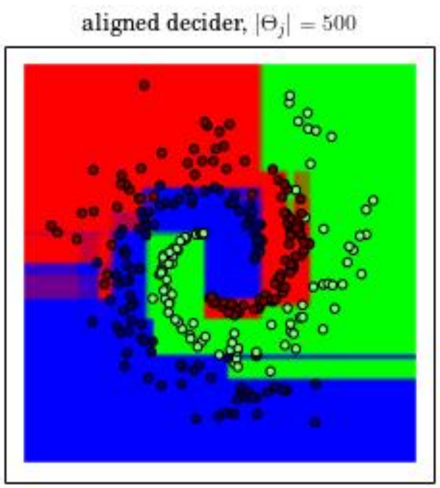


$|theta| = 6$

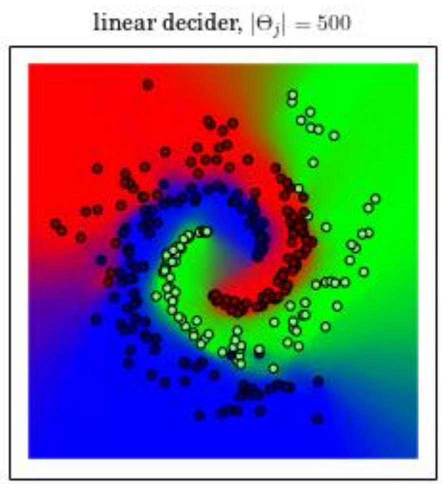


$|theta| = 6$

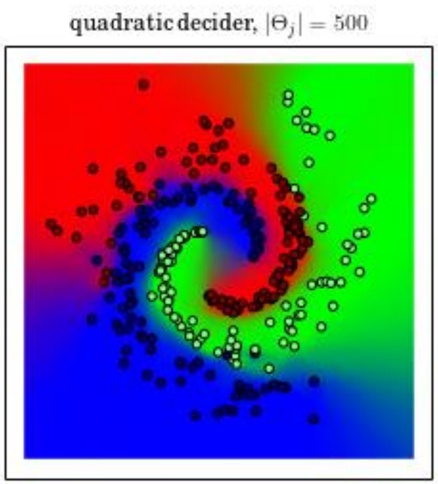




$|theta| = 500$



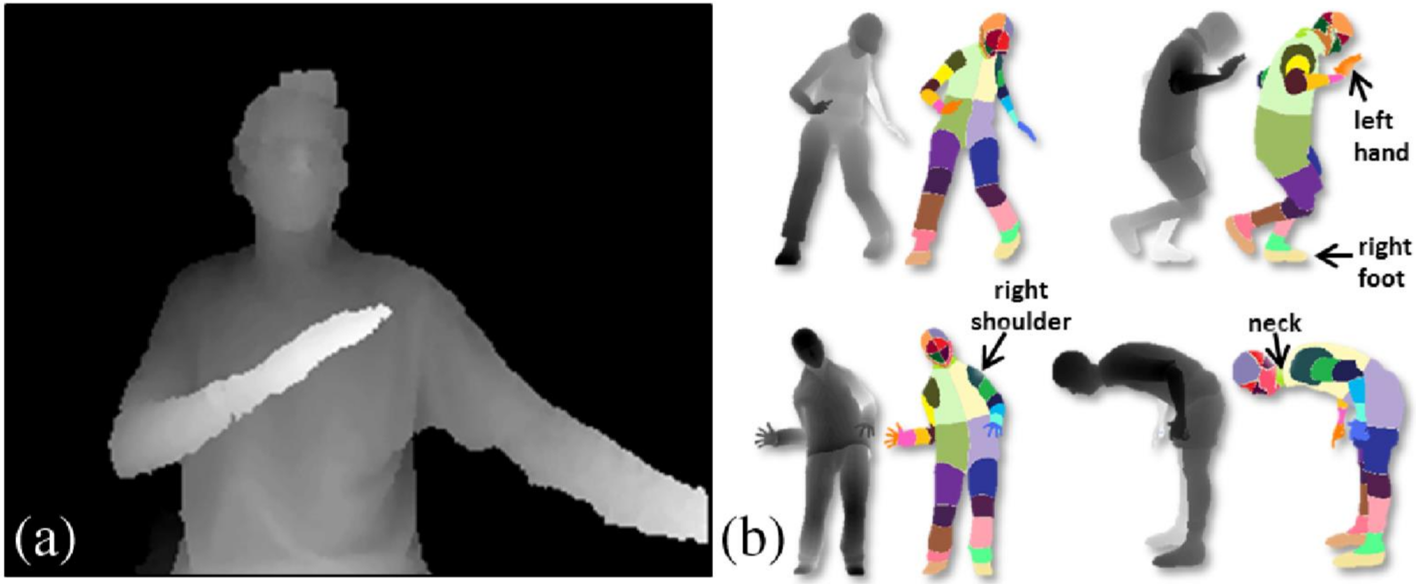
$|theta| = 500$



$|theta| = 500$

Example 4 (1)

Fig. 3.16 Classification forests in Microsoft Kinect for XBox 360. (a) An input frame as acquired by the Kinect depth camera. (b) Synthetically generated ground-truth labeling of 31 different body parts.



Example 5 (2)

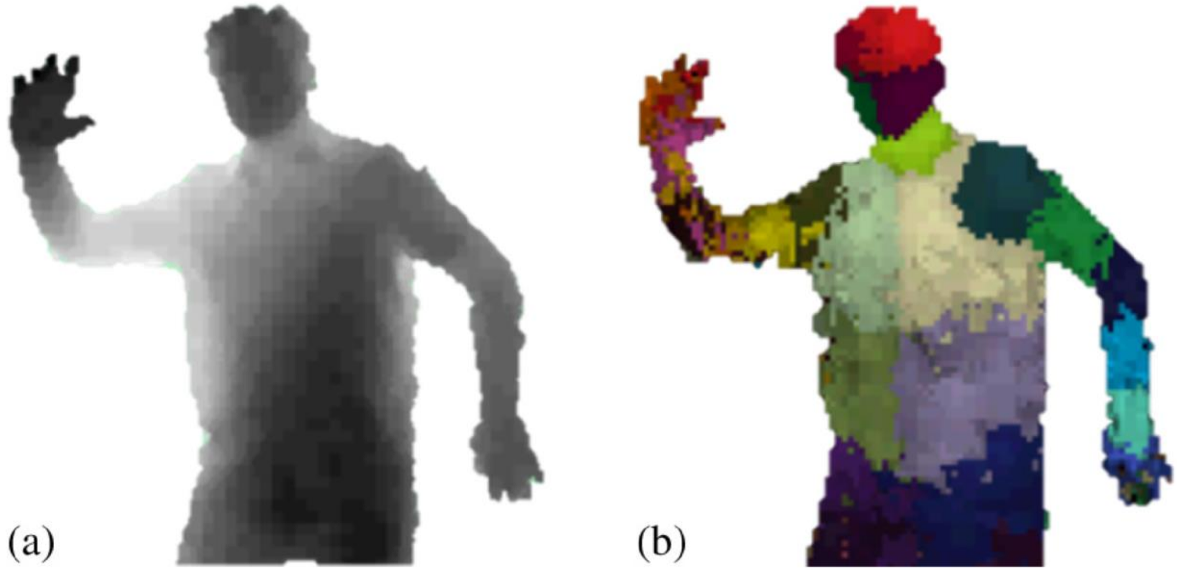


Fig. 3.17 Classification forests in Kinect for XBox 360. (a) An input depth frame with background removed. (b) The body part classification posterior. Different colors corresponding to different body parts, out of 31 different classes.