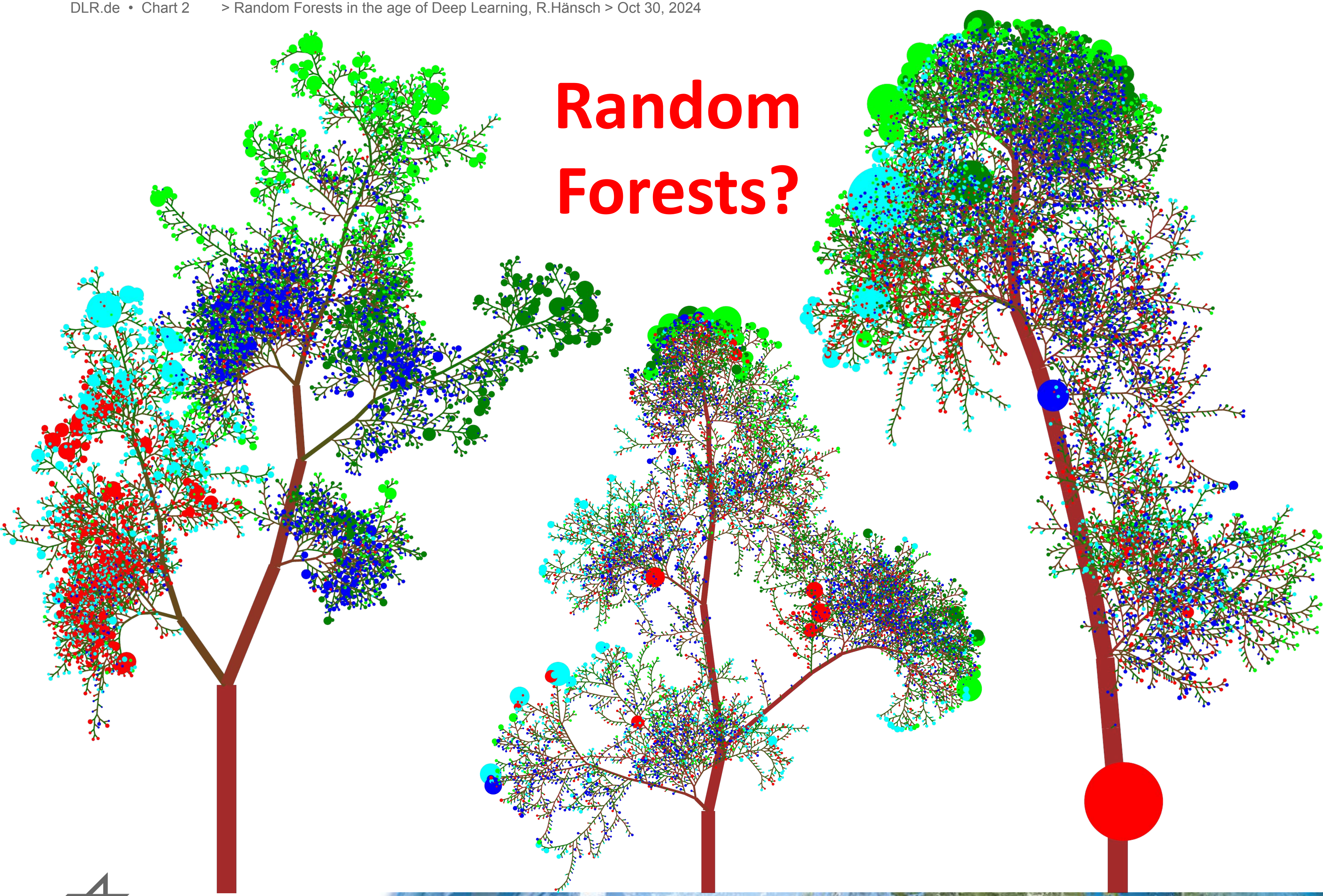
Shallow learners are dead – Long live shallow learners!

Random Forests in the age of Deep Learning

Ronny Hänsch

DLR

Random Forests?



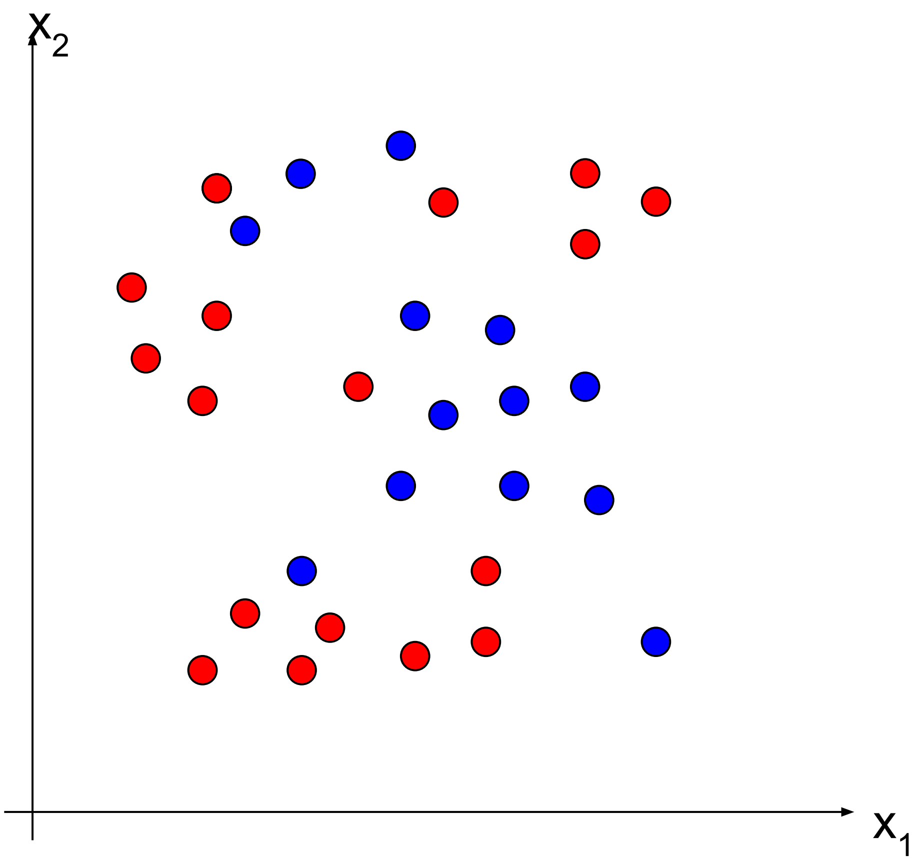
DLR

From kNN to Search Trees

- Data samples x
  ⇒ Pixel information, image patch, feature vector, etc.
  ⇒ Often $x \in R^n$

- Classification:
  ⇒ Estimate class label

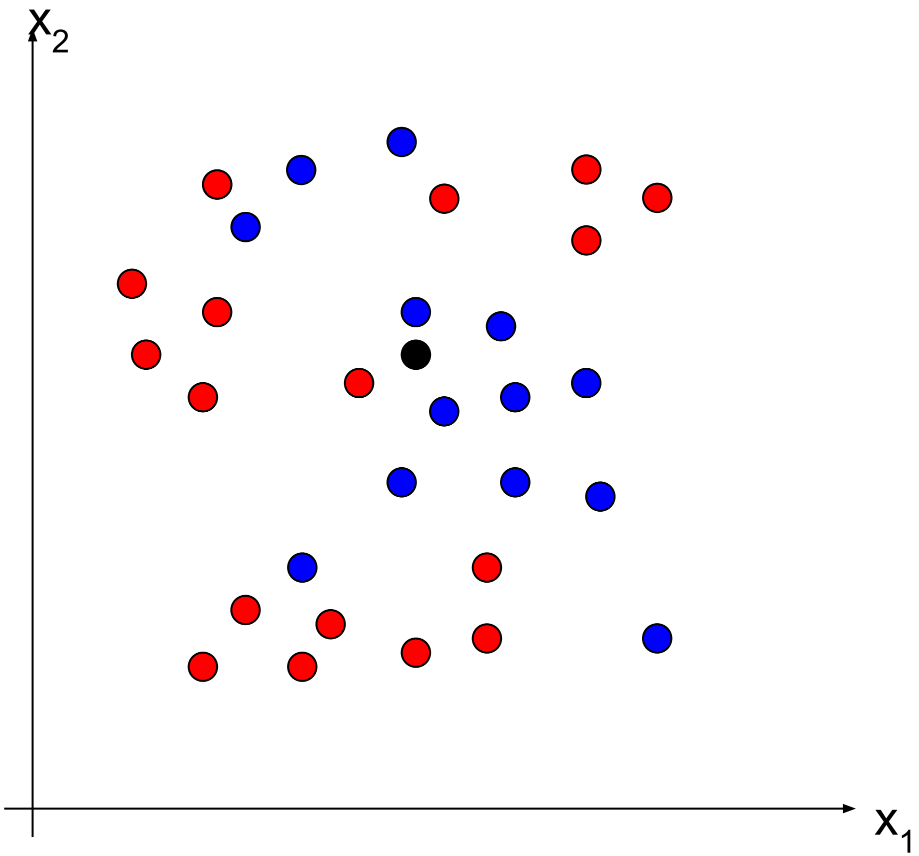
- Training data: Values of target variable given e.g. class label



DLR

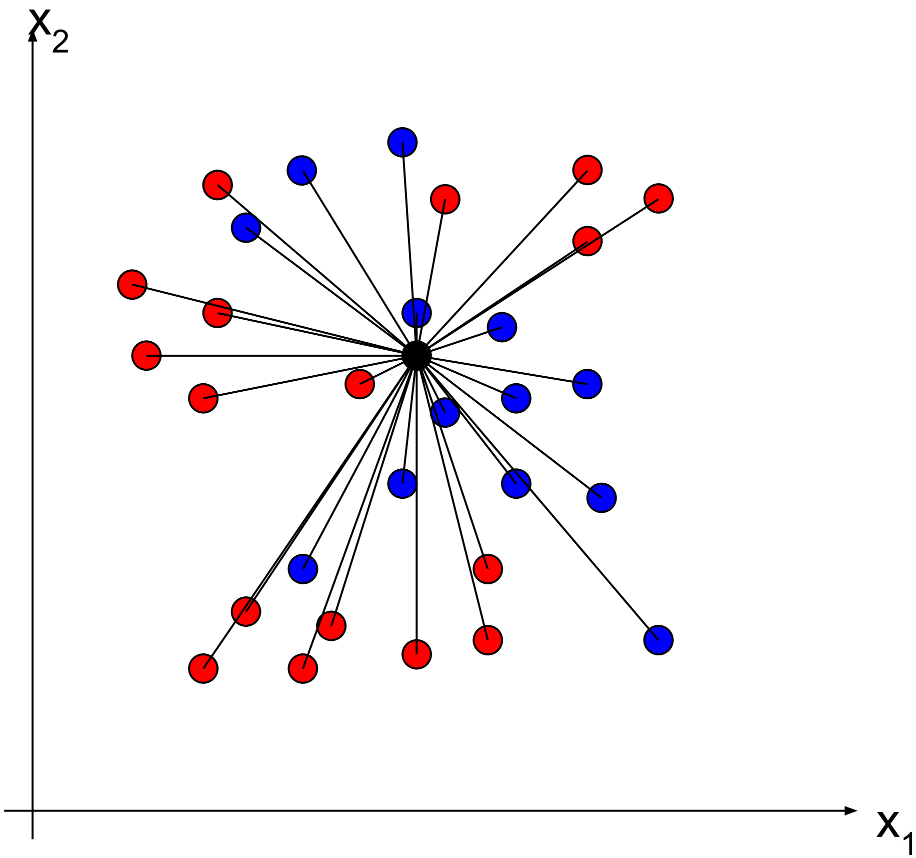
From kNN to Search Trees

- Task: Given training data, estimate label of query sample



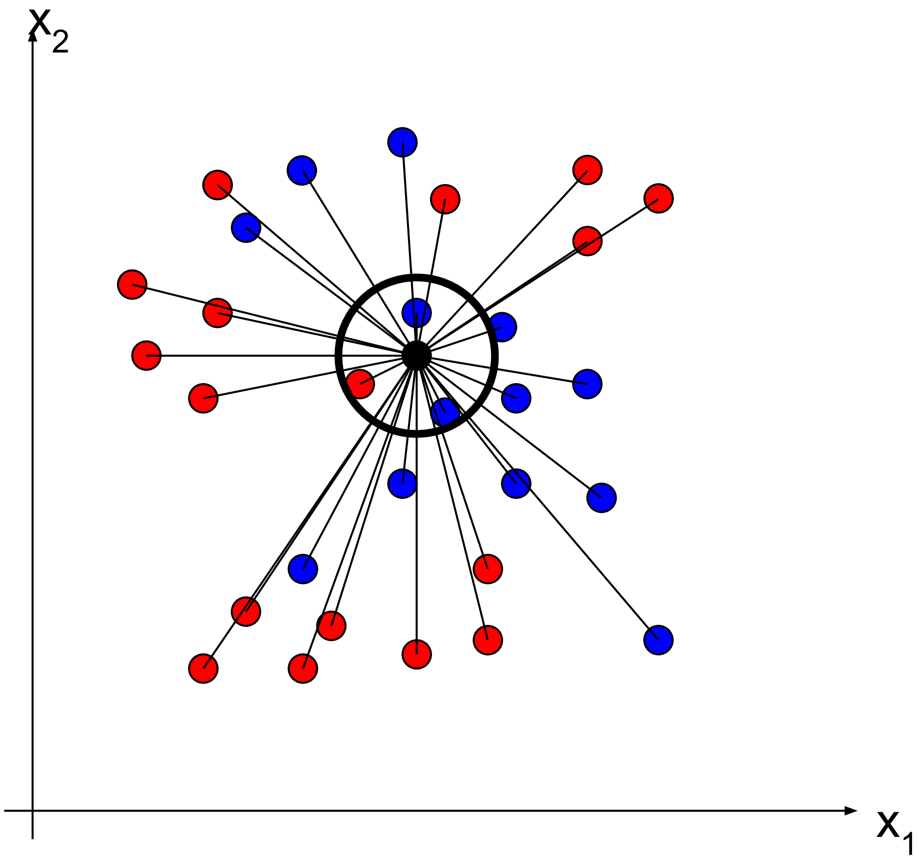
From kNN to Search Trees

- Task: Given training data, estimate label of query sample
- kNN/Parzen Window:
  → Compute distance to **all** samples



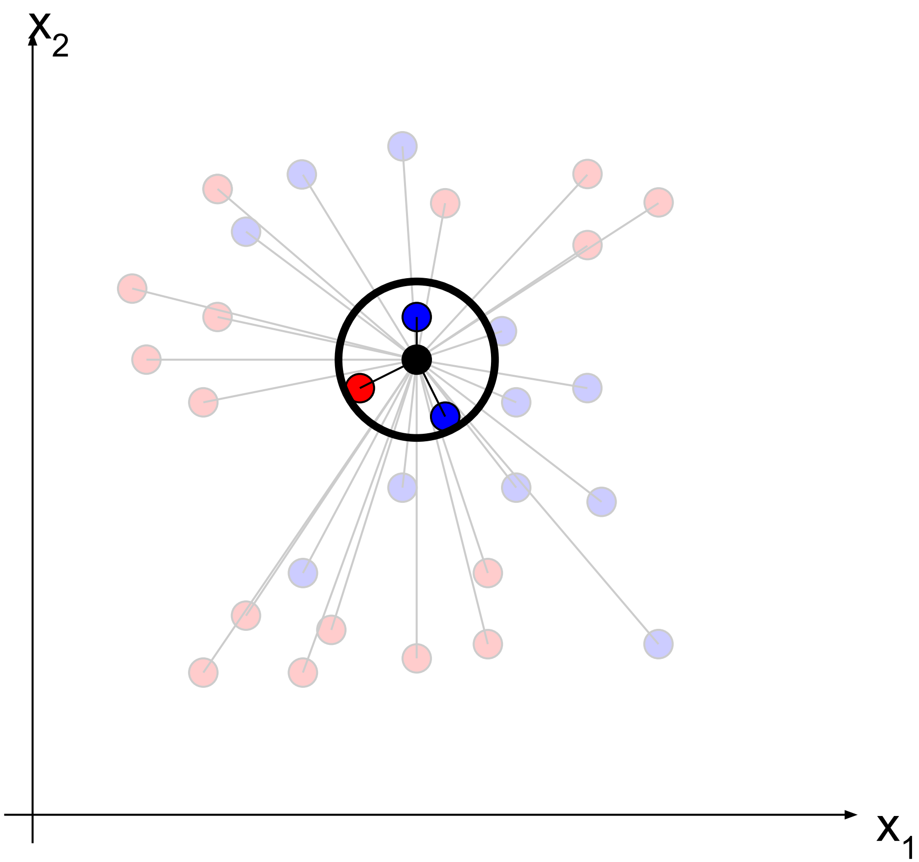
From kNN to Search Trees

- Task: Given training data, estimate label of query sample
- kNN/Parzen Window:
  → Compute distance to **all** samples
  → Select samples within window of given size (Parzen)



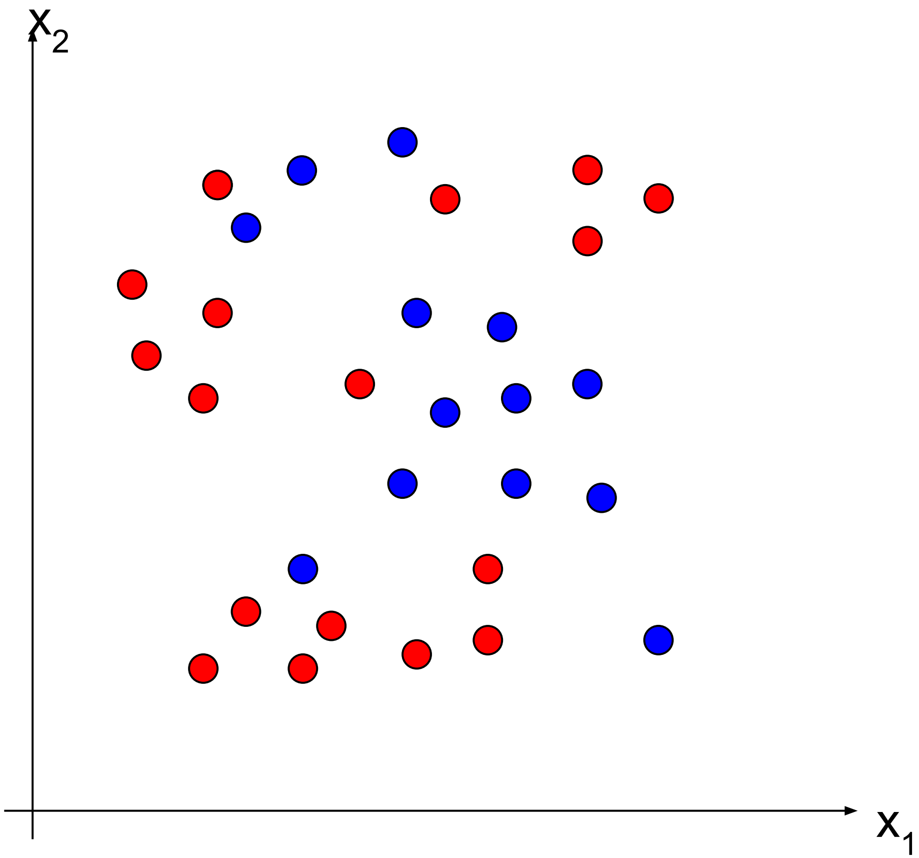
From kNN to Search Trees

- Task: Given training data, estimate label of query sample
- kNN/Parzen Window:
  → Compute distance to all samples
  → Select samples within window of given size (Parzen)
  → Use these samples to estimate target variable, e.g. class label
- Problem: Computationally expensive (exhaustive search)



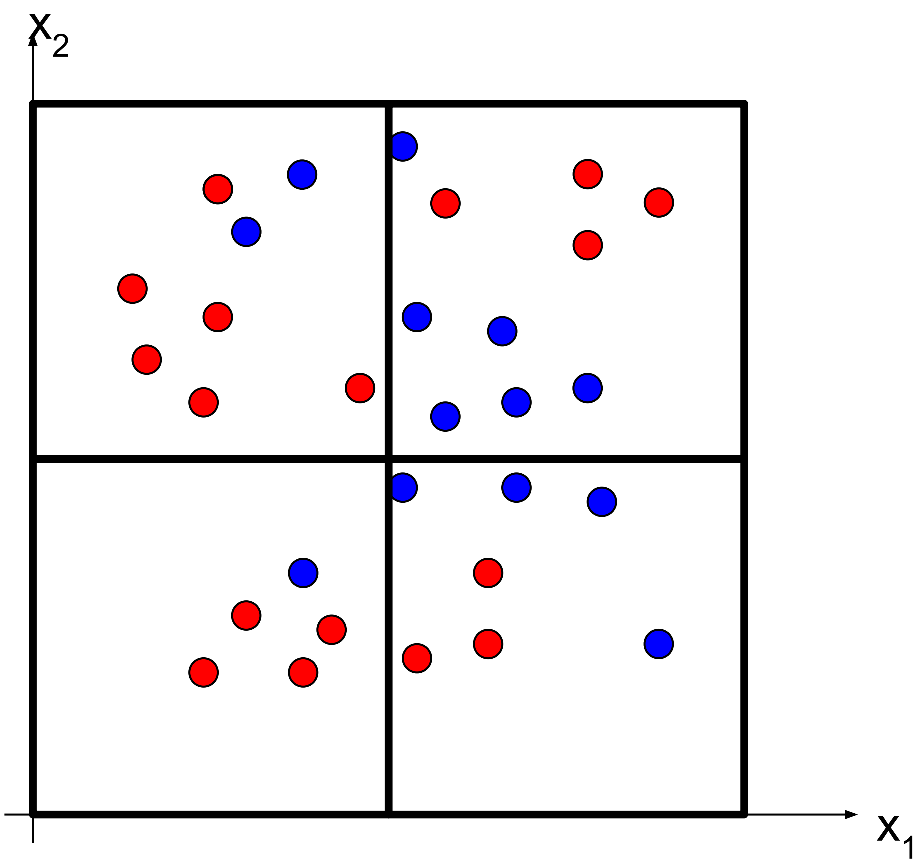
From kNN to Search Trees

- Search trees
  → Quad/Octree, KD tree, etc.



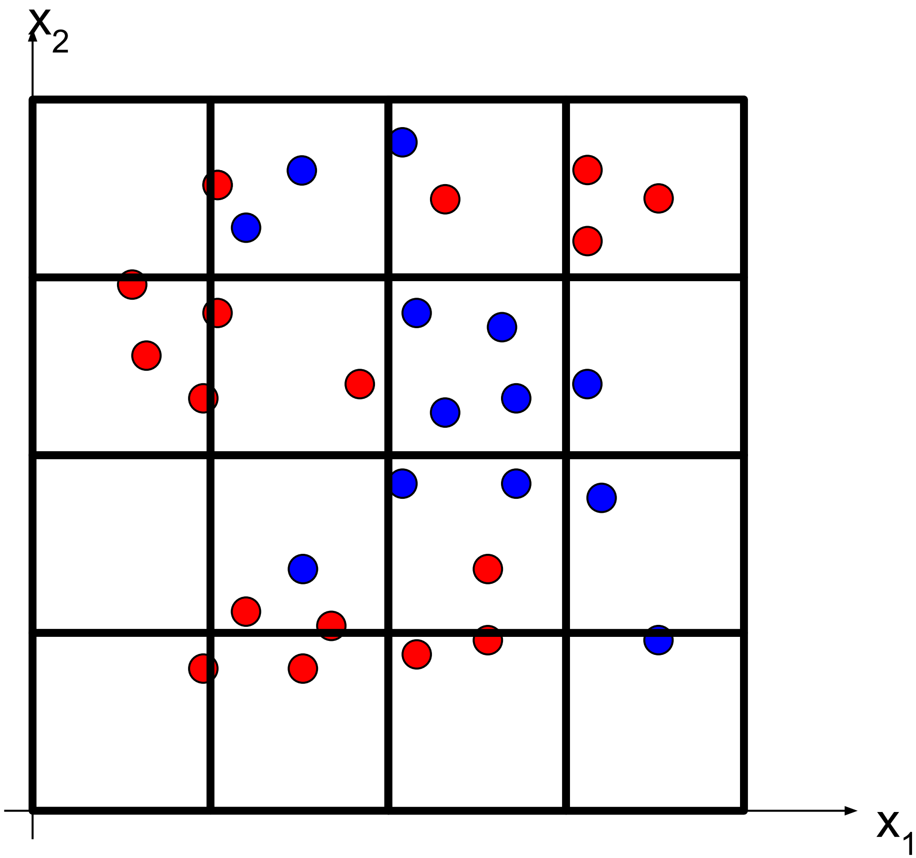
From kNN to Search Trees

- Search trees
  → Quad/Octree, KD tree, etc.
  → Divide space recursively into cells



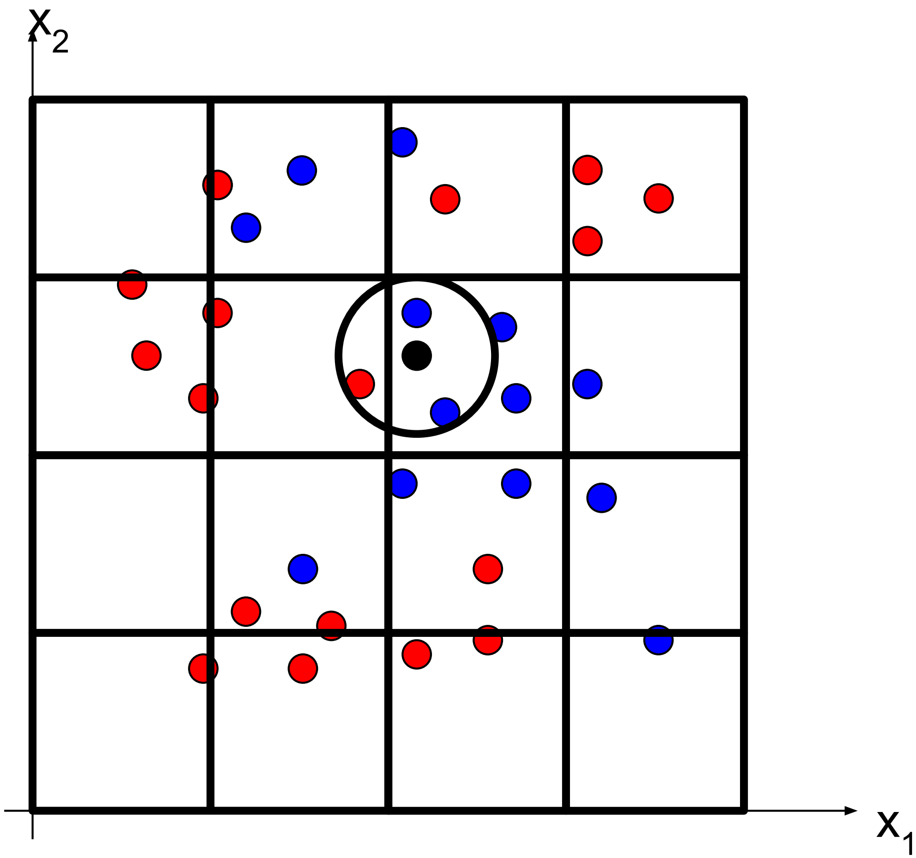
From kNN to Search Trees

- Search trees
  → Quad/Octree, KD tree, etc.
  → Divide space recursively into cells



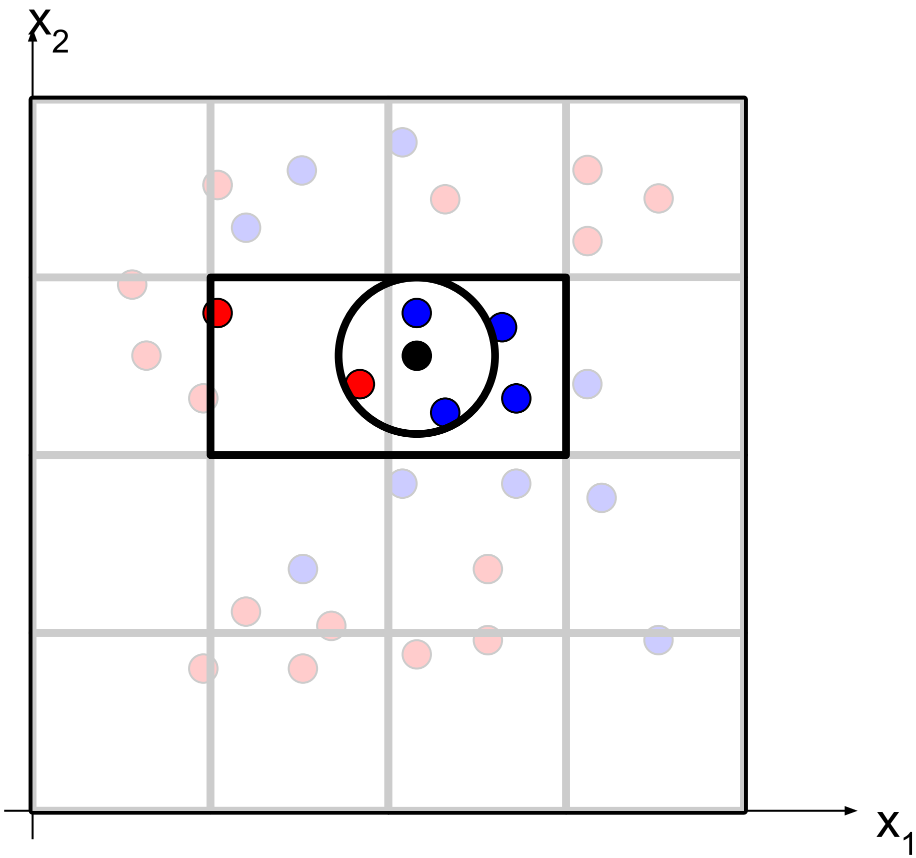
From kNN to Search Trees

- Search trees
  → Quad/Octree, KD tree, etc.
  → Divide space recursively into cells
  → Given a query, find relevant cells



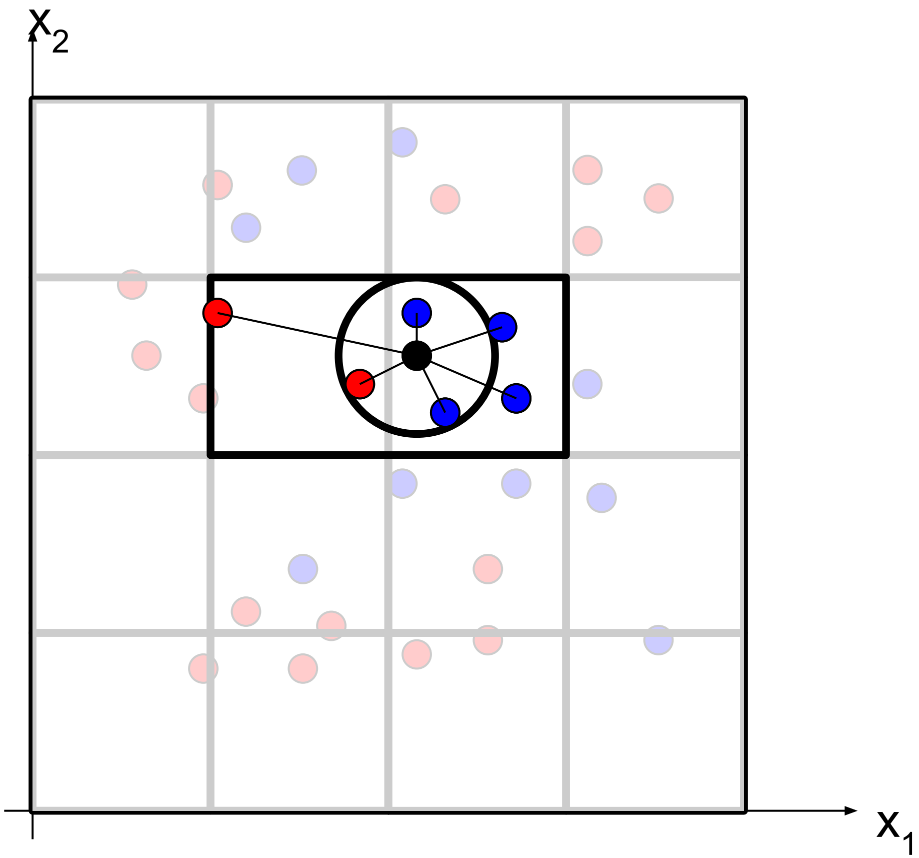
From kNN to Search Trees

- Search trees
  → Quad/Octree, KD tree, etc.
  → Divide space recursively into cells
  → Given a query, find relevant cells



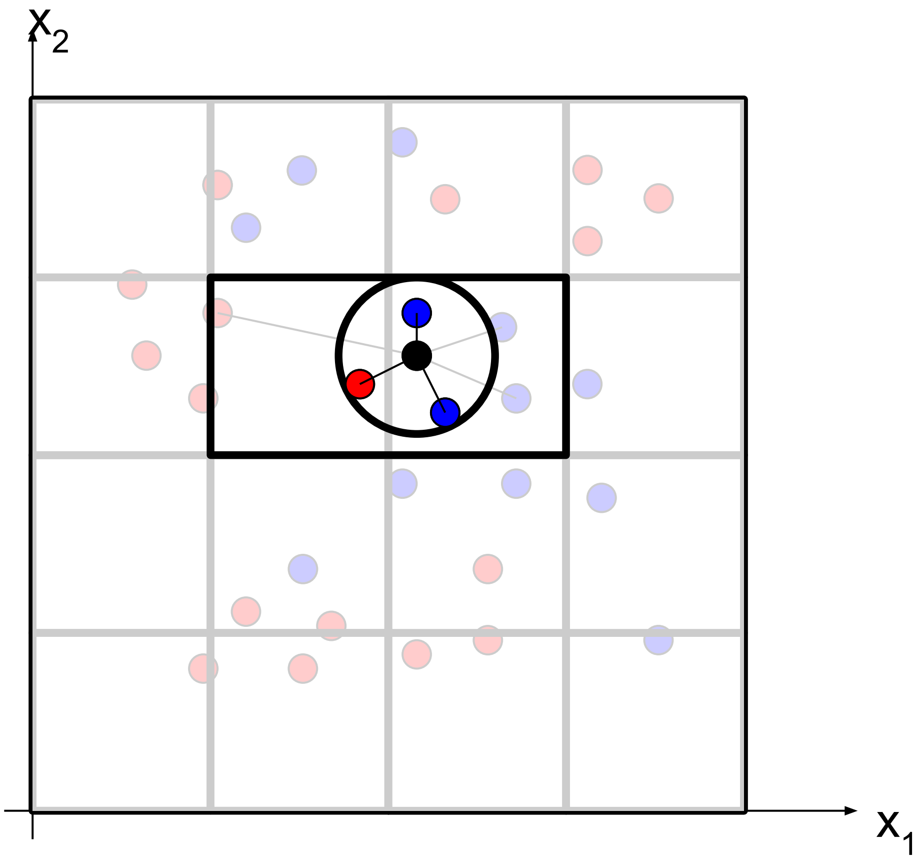
From kNN to Search Trees

- Search trees
  → Quad/Octree, KD tree, etc.
  → Divide space recursively into cells
  → Given a query, find relevant cells
  → Perform exhaustive search in these cells ONLY



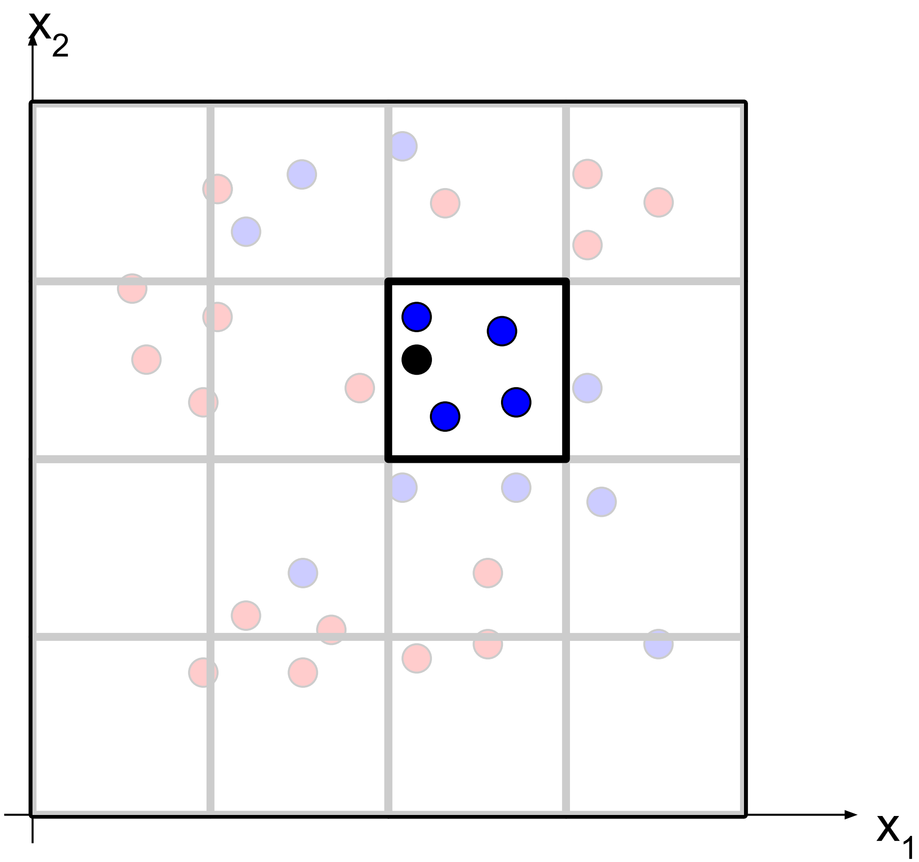
From kNN to Search Trees

- Search trees
  → Quad/Octree, KD tree, etc.
  → Divide space recursively into cells
  → Given a query, find relevant cells
  → Perform exhaustive search in these cells ONLY
- Exact search: Leads to equivalent results



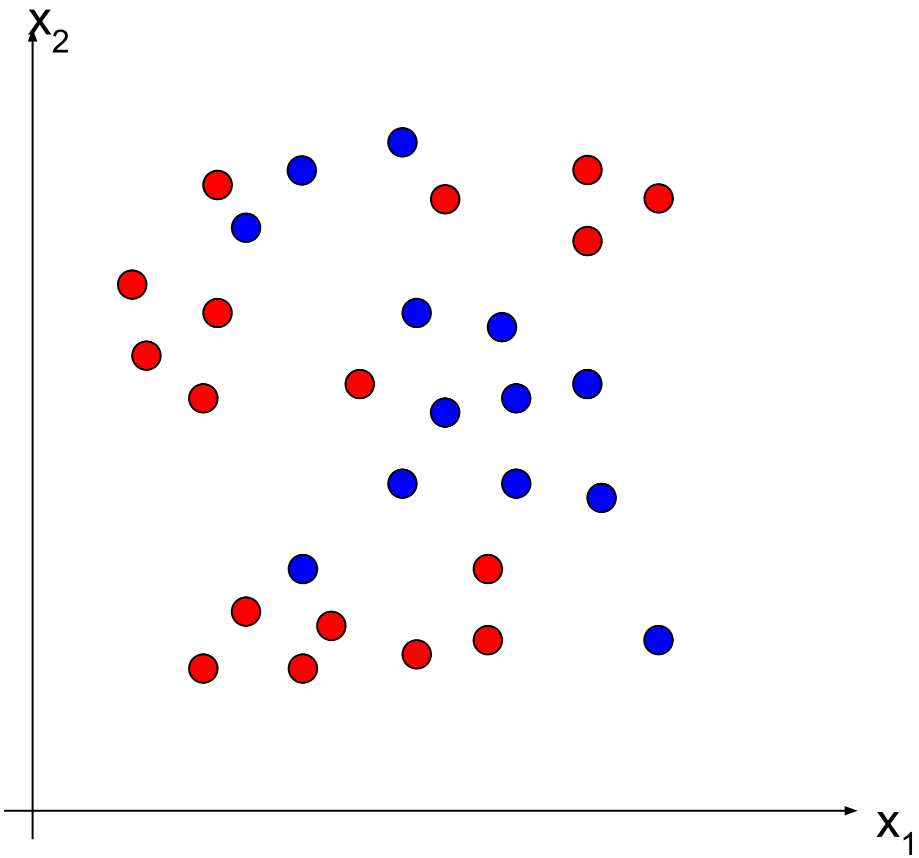
From kNN to Search Trees

- Search trees
  → Quad/Octree, KD tree, etc.
  → Divide space recursively into cells
  → Given a query, find relevant cells
  → Perform exhaustive search in these cells ONLY
- Exact search: Leads to equivalent results
- Approximation: Use samples within query cell directly



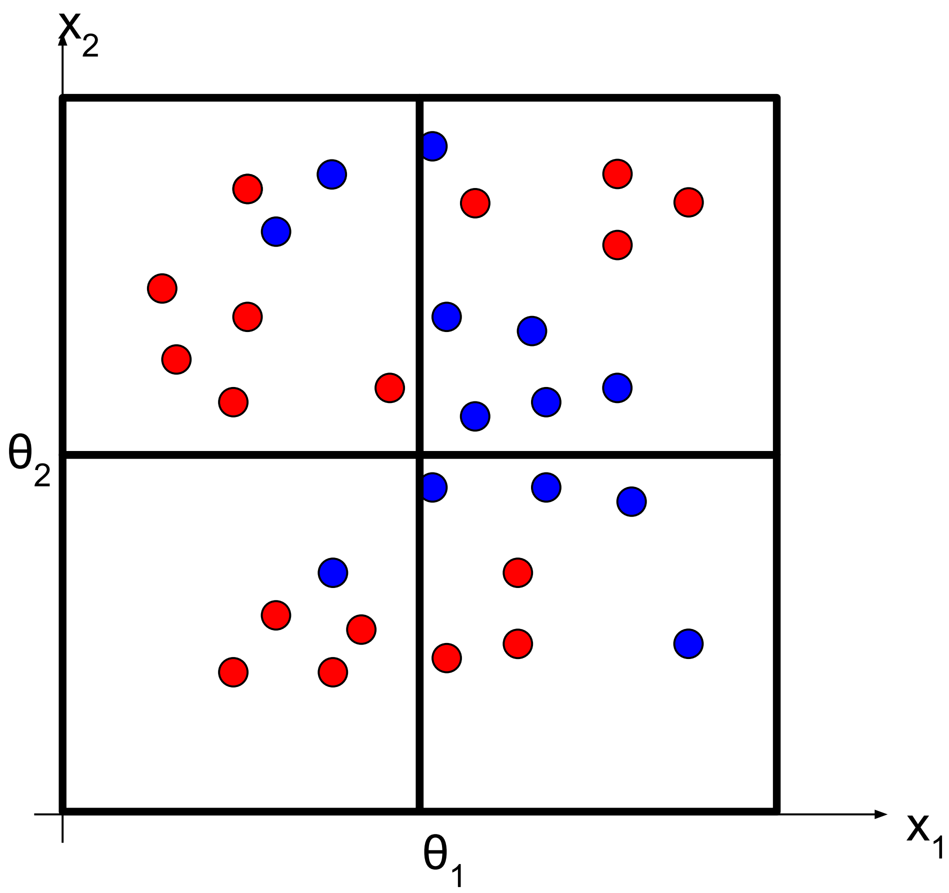
From Search Trees to (Random) Decision Trees

- Cell construction



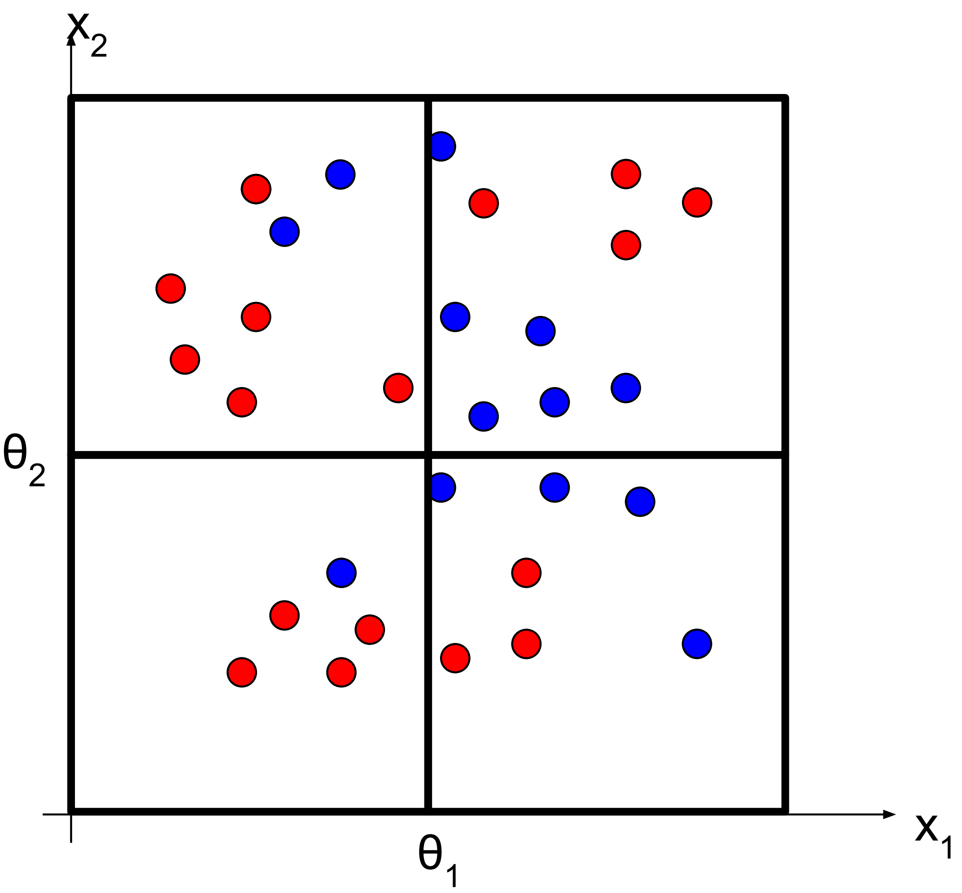
From Search Trees to (Random) Decision Trees

- Cell construction



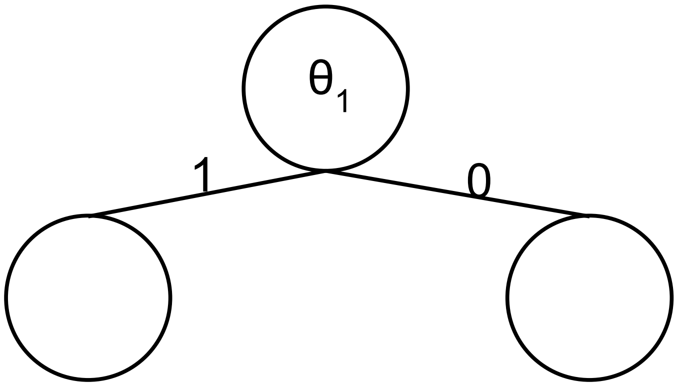
From Search Trees to (Random) Decision Trees

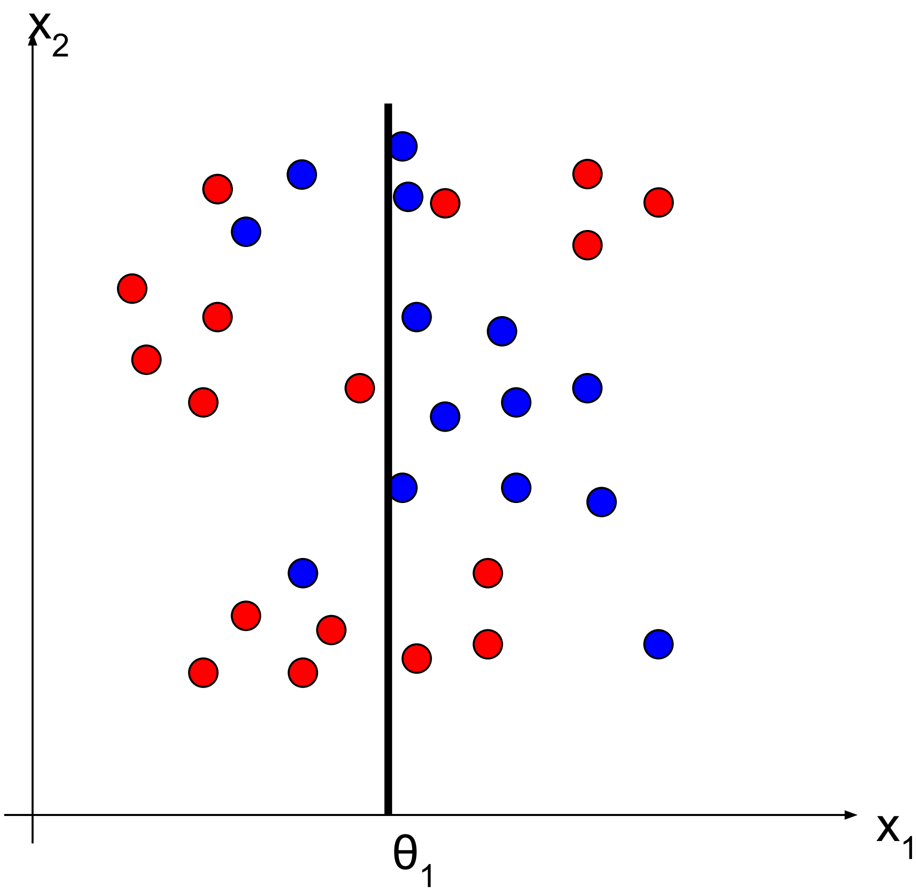
- Cell construction
  → Simple threshold operation
  → Different threshold definitions (e.g. equi-sized cells, threshold as data median) lead to different search tree variants (e.g. quad-tree, k-D tree).



From Search Trees to (Random) Decision Trees

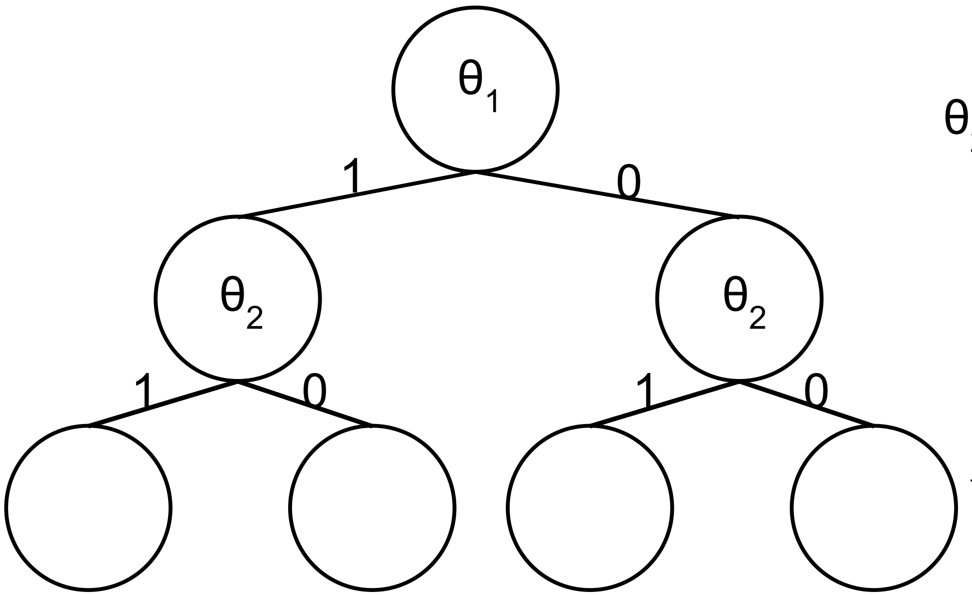
- Cell construction
  → Simple threshold operation
- Decision stump:

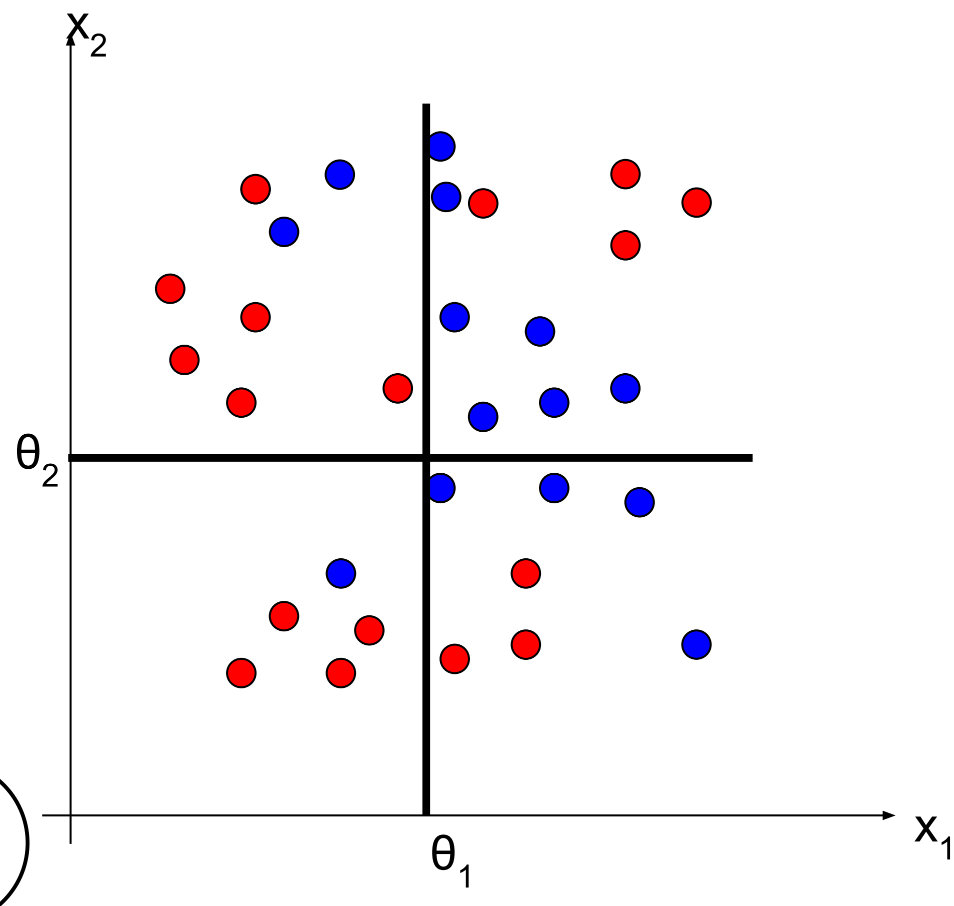




From Search Trees to (Random) Decision Trees

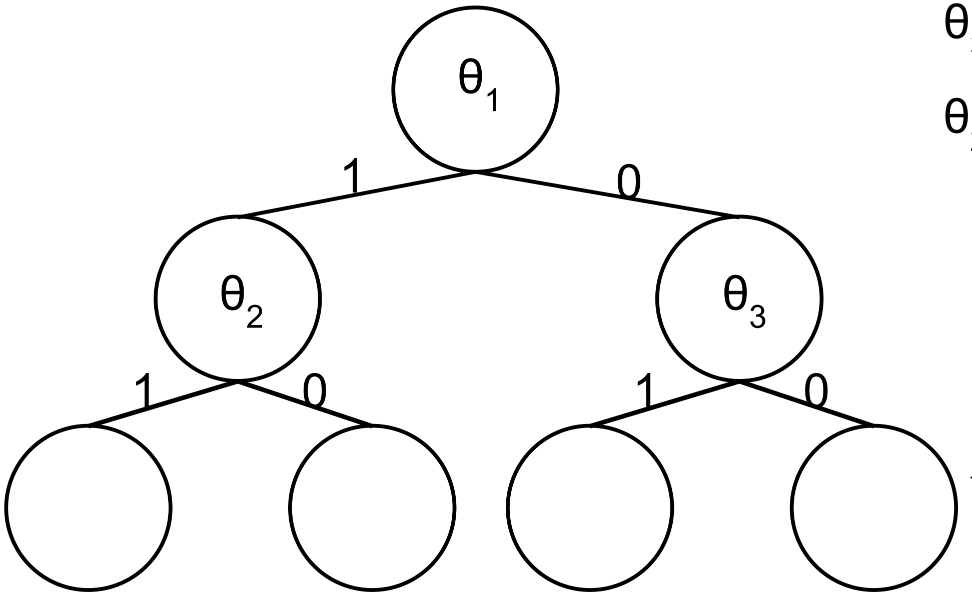
- Cell construction
  → Simple threshold operation
- Decision stump:

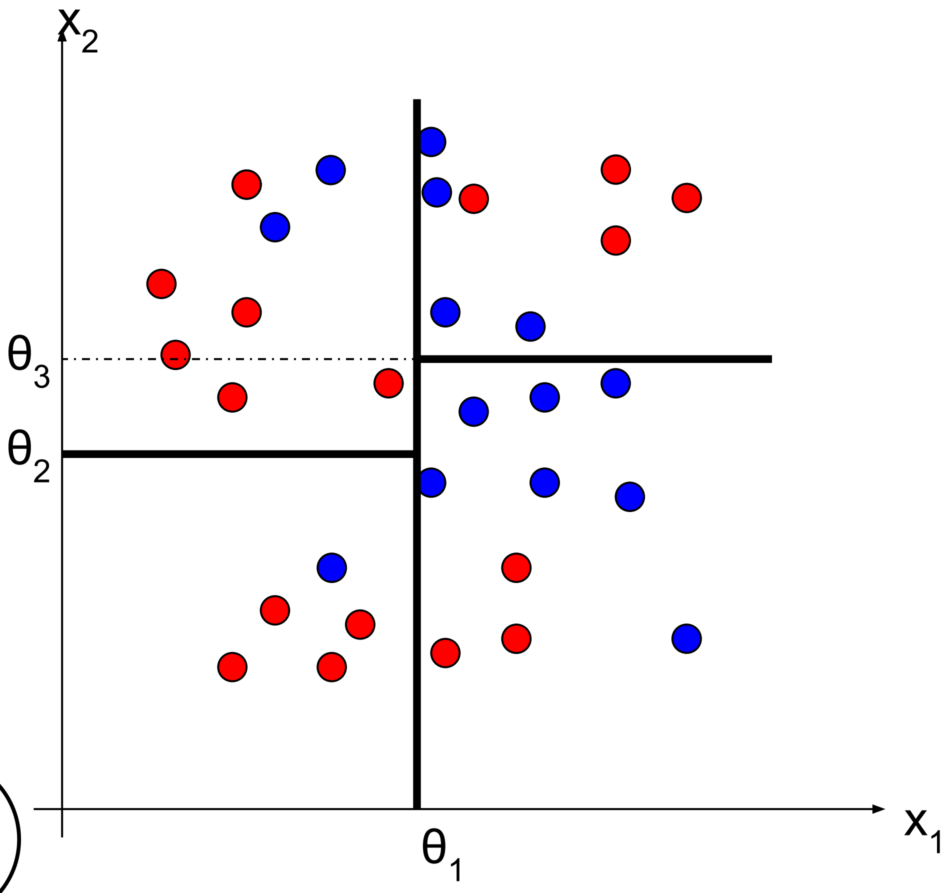




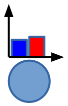
From Search Trees to (Random) Decision Trees

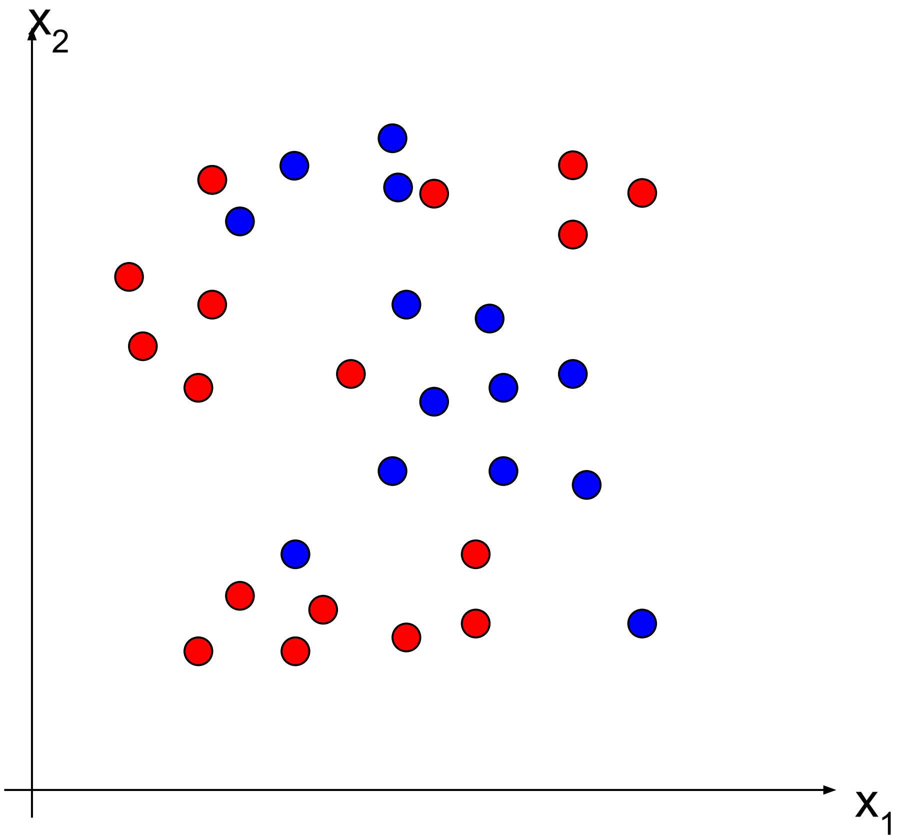
- Cell construction
  → Simple threshold operation
- Decision stump:



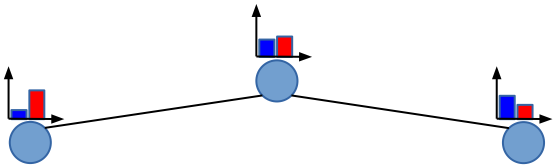


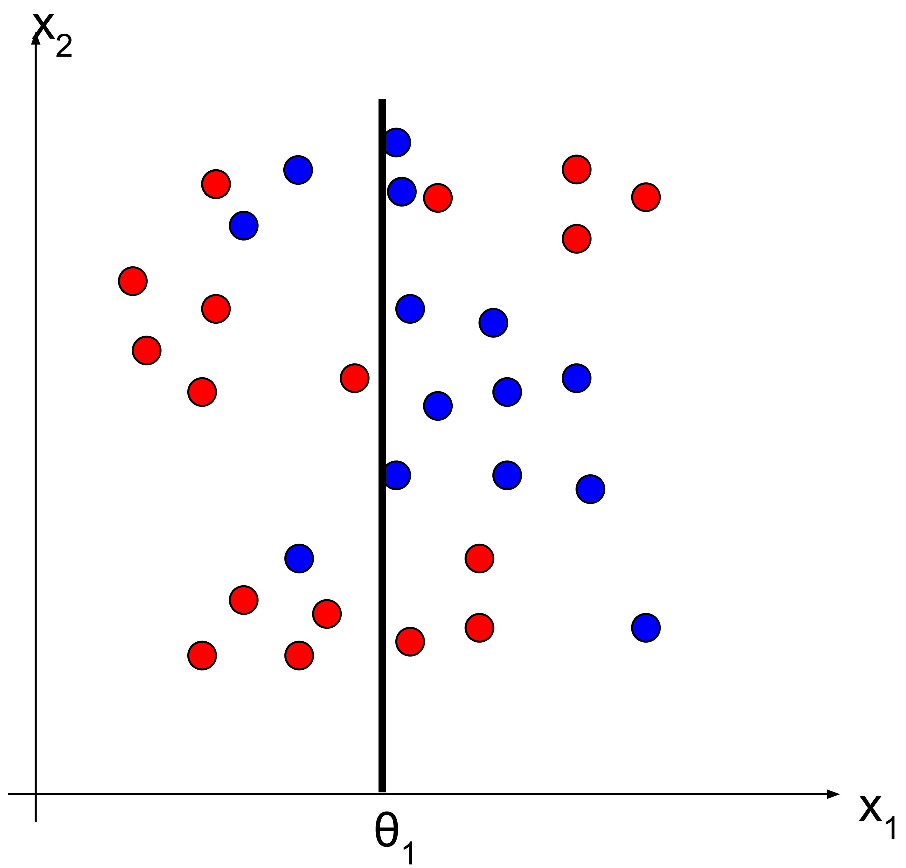
From Search Trees to (Random) Decision Trees





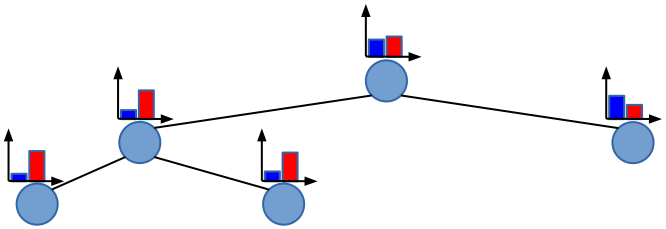
From Search Trees to (Random) Decision Trees

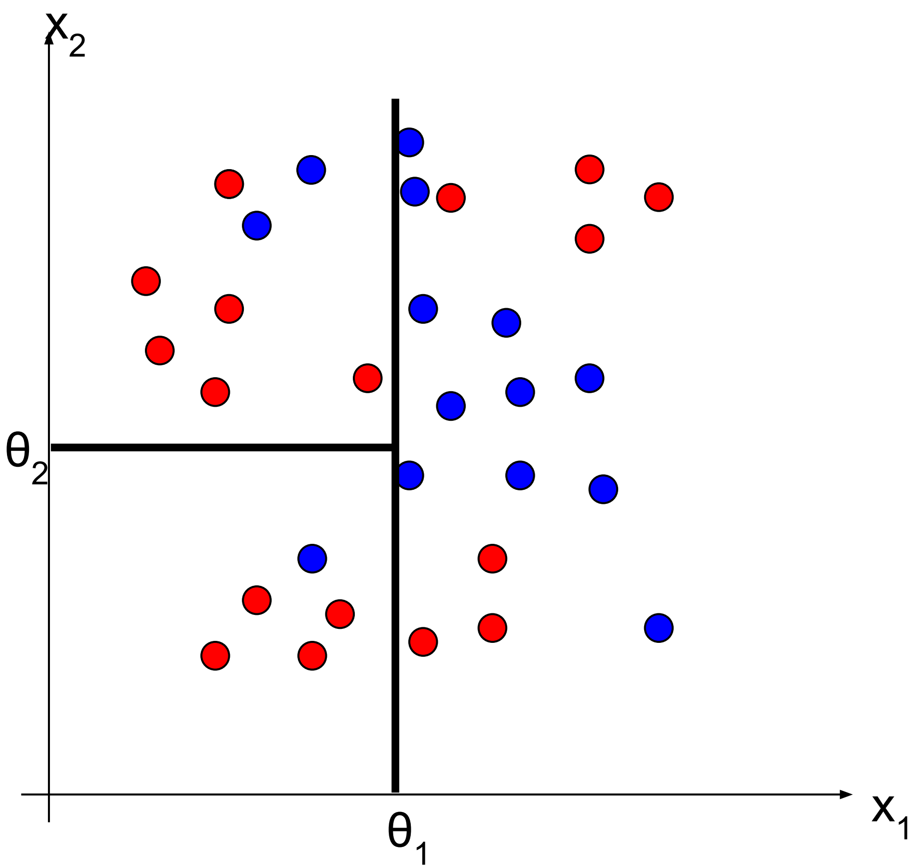




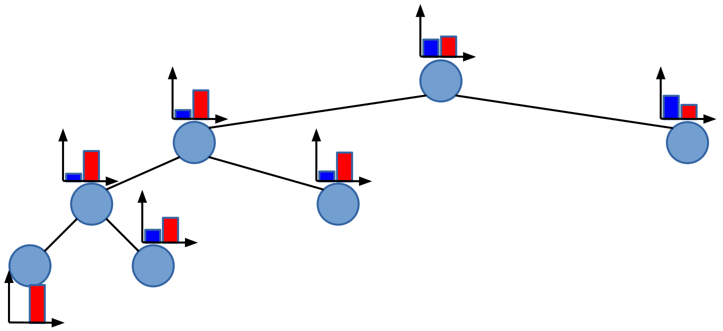
DLR

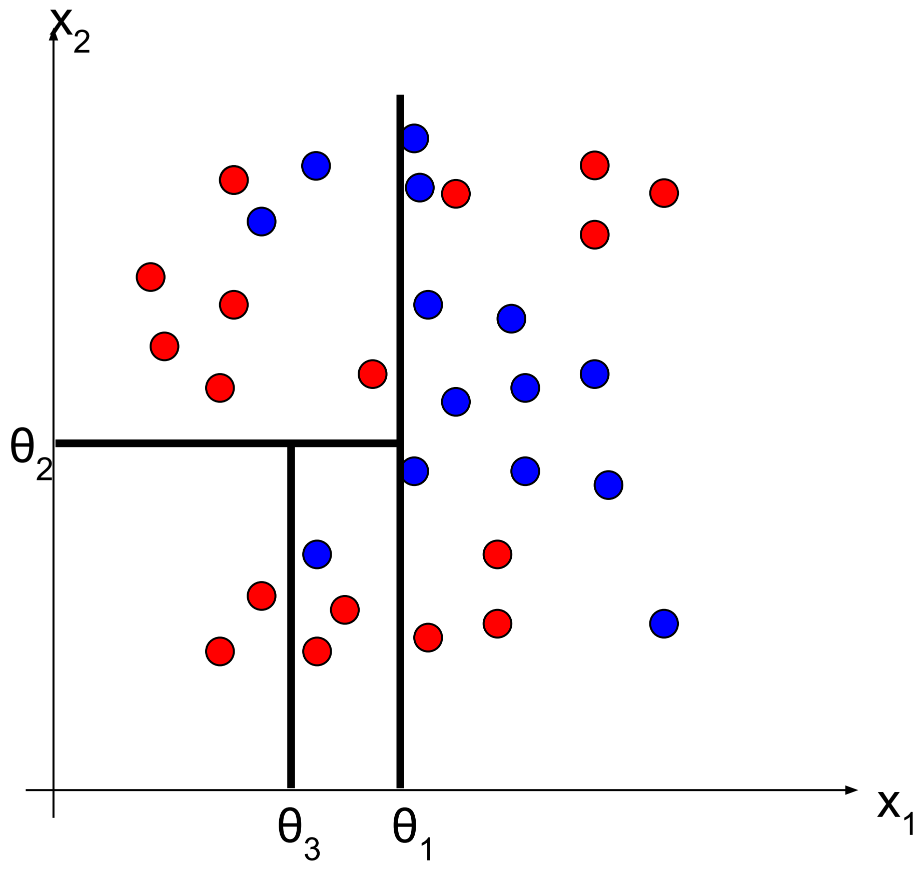
From Search Trees to (Random) Decision Trees



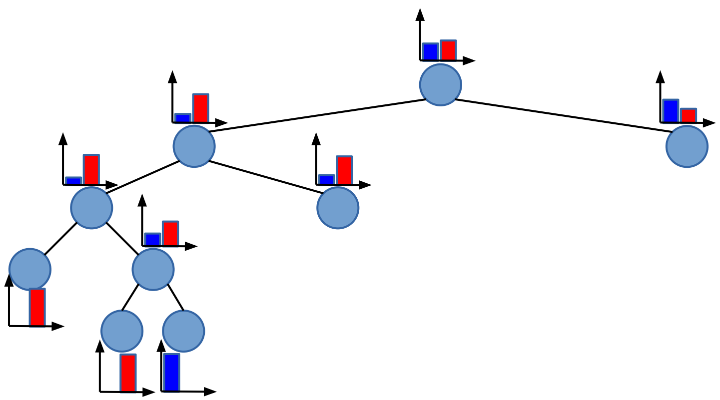


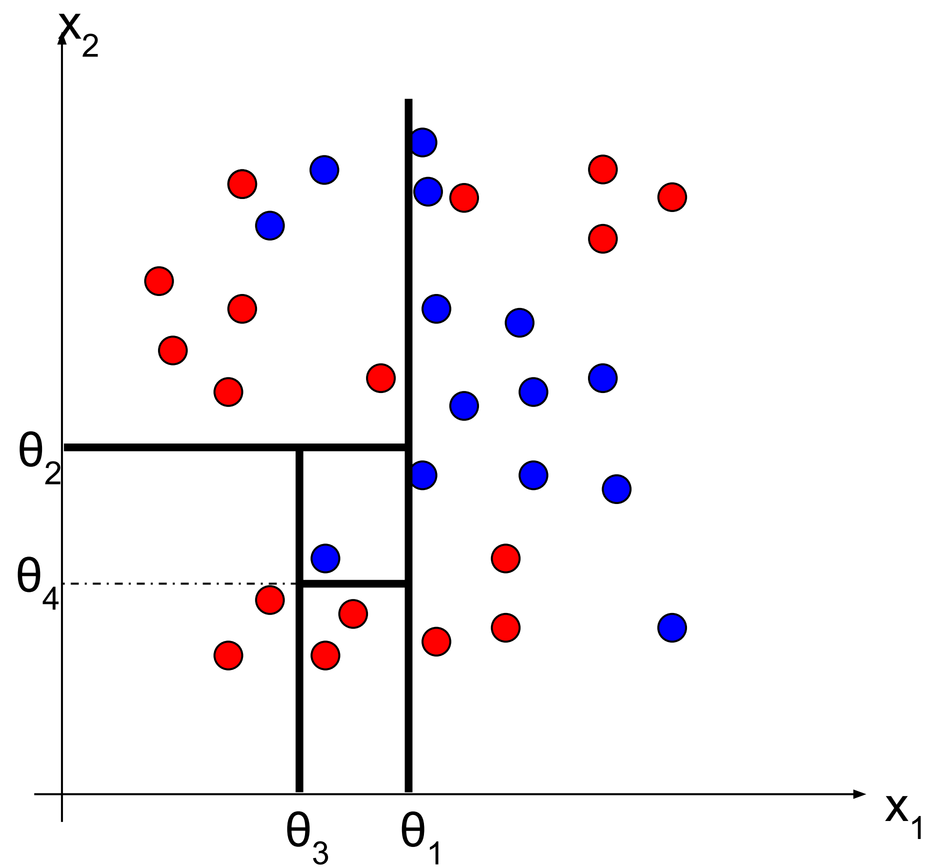
From Search Trees to (Random) Decision Trees



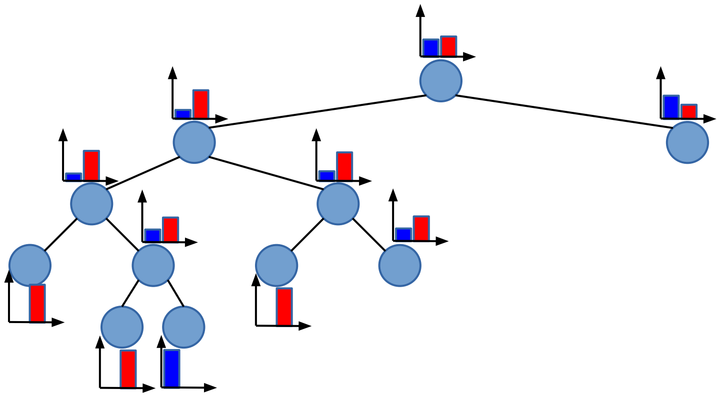


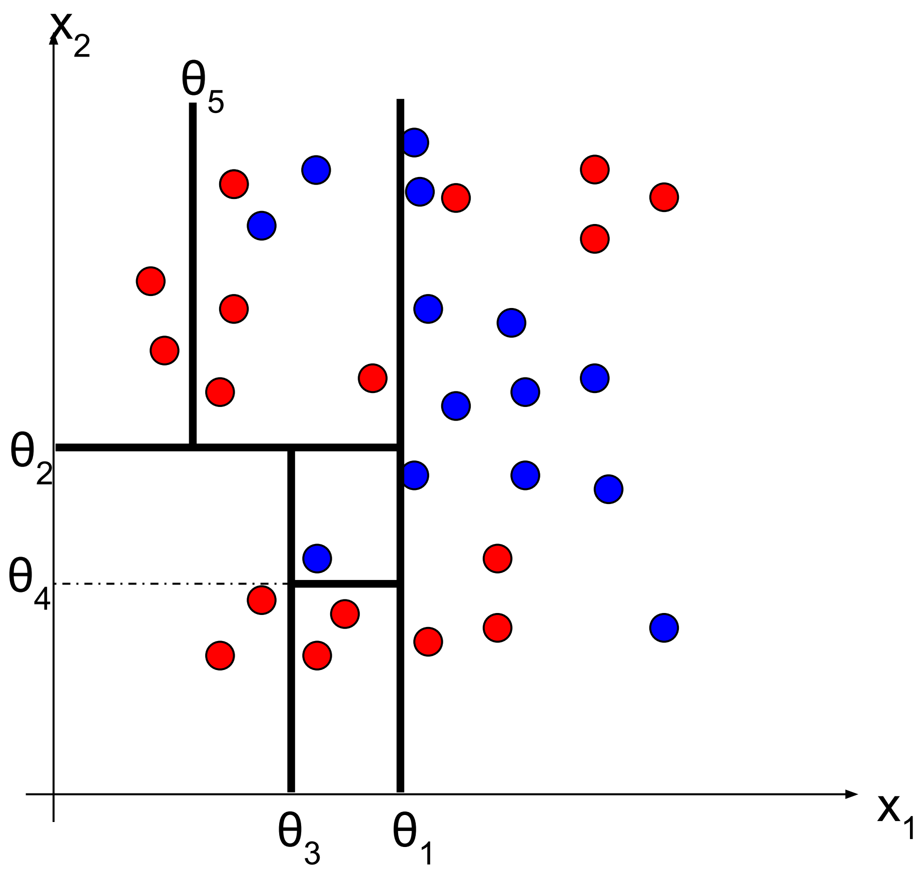
From Search Trees to (Random) Decision Trees



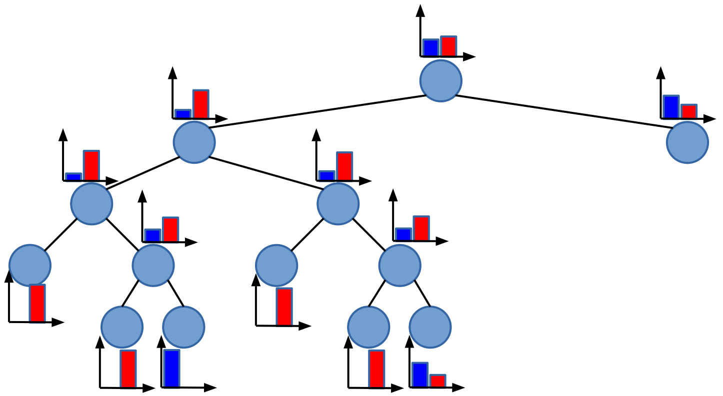


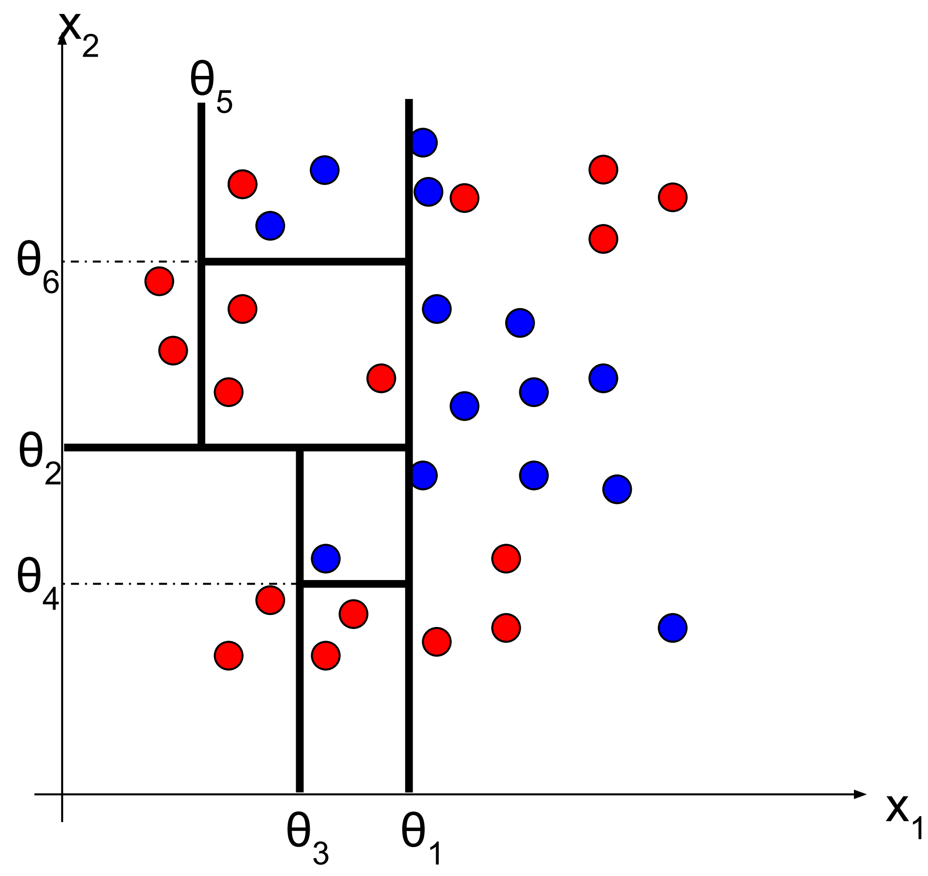
From Search Trees to (Random) Decision Trees



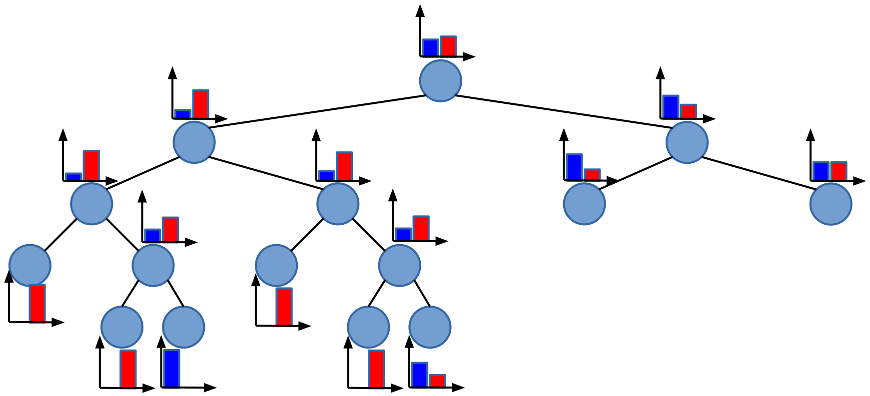


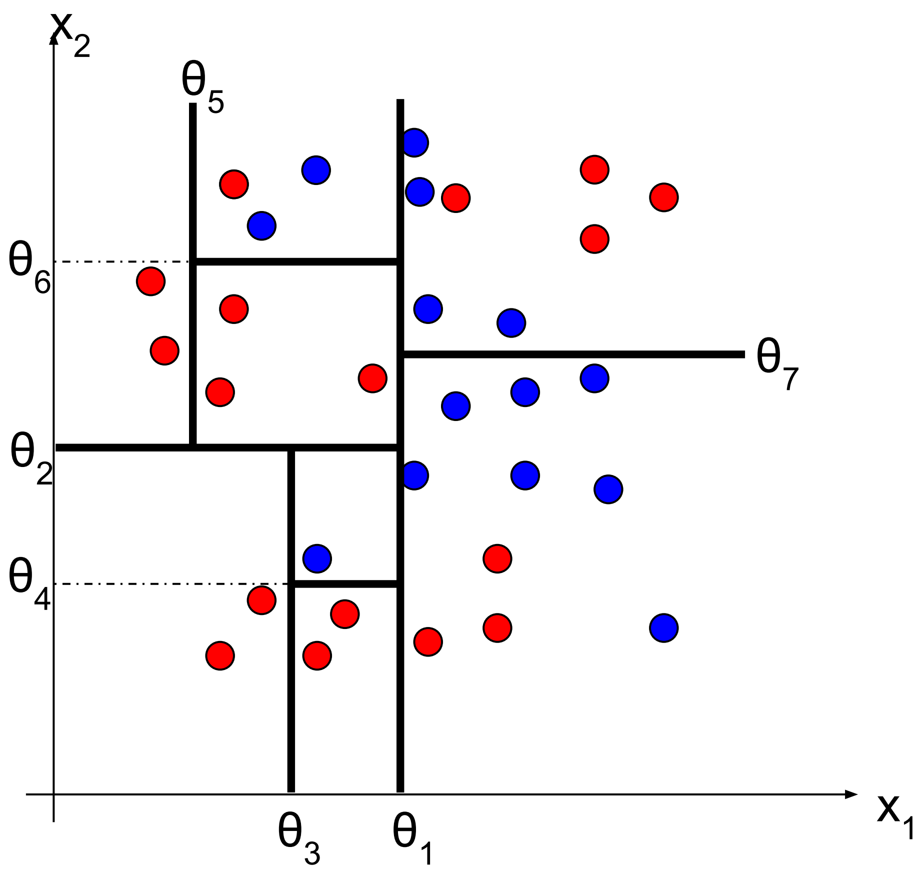
From Search Trees to (Random) Decision Trees



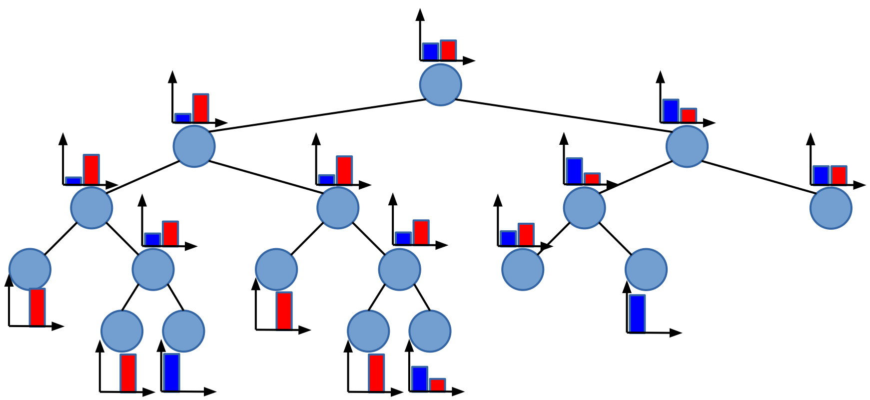


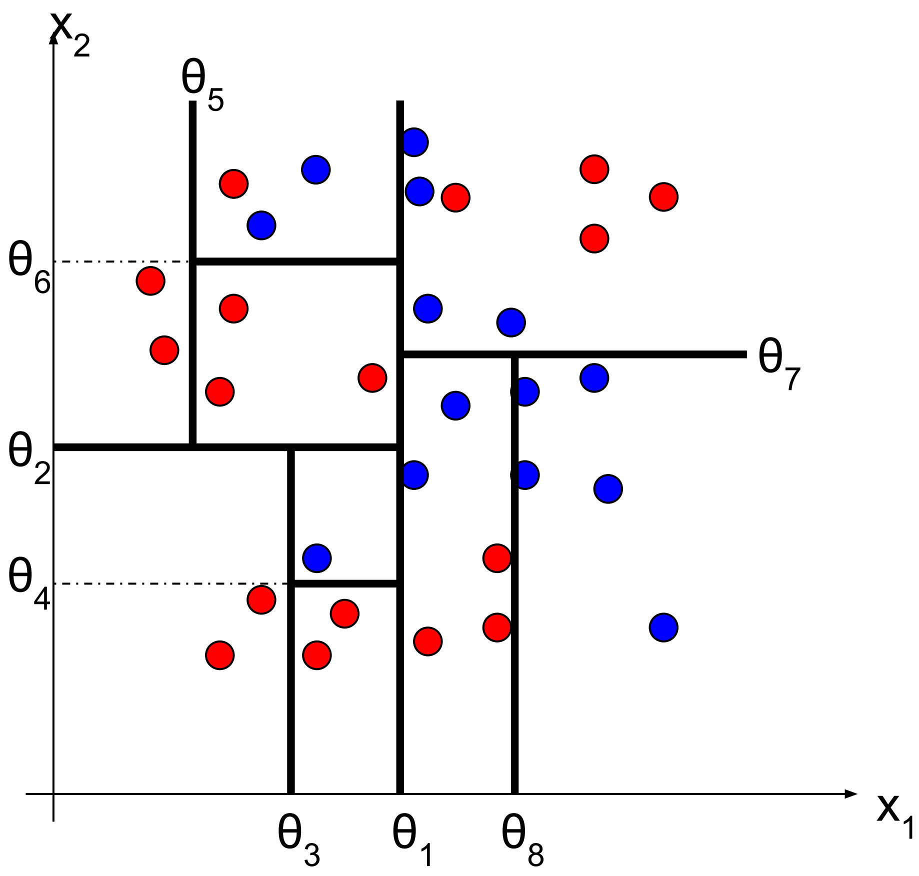
From Search Trees to (Random) Decision Trees



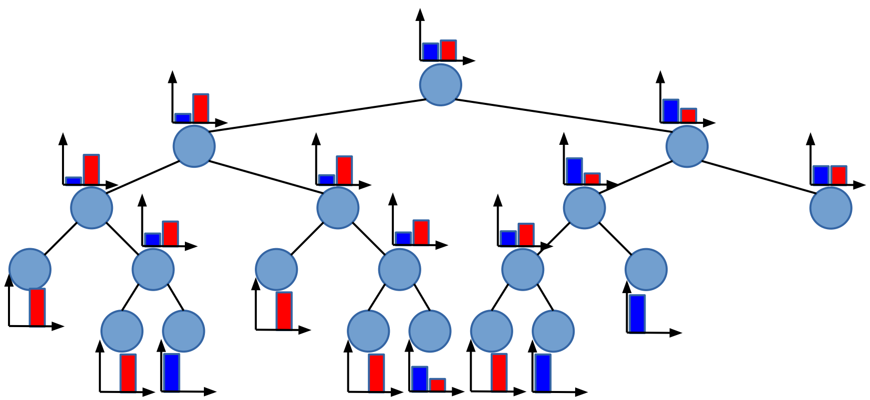


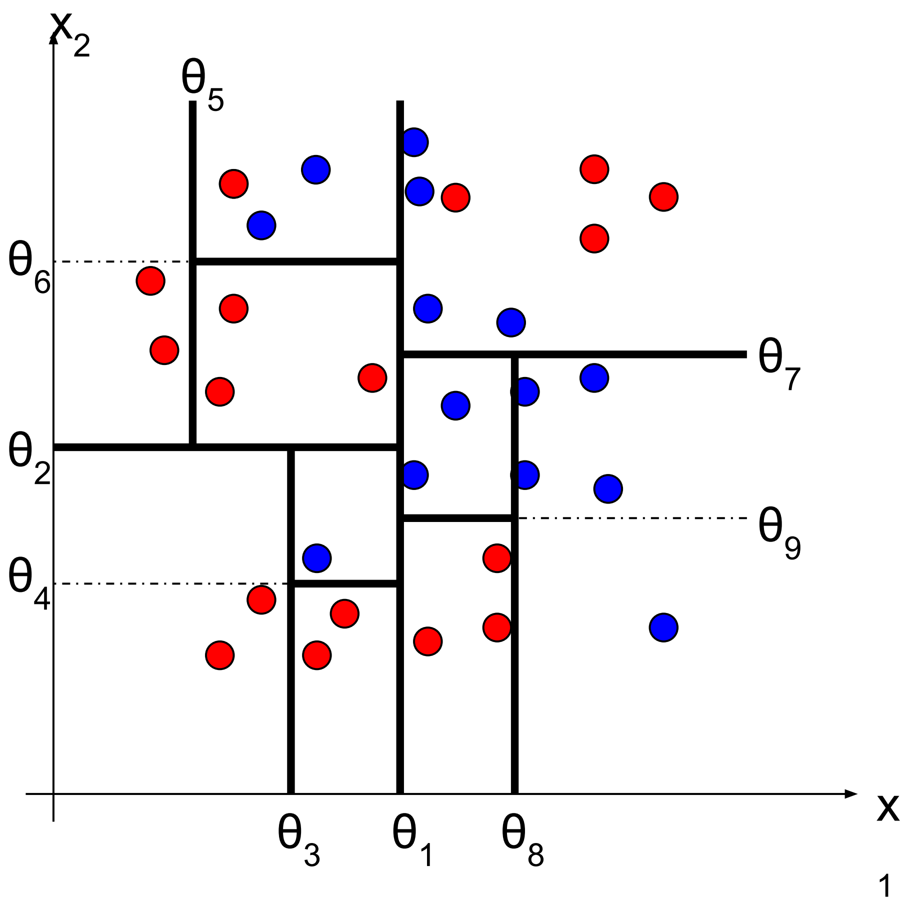
From Search Trees to (Random) Decision Trees



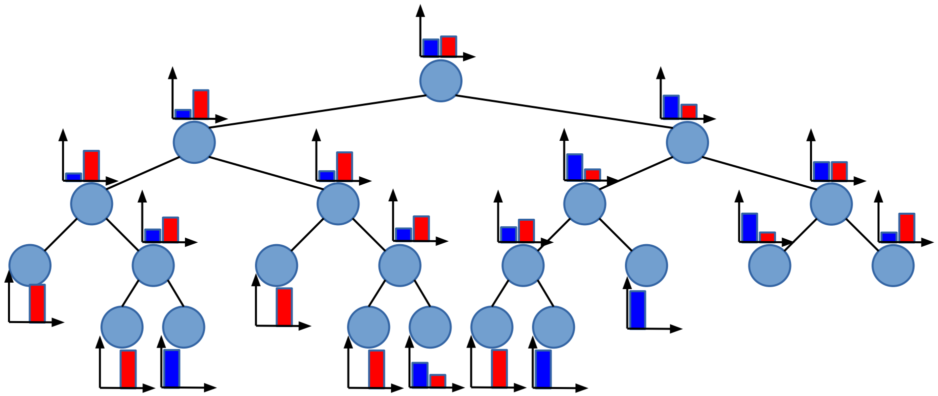


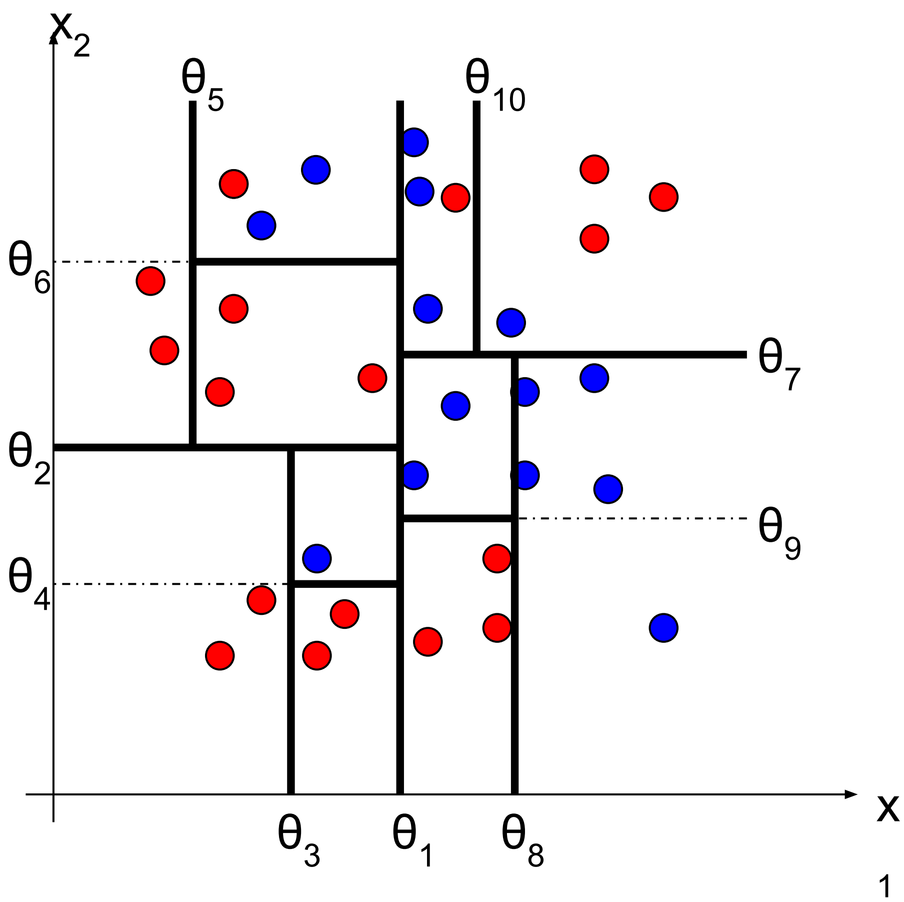
From Search Trees to (Random) Decision Trees



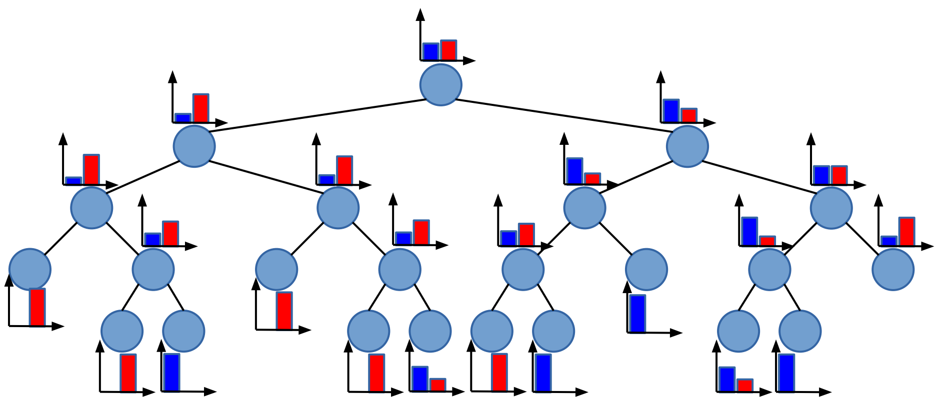


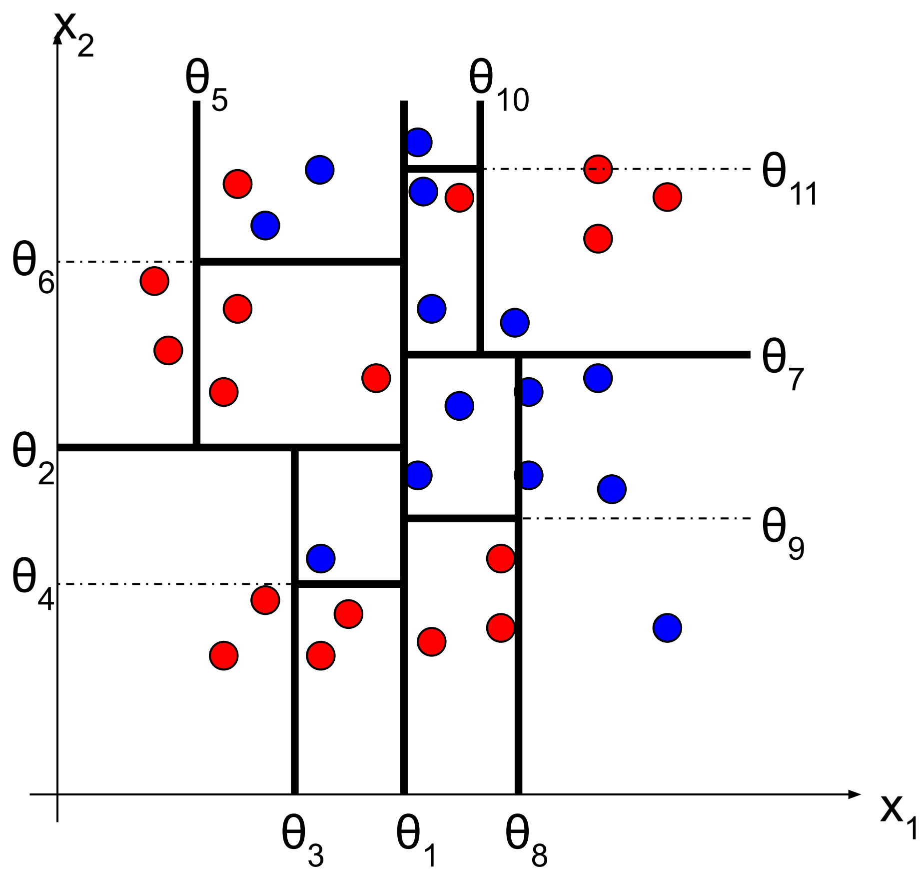
From Search Trees to (Random) Decision Trees



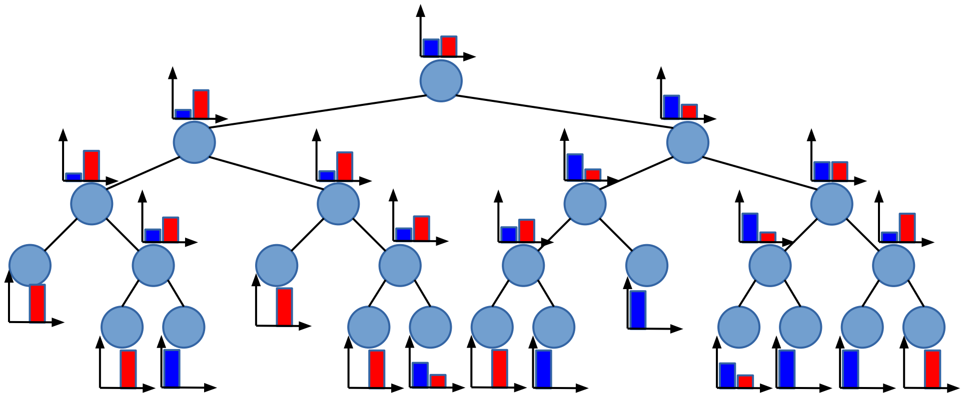


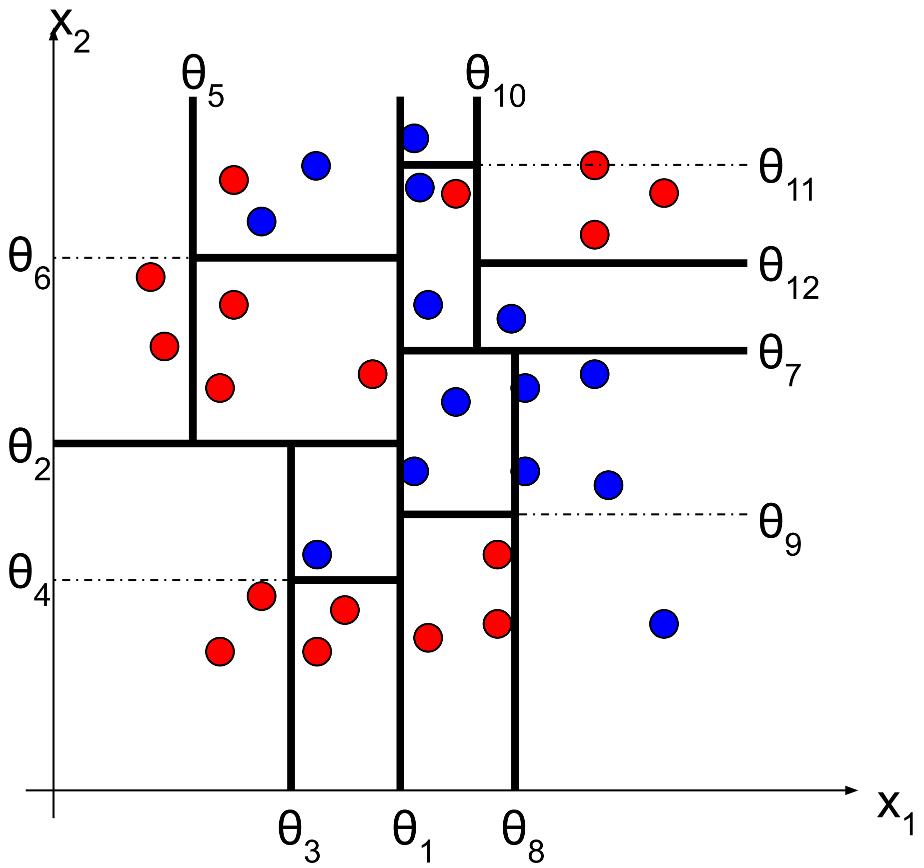
From Search Trees to (Random) Decision Trees



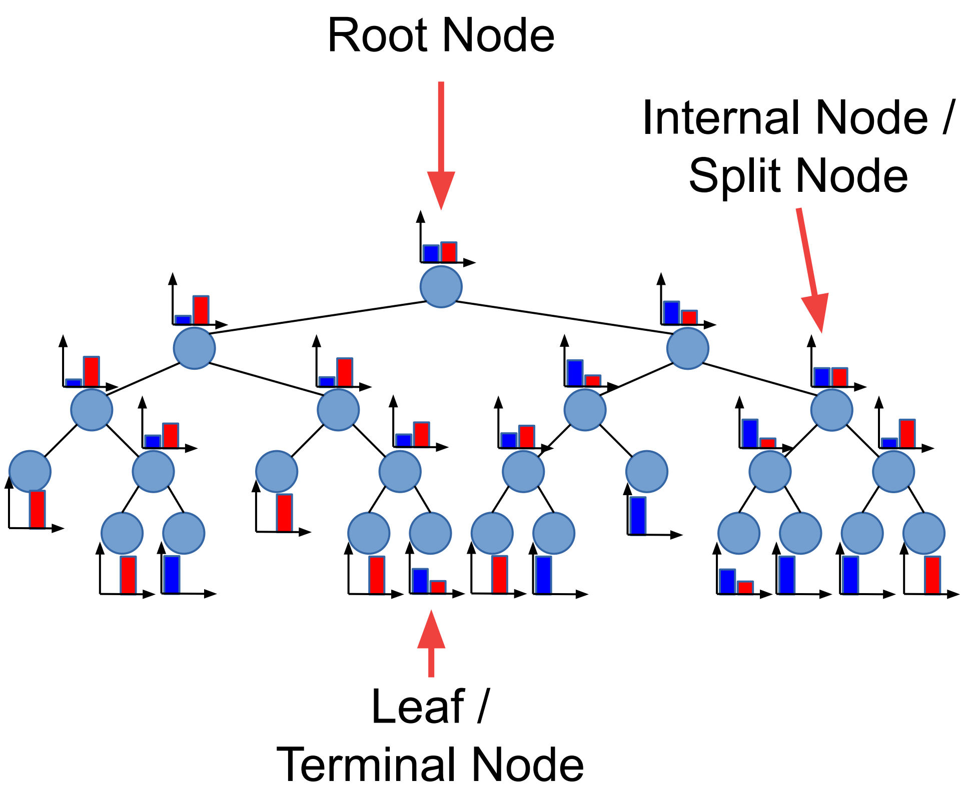


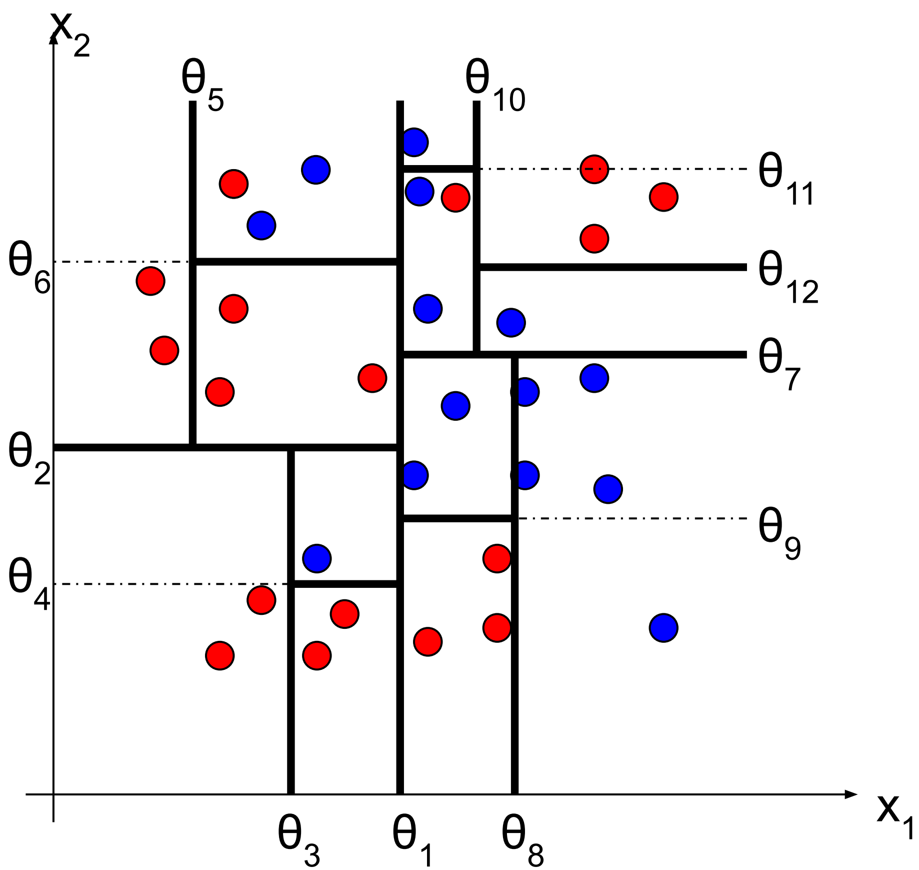
From Search Trees to (Random) Decision Trees





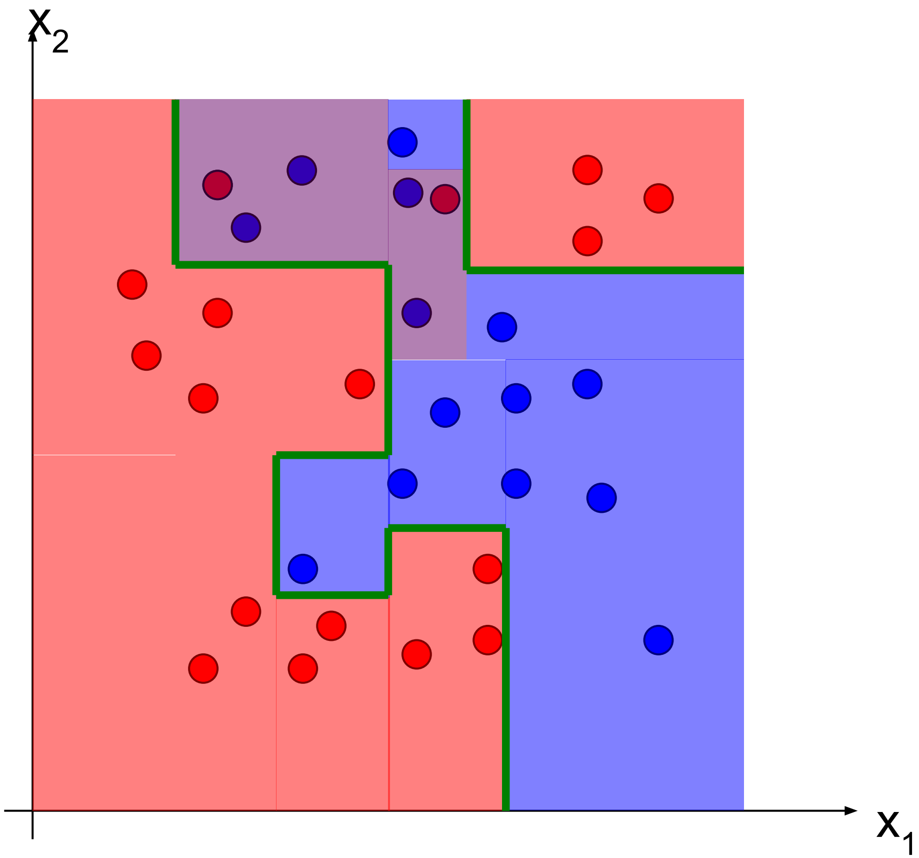
From Search Trees to (Random) Decision Trees





From Search Trees to (Random) Decision Trees

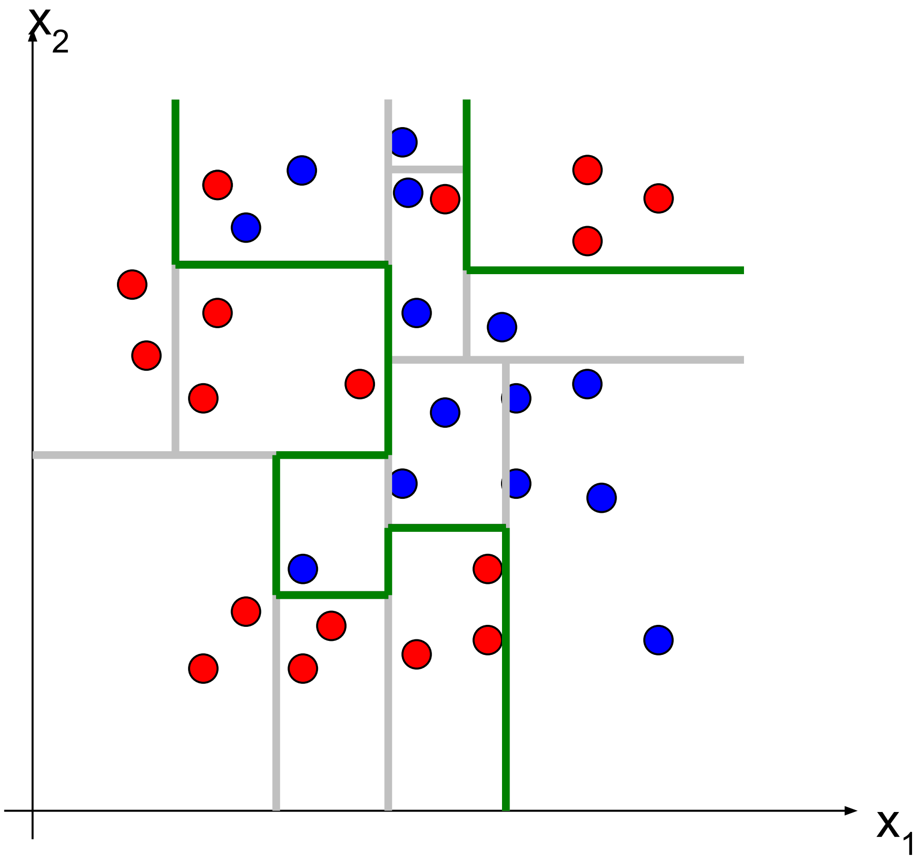
Local estimate of the target variable (e.g. class posterior) is assigned to cells



From Search Trees to (Random) Decision Trees

Local estimate of the target variable (e.g. class posterior) is assigned to cells

Results in highly non-linear, even non-connected (but piecewise constant) decision boundaries



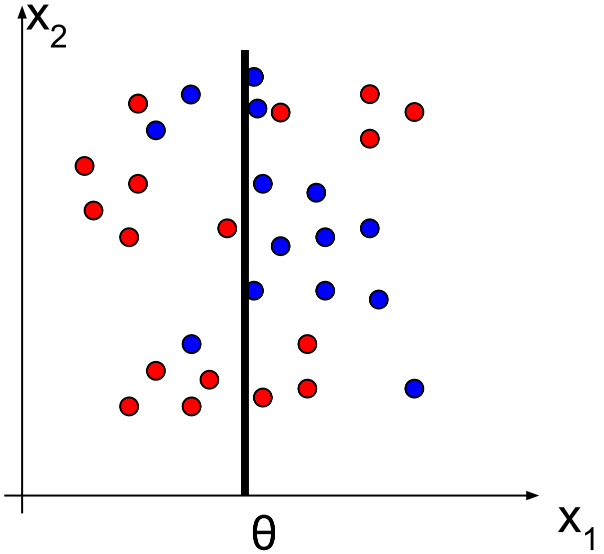
From Search Trees to (Random) Decision Trees

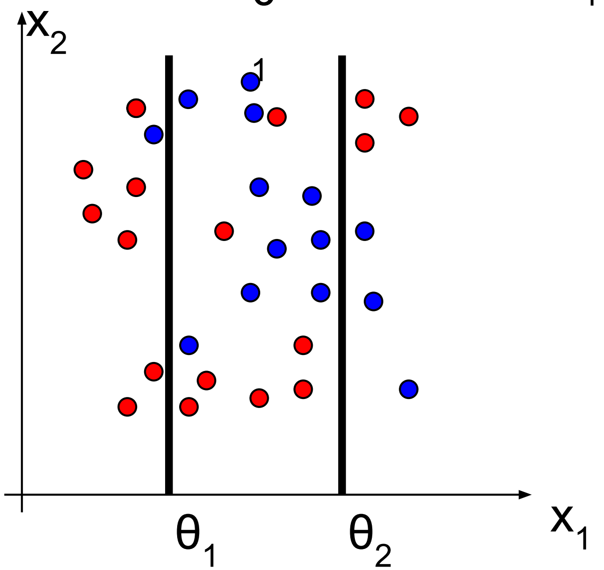
Other node tests are possible:

→ Axis-aligned

$$t(\boldsymbol{x}) = \begin{cases} 0 & \text{if } x_i < \theta_r \\ 1 & \text{otherwise.} \end{cases}$$

$$t(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \theta_r < x_i < \theta_s \\ 1 & \text{otherwise.} \end{cases}$$





From Search Trees to (Random) Decision Trees

Other node tests are possible:

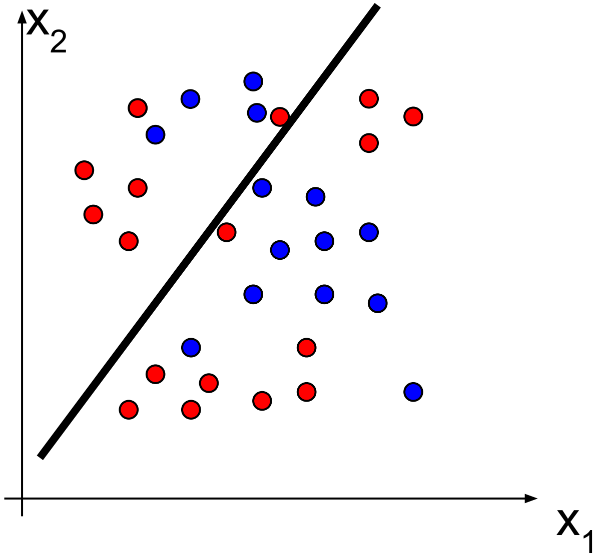
→ Axis-aligned

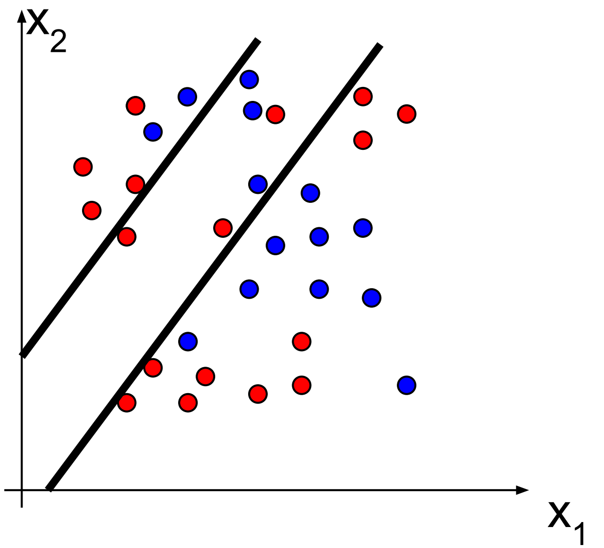
→ Linear

$$\widetilde{\mathbf{x}} = [\mathbf{x}, 1] \in \mathbb{R}^{d+1}, \psi \in \mathbb{R}^{d+1}$$

$$t(\mathbf{x}) = \begin{cases} 0 & \text{if } \psi^T \widetilde{\mathbf{x}} < \theta_r \\ 1 & \text{otherwise.} \end{cases}$$

$$t(\mathbf{x}) = \begin{cases} 0 & \text{if } \theta_r < \psi^T \widetilde{\mathbf{x}} < \theta_s \\ 1 & \text{otherwise.} \end{cases}$$





DLR

From Search Trees to (Random) Decision Trees

Other node tests are possible:

→ Axis-aligned

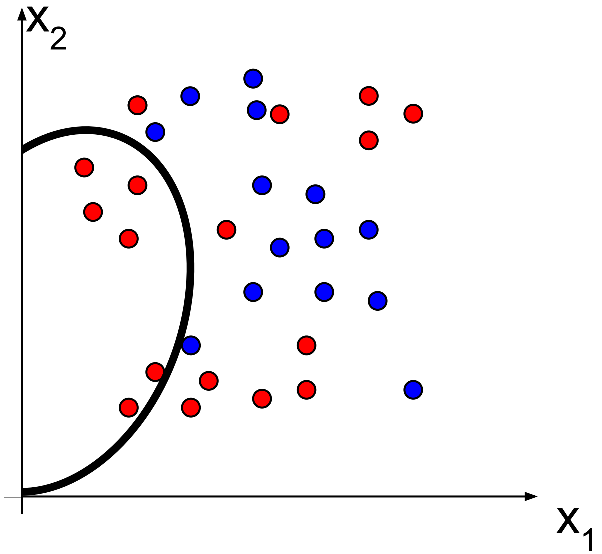
→ Linear

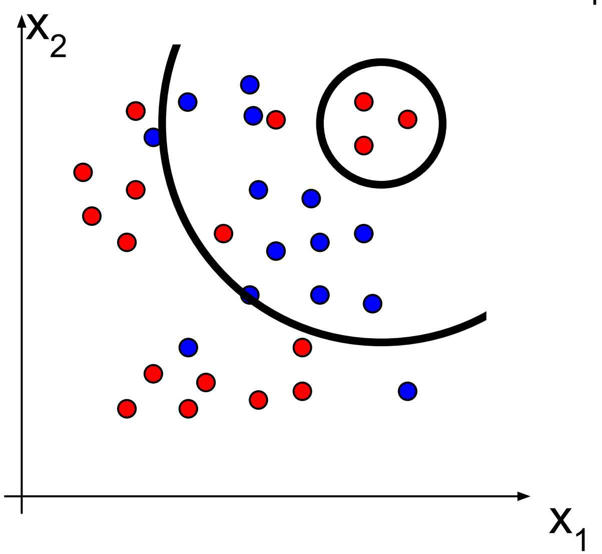
→ Conic section

$$\tilde{\mathbf{x}} = [\mathbf{x}, 1] \in \mathbb{R}^{d+1},\ \psi \in \mathbb{R}^{(d+1)\times(d+1)}$$

$$t(\mathbf{x}) = \begin{cases} 0 & \text{if } \tilde{\mathbf{x}}^T \cdot \psi \cdot \tilde{\mathbf{x}} < \theta_r \\ 1 & \text{otherwise.} \end{cases}$$

$$t(\mathbf{x}) = \begin{cases} 0 & \text{if } \theta_r < \tilde{\mathbf{x}}^T \cdot \psi \cdot \tilde{\mathbf{x}} < \theta_s \\ 1 & \text{otherwise.} \end{cases}$$





From Search Trees to (Random) Decision Trees

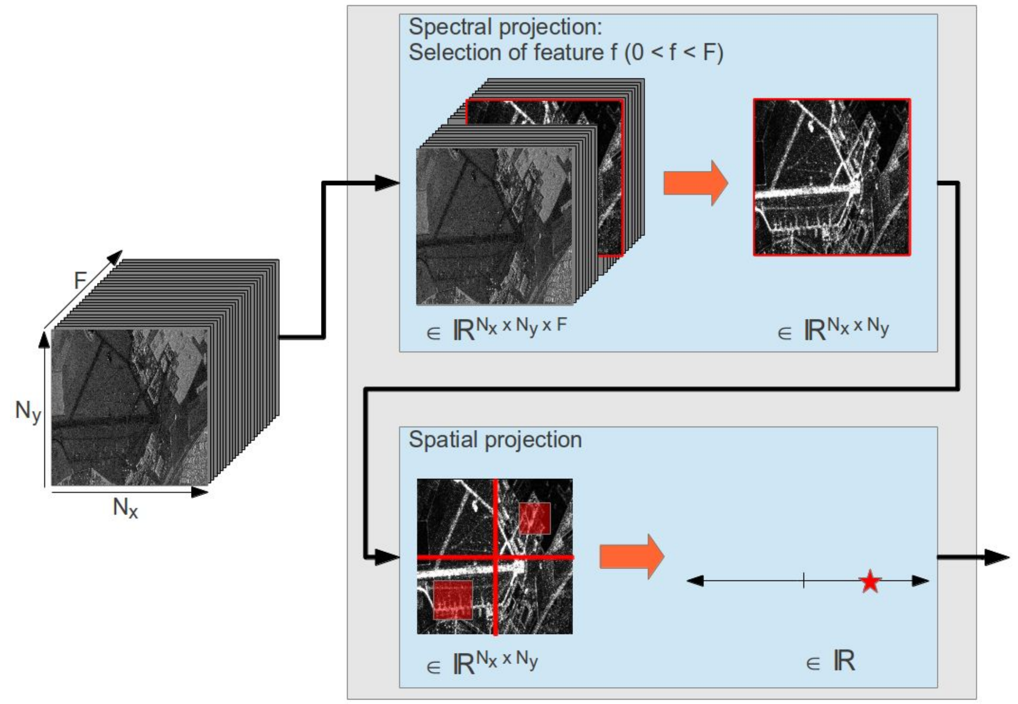
Other node tests are possible:

→ Axis-aligned

→ Linear

→ Conic section

→ Other data spaces than
- Image patches: $x \in R^{nxn}$
- Non-scalar features (histograms, categorical)
- ...



From (Random) Decision Trees to Random Forests

Advantages

- Can deal with very heterogeneous data
  → Different, data-specific types of node tests

From (Random) Decision Trees to Random Forests

Advantages

- Can deal with very heterogeneous data
  → Different, data-specific types of node tests
- Not prone to the curse of dimensionality
  → Each node only works on a very limited set of dimensions

From (Random) Decision Trees to Random Forests

Advantages

- Can deal with very heterogeneous data
  → Different, data-specific types of node tests
- Not prone to the curse of dimensionality
  → Each node only works on a very limited set of dimensions
- Very efficient
  → Each sample passes maximal H nodes (H = maximal height)

From (Random) Decision Trees to Random Forests

Advantages

- Can deal with very heterogeneous data
  → Different, data-specific types of node tests
- Not prone to the curse of dimensionality
  → Each node only works on a very limited set of dimensions
- Very efficient
  → Each sample passes maximal H nodes (H = maximal height)
- Easy to implement
  → Binary trees are one of the most basic data structures

From (Random) Decision Trees to Random Forests

Advantages

- Can deal with very heterogeneous data
  → Different, data-specific types of node tests
- Not prone to the curse of dimensionality
  → Each node only works on a very limited set of dimensions
- Very efficient
  → Each sample passes maximal H nodes (H = maximal height)
- Easy to implement
  → Binary trees are one of the most basic data structures
- Easy to interpret
  → Path through tree is a connected set of decision rules

From (Random) Decision Trees to Random Forests

Advantages

- Can deal with very heterogeneous data
  → Different, data-specific types of node tests
- Not prone to the curse of dimensionality
  → Each node only works on a very limited set of dimensions
- Very efficient
  → Each sample passes maximal H nodes (H = maximal height)
- Easy to implement
  → Binary trees are one of the most basic data structures
- Easy to interpret
  → Path through tree is a connected set of decision rules
- Well understood
  → Theoretical and practical implications of design decisions have been researched for more than 4 decades

From (Random) Decision Trees to Random Forests

Disadvantages

- Optimized by greedy algorithms

  → A chain of individually optimal decisions, might not lead to an overall optimum

From (Random) Decision Trees to Random Forests

Disadvantages

- Optimized by greedy algorithms
  → A chain of individually optimal decisions, might not lead to an overall optimum
- The optimal solution (i.e. decision boundary) might not be part of the model class (e.g. piece-wise linear and axis-aligned functions)

From (Random) Decision Trees to Random Forests

Disadvantages

- Optimized by greedy algorithms
  → A chain of individually optimal decisions, might not lead to an overall optimum
- The optimal solution (i.e. decision boundary) might not be part of the model class (e.g. piece-wise linear and axis-aligned functions)
- Prone to overfitting

From (Random) Decision Trees to Random Forests

Disadvantages

- Optimized by greedy algorithms
  → A chain of individually optimal decisions, might not lead to an overall optimum
- The optimal solution (i.e. decision boundary) might not be part of the model class (e.g. piece-wise linear and axis-aligned functions)
- Prone to overfitting
- Model capacity depends on amount of data
  → Few samples = small trees: Only few questions can be asked.
  → Many samples (might) lead to very high trees: Long processing times, large memory footprint.

From (Random) Decision Trees to Random Forests

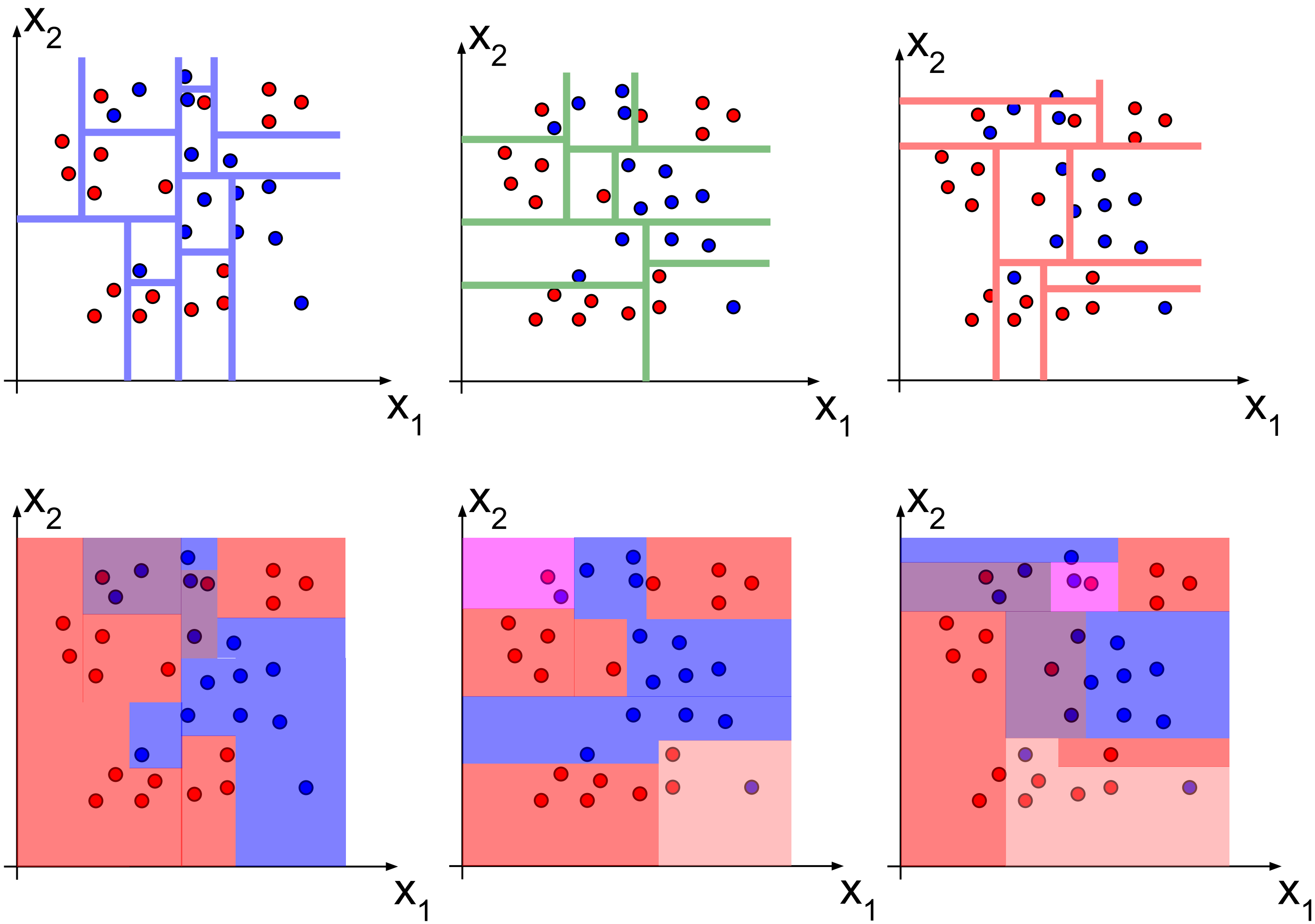
Disadvantages

- Optimized by greedy algorithms
  → A chain of individually optimal decisions, might not lead to an overall optimum
- The optimal solution (i.e. decision boundary) might not be part of the model class (e.g. piece-wise linear and axis-aligned functions)
- Prone to overfitting
- Model capacity depends on amount of data
  → Few samples = small trees: Only few questions can be asked.
  → Many samples (might) lead to very high trees: Long processing times, large memory footprint.

How to
→ keep (most) of the advantages
→ getting rid of (most) disadvantages?

From (Random) Decision Trees to Random Forests

- Individual Random Trees

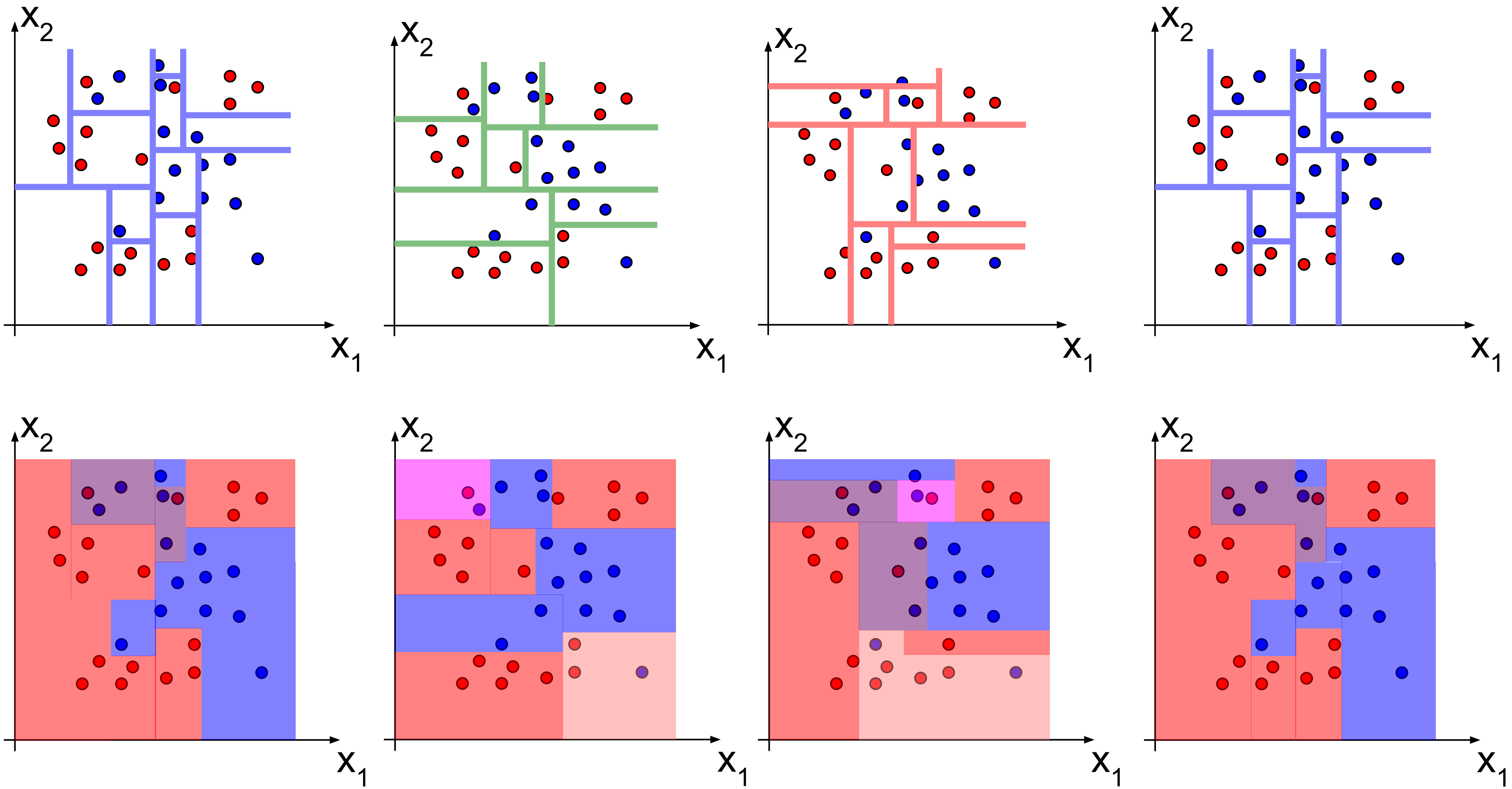


DLR

From (Random) Decision Trees to Random Forests

– Individual Random Trees

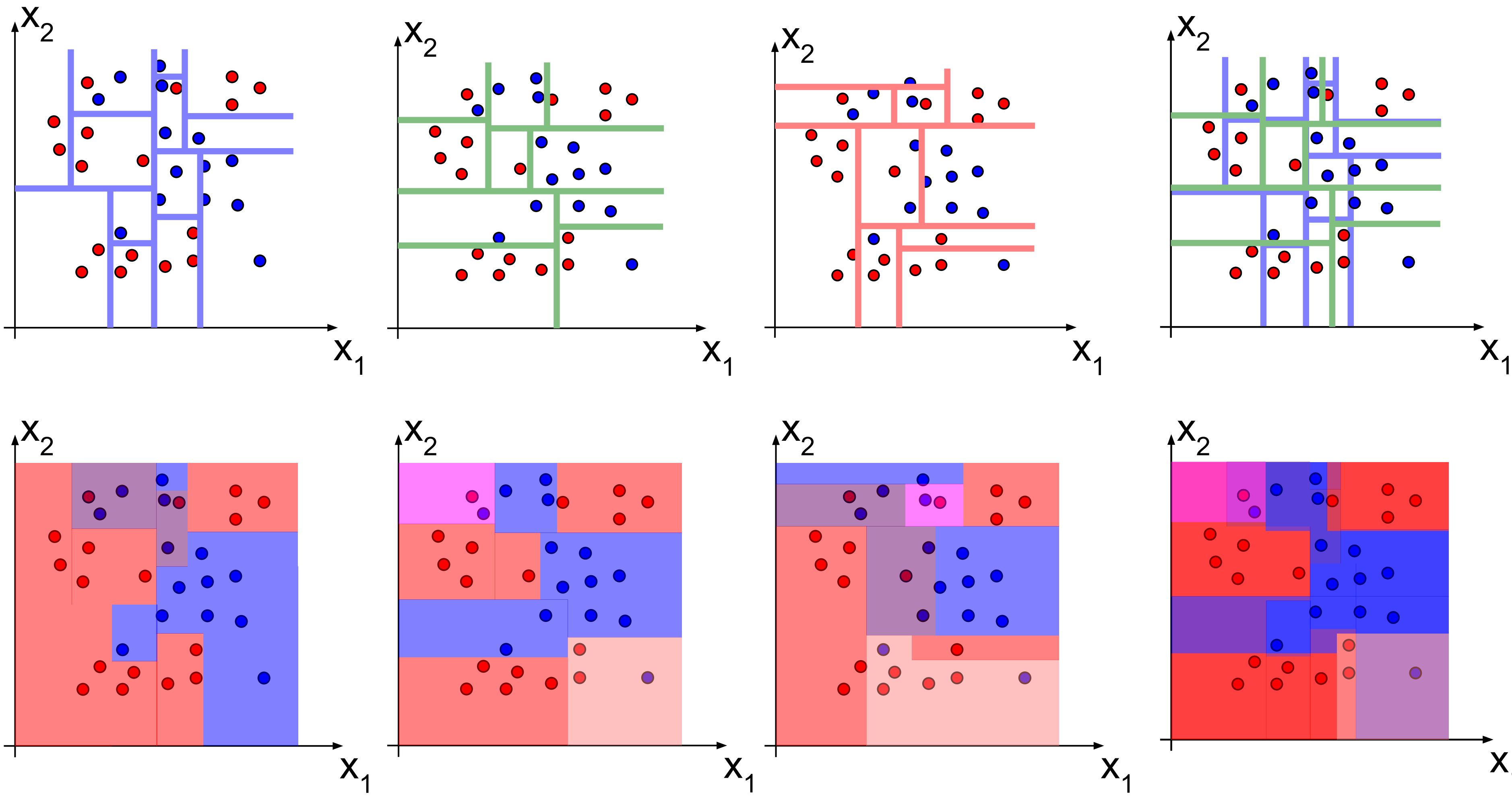
RF (T=1)



From (Random) Decision Trees to Random Forests

– Individual Random Trees

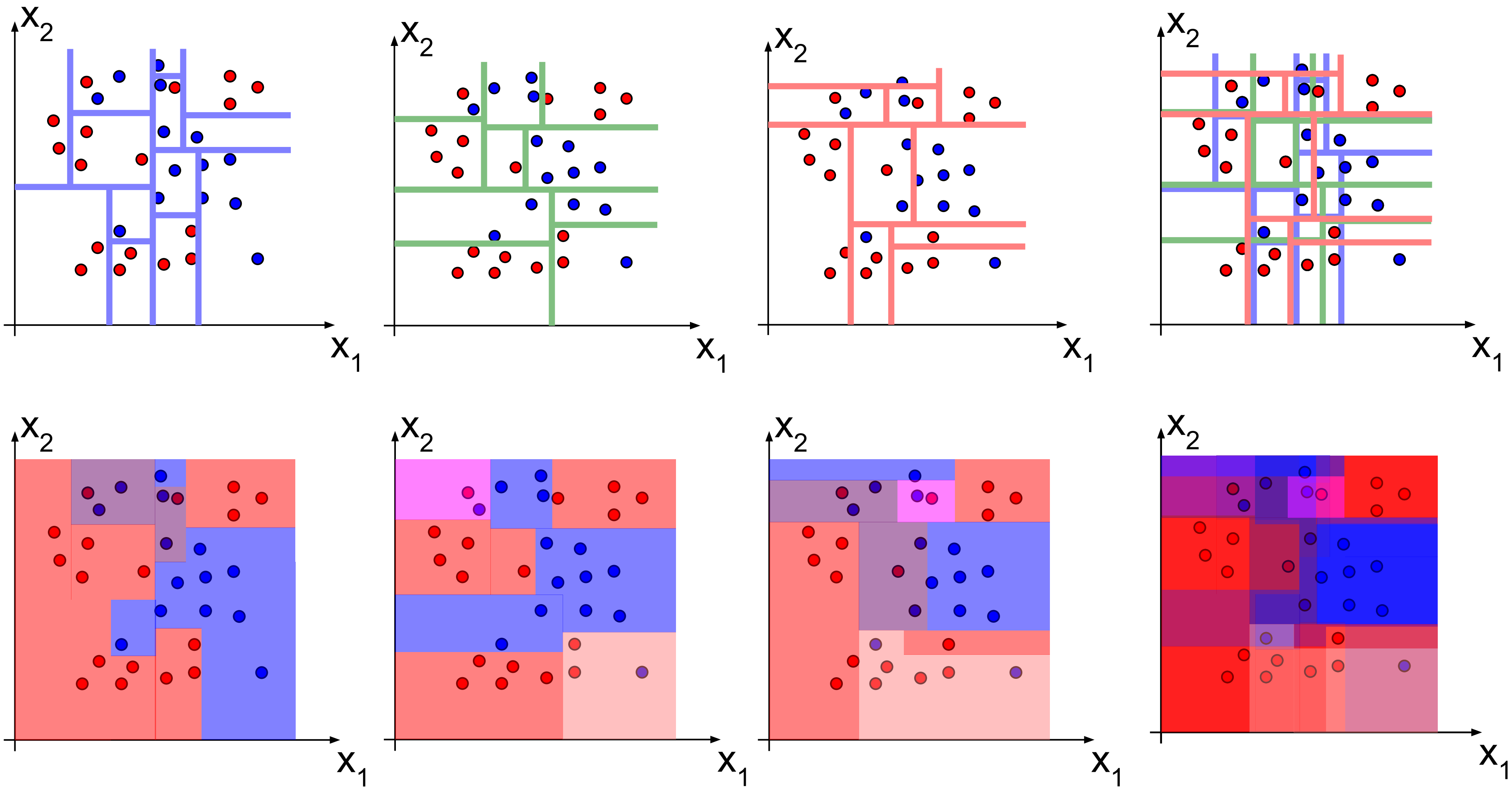
RF (T=2)



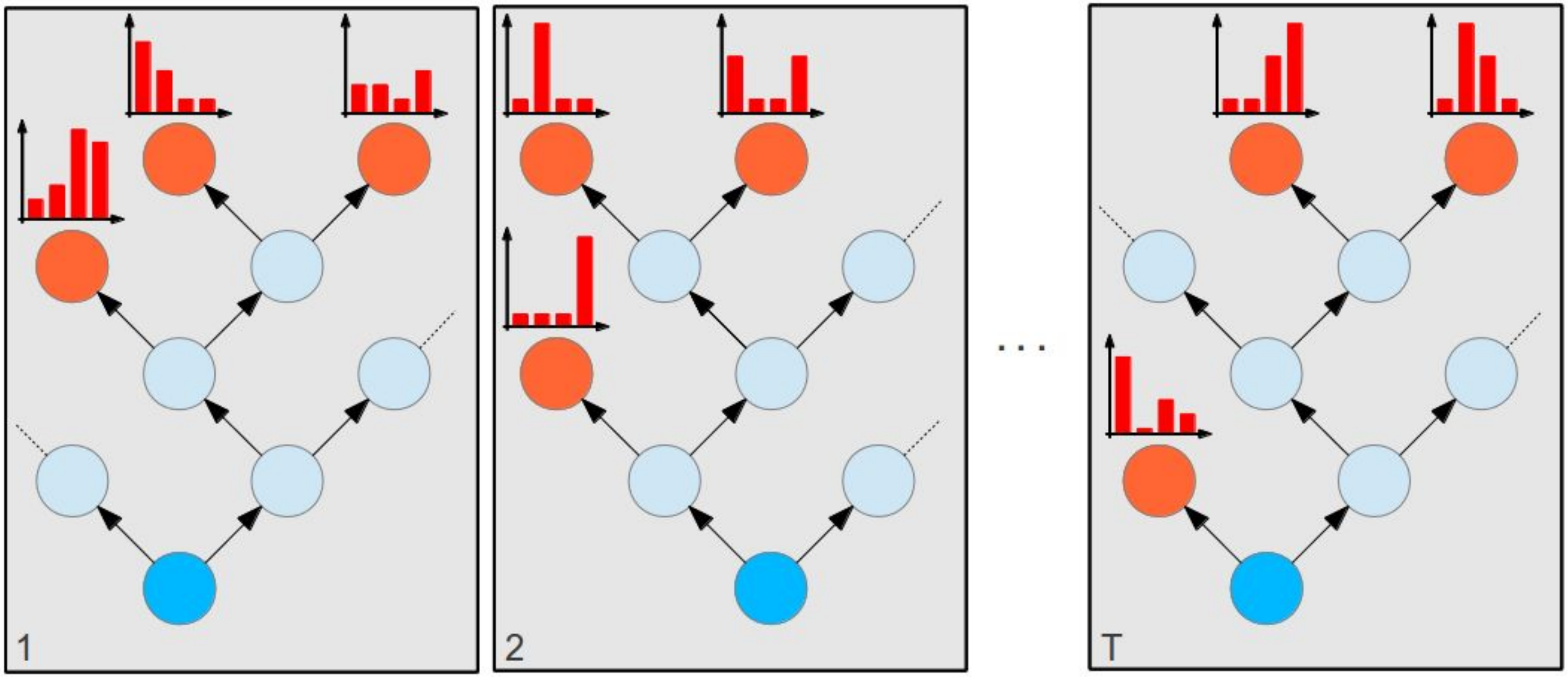
From (Random) Decision Trees to Random Forests

– Individual Random Trees

RF (T=3)



Random Forests



Set of decision trees

- Each tree *t* generated from training data
- Creation of one tree independent of all other trees
- Based on random processes to produce diverse set of trees
- Individual tree outcomes are fused (voting, averaging, …)

Random Forests

- Many (suboptimal) baselearners, i.e. decision trees
- Combined output on average better than individual output
- Minimization of the risk to use wrong model
- Extension of the model space
- Decreased dependence on initialization
- One name to rule them all
  - Bagged Decision Trees
  - Randomized Trees
  - Decision Forests
  - ERT, PERT, Rotation Forests, Canonical Correlation Forests, Hough Forests, Semantic Texton Forests, ...

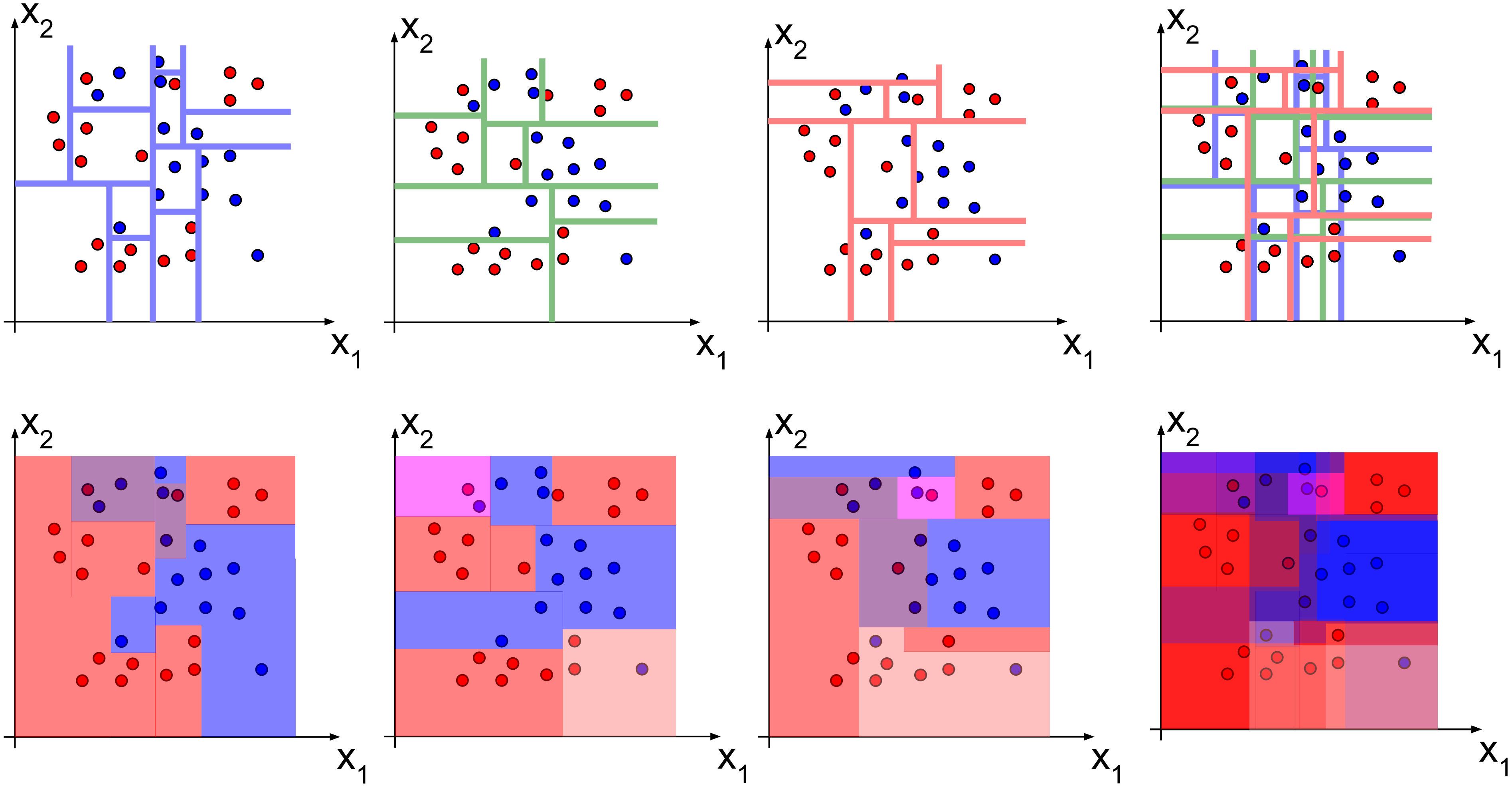
Random Forests - Key questions

- **Why randomization?**
  → How to achieve a diverse and strong ensemble?
- What kind of node tests?
  → For images, for other data spaces than R^n
- How to select node tests?
  → How to measure good tests?
- What kind of target variables?
  → More than a single class label?
- How to limit model capacity (tree height, tree number)?
  → The more the better? What about overfitting?
- How to fuse tree decisions?
  → Whom to trust?
- How to interpret results?
  → Tree properties and visualization.

Random Forests - Why randomization?

– Individual Random Trees

RF (T=3)



DLR

Random Forests - Why randomization?

Generalization error

$$PE \leq \frac{\bar{\rho}(1 - s^2)}{s^2}$$

DLR

Random Forests - Why randomization?

Generalization error

Avg. tree strength

$$PE \leq \frac{\bar{\rho}(1 - s^2)}{s^2}$$

- The stronger the trees (large *s*), the stronger the ensemble!

DLR

Random Forests - Why randomization?

Generalization error

Avg. tree correlation

Avg. tree strength

$$PE \leq \frac{\bar{\rho}(1 - s^2)}{s^2}$$

- The stronger the trees (large *s*), the stronger the ensemble!

- The more correlated the trees (large ρ), the weaker the ensemble!

[Difference between asking 10 persons 1 time, or 1 person 10 times.]

Random Forests - Randomization through Bagging

Given: Training set D with |D| = N samples.

Bagging (Bootstrap aggregating):
1. Randomly sample M data sets D_m with replacement ($|D_m| = N$).
2. Train M models where m-th model has only access to m-th dataset.
3. Average all models.

Random Forests - Randomization through Bagging

Given: Training set D with |D| = N samples.

Bagging (Bootstrap aggregating):
1. Randomly sample M data sets D_m with replacement ($|D_m| = N$).
2. Train M models where m-th model has only access to m-th dataset.
3. Average all models.

Meta learning technique
- Works if small change in input data leads to large model variation
- Reduces variance (of final model), avoids overfitting.
- Leads to diverse decision trees, even if all other parameters are fixed
- Variant: Subagging ≡ Sample without replacement
- Disadvantage: Less samples per tree (yet forest does see all samples)

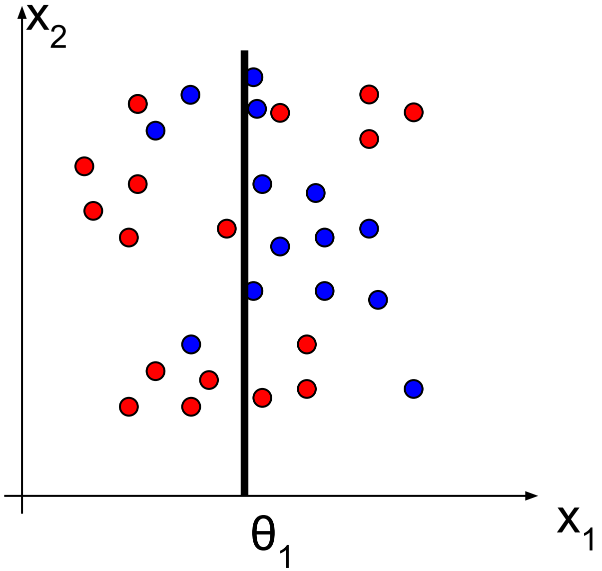
Random Forests - Randomization through node tests

Per tree:

- Use randomized projections into subspaces (e.g. subset, PCA, LDA, …)

Per node:

- Select a feature randomly
- Select threshold randomly



→ Works only if

- Many features are available
- Each feature has many possible values

→ Will prefer features with many values (e.g. real values) over features with few values (e.g. categorical variables)

Random Forests - Key questions

- Why randomization?
  → How to achieve a diverse and strong ensemble?
- **What kind of node tests?**
  → For images, for other data spaces than $\mathbf{R}^n$
- How to select node tests?
  → How to measure good tests?
- What kind of target variables?
  → More than a single class label?
- How to limit model capacity (tree height, tree number)?
  → The more the better? What about overfitting?
- How to fuse tree decisions?
  → Whom to trust?
- How to interpret results?
  → Tree properties and visualization.

RF - Holistic Feature Selection and Classification

Fully-polarimetric

- Subaperture
- Intensity in different channels
- Decompositions
- Eigenvalue based features
- Correlation among channels
- Physica classification

Color

- Normalised RGB Values
- HSV

Grayscale

- Local statistics
- Gradient histogram
- Gray-Level Coocurrence Matrix

Single-polarimetric

- Speckle-based texture

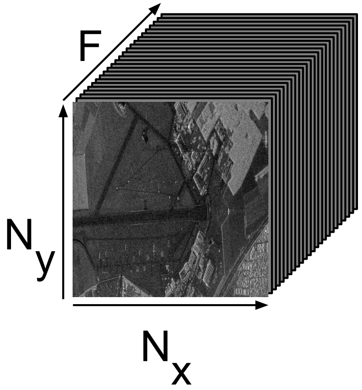
Binary

- Distance transformation

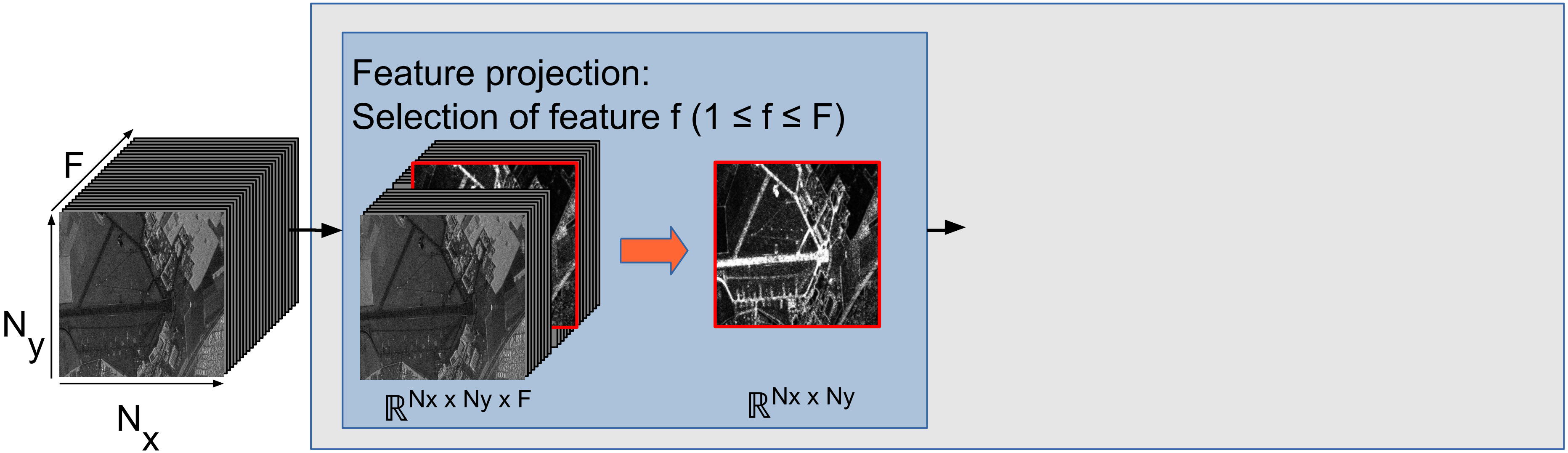
Generic object categorization in PolSAR images - and beyond, Hänsch, R., 2014.

DLR

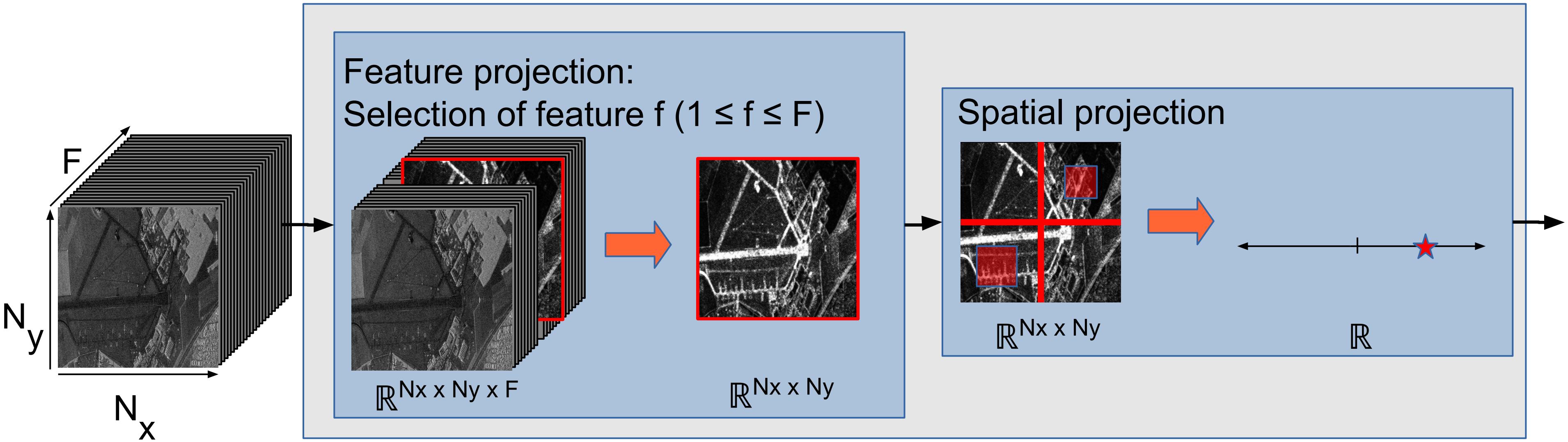
RF - Holistic Feature Selection and Classification



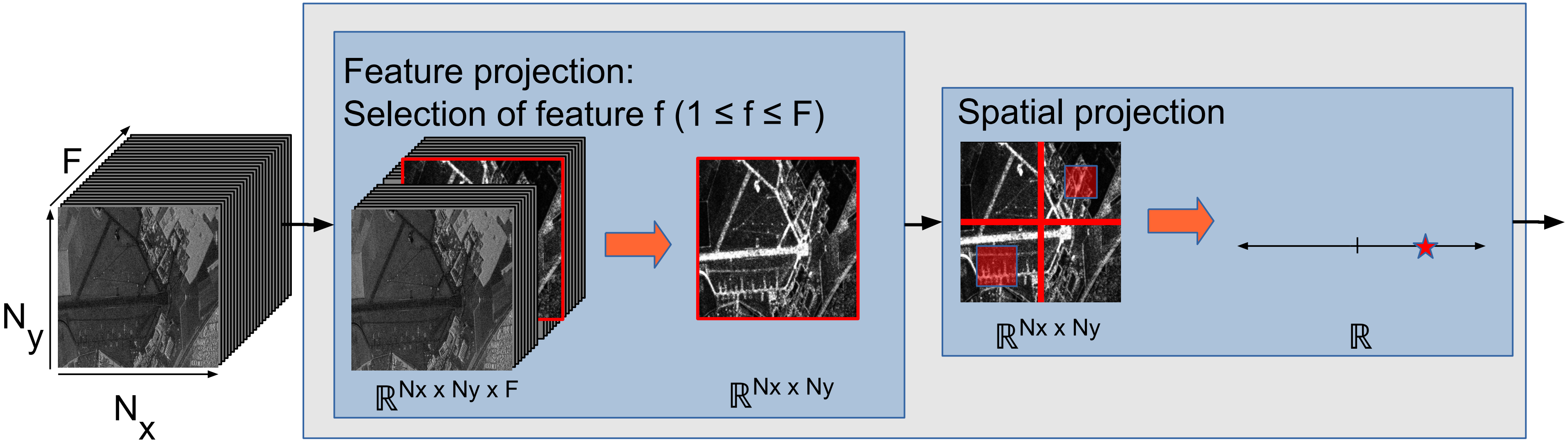
RF - Holistic Feature Selection and Classification



RF - Holistic Feature Selection and Classification

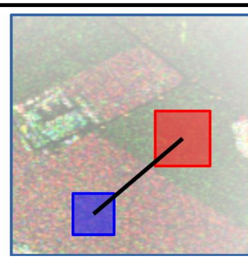


RF - Holistic Feature Selection and Classification



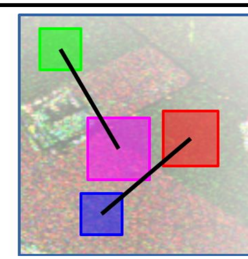
1-Point projection

$op(R_1) - op(R)$



2-Point projection

$op(R_1) - op(R_2)$

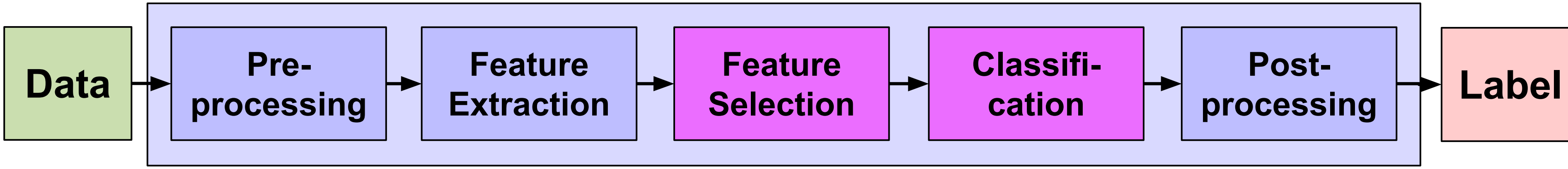


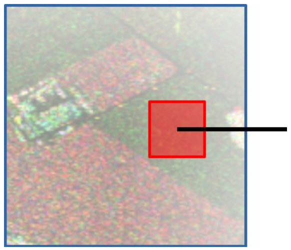
4-Point projection

$(op(R_1) - op(R_2)) - (op(R_3) - op(R_4))$

DLR

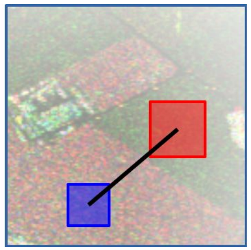
RF - Holistic Feature Selection and Classification





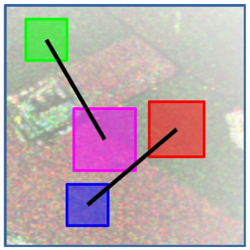
1-Point projection

$op(R_1) - op(R)$



2-Point projection

$op(R_1) - op(R_2)$



4-Point projection

$(op(R_1) - op(R_2)) - (op(R_3) - op(R_4))$

op **: Patch → Pixel (Scalar)**

- Max. / min. value
- Central pixel
- Average

RF - Holistic Feature Selection and Classification

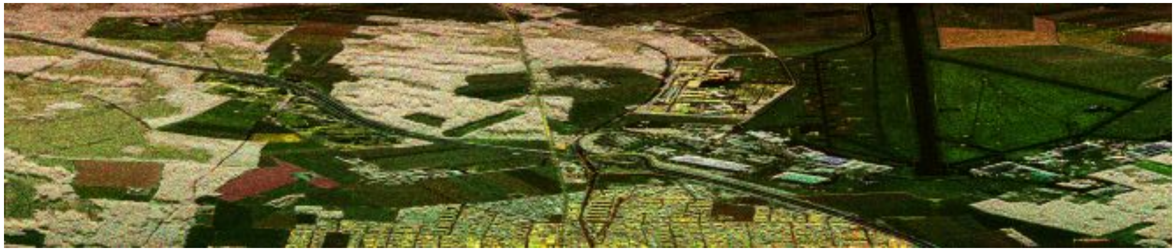
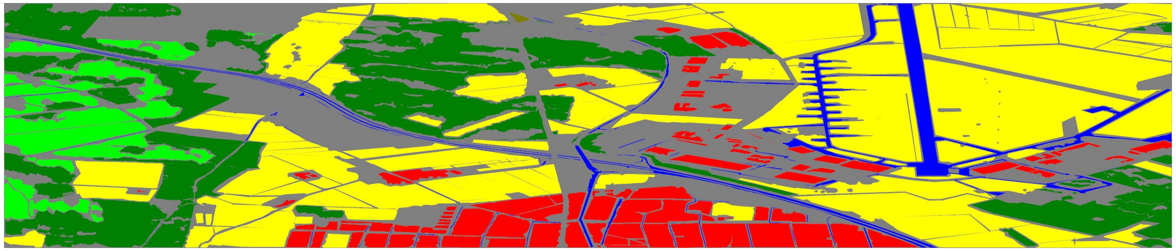
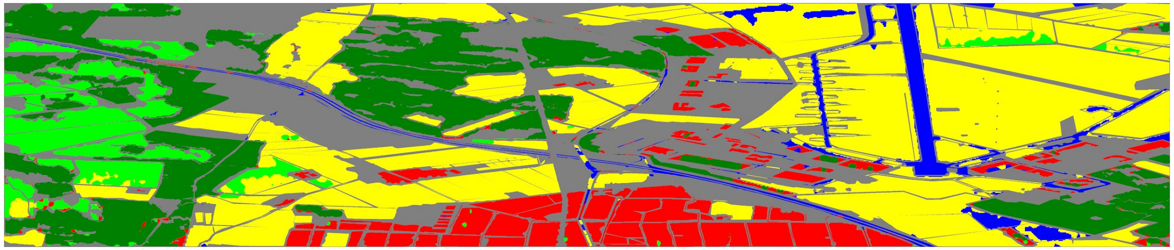


Image data
- Oberpfaffenhofen data set
- fully polarimetric
- E-SAR, DLR



Reference data

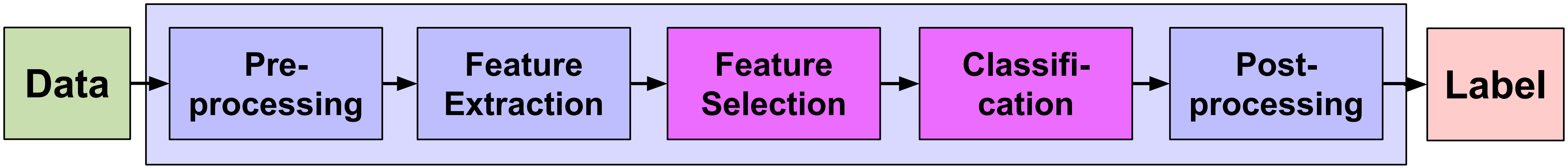


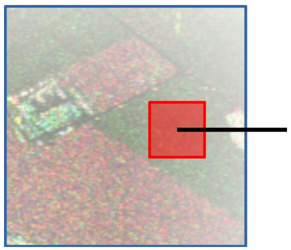
ProB-RF

BA = 89.4%	Urban	Forest	Field	Shrubl.	Road
Urban	0.94	0.05	0.00	0.00	0.01
Forest	0.02	0.97	0.00	0.01	0.00
Field	0.00	0.00	0.94	0.04	0.02
Shurbl.	0.02	0.03	0.06	0.89	0.00
Road	0.11	0.01	0.14	0.01	0.73

DLR

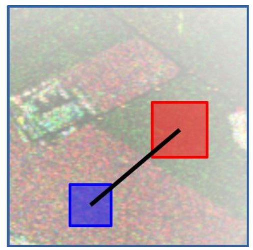
RF - Holistic Feature Selection and Classification





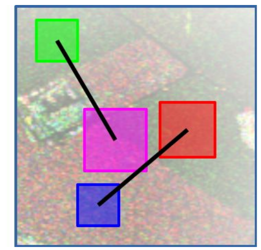
1-Point projection

$op(R_1) - op(R)$



2-Point projection

$op(R_1) - op(R_2)$



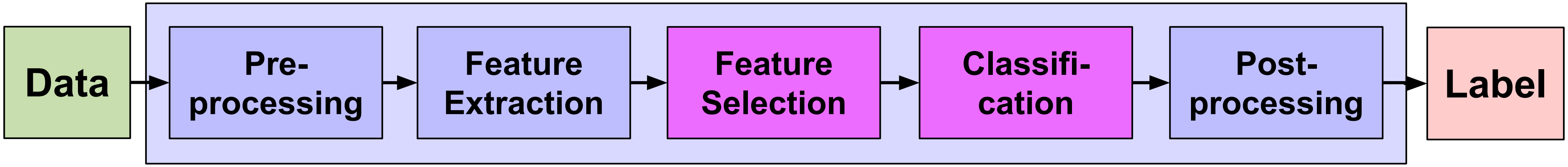
4-Point projection

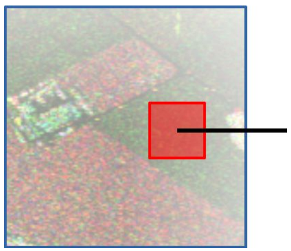
$(op(R_1) - op(R_2)) - (op(R_3) - op(R_4))$

op **: Patch → Pixel (Scalar)**

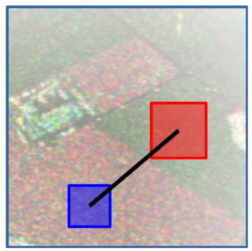
- Max. / min. value
- Central pixel
- Average

RF - Holistic Feature Selection and Classification

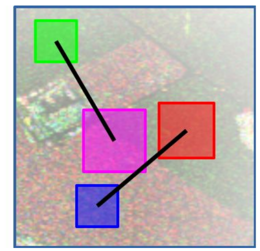




1-Point projection
$d(op(R_1), op(R))$



2-Point projection
$d(op(R_1), op(R_2))$

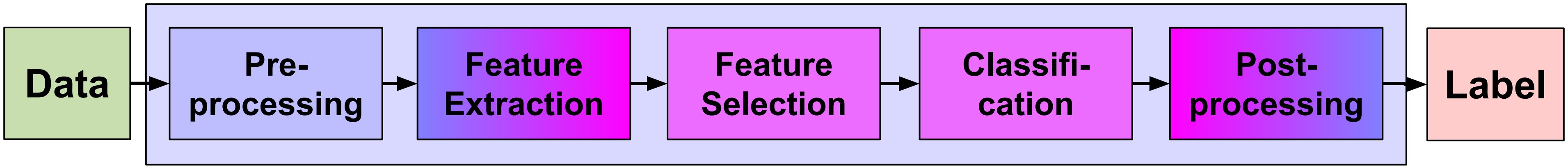


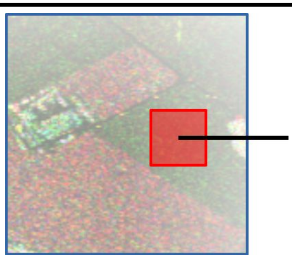
4-Point projection
$d(op(R_1), op(R_2)) - d(op(R_3), op(R_4))$

op **: Patch → Pixel (Scalar)**
- Max. / min. value
- Central pixel
- Average

d **: Scalar** x **Scalar→ Scalar**
- Signed / absolute difference

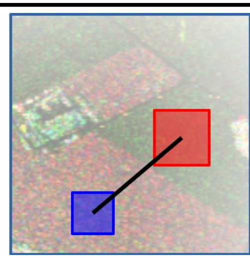
RF - Holistic Feature Extraction and Classification





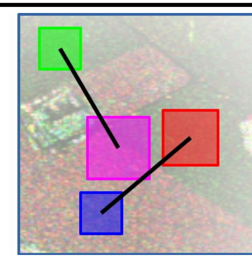
1-Point projection

$$d(op(R_1), op(R))$$



2-Point projection

$$d(op(R_1), op(R_2))$$



4-Point projection

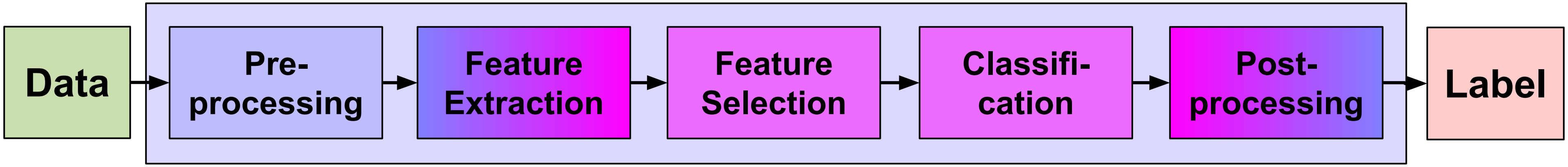
$$d(op(R_1), op(R_2)) - d(op(R_3), op(R_4))$$

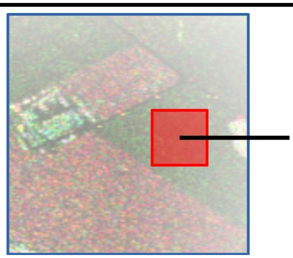
op : **Patch → Pixel (3-Vector)**
- Max. / min. grey value
- Central pixel
- Average

d : **3-Vector** x **3-Vector → Scalar**
- Euclidean distance in any color space
- Difference in hue

DLR

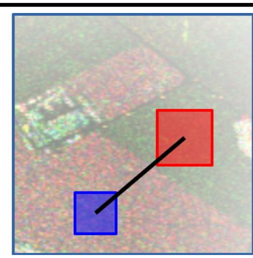
RF - Holistic Feature **Extraction** and Classification





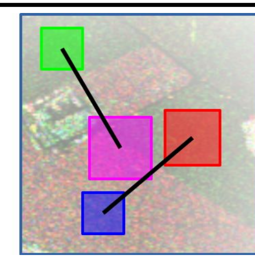
1-Point projection

$d(op(R_1), op(R))$



2-Point projection

$d(op(R_1), op(R_2))$



4-Point projection

$d(op(R_1), op(R_2)) - d(op(R_3), op(R_4))$

op : **Patch → Pixel (Matrix)**

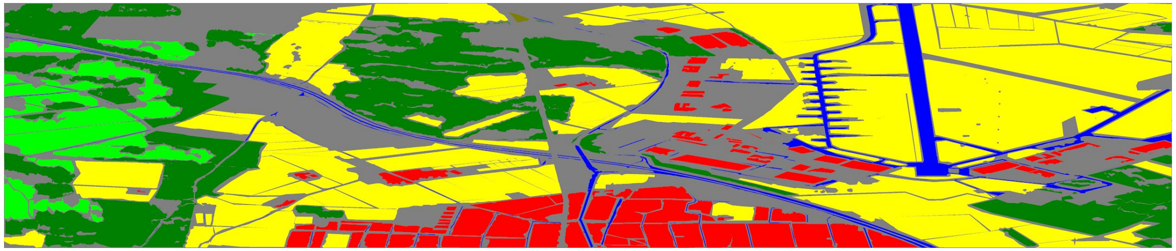
- Max. / min. span
- Central pixel
- Average

d : **Matrix** x **Matrix → Scalar**

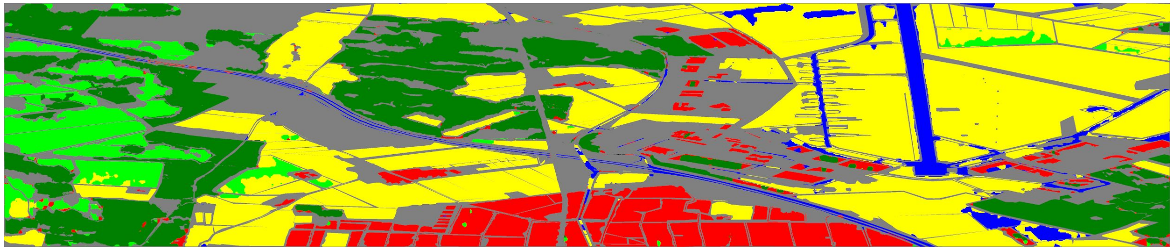
- Difference of polarimetric features
- General matrix distances
- Polarimetric distance measures

Skipping the real world: Classification of PolSAR images without explicit feature extraction,
R. Hänsch, O. Hellwich, ISPRS Journal of Photogrammetry and Remote Sensing, 2017

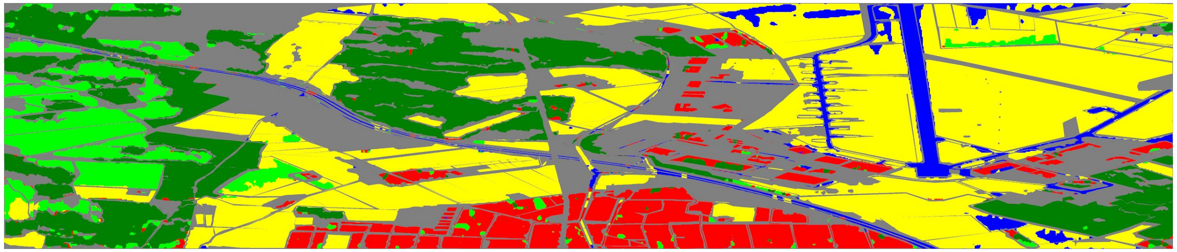
RF - Holistic Feature Extraction and Classification



Reference data



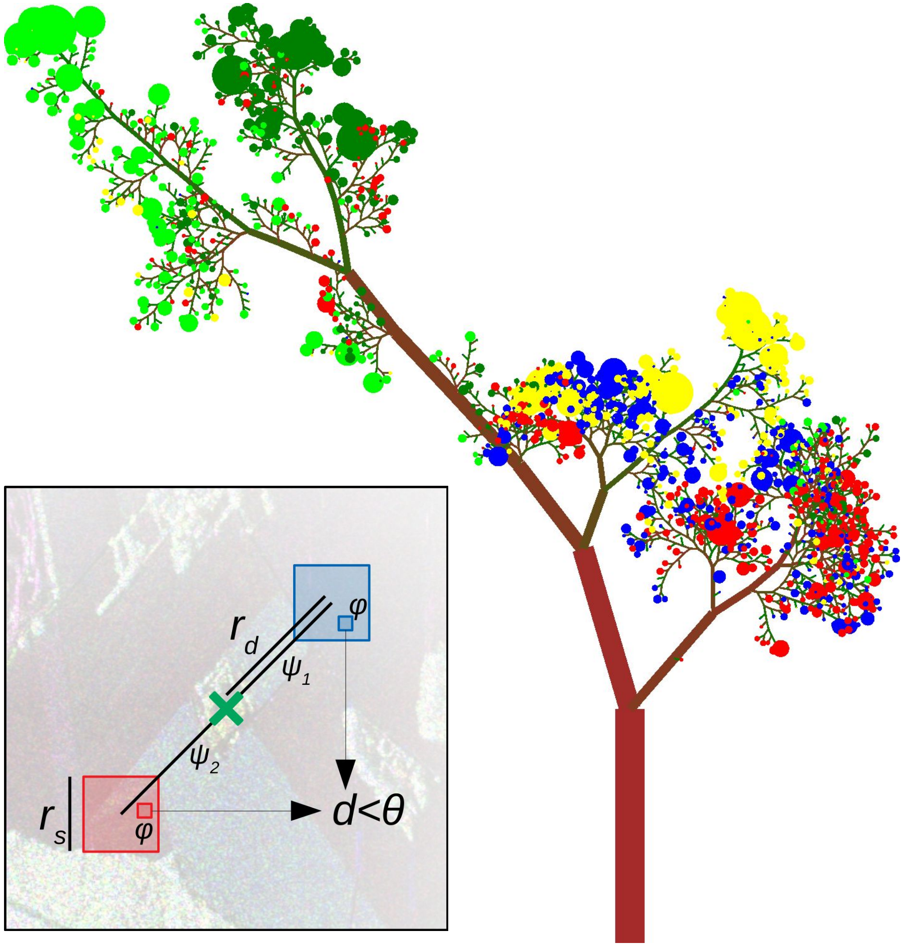
RF with
explicit feature extraction
BA = 89.4%



RF without
explicit feature extraction
BA = 87.5%

Summary: Projection-based Random Forests

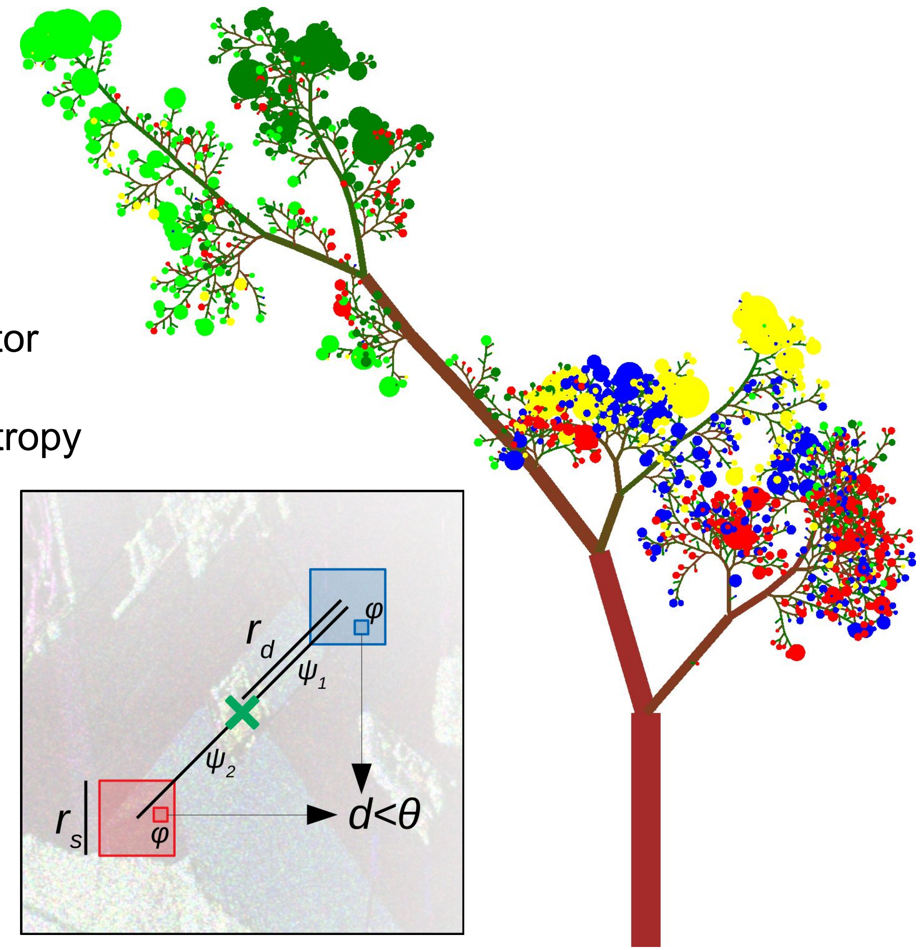
1. ψ: Select regions within a patch
→ Random size and position



Summary: Projection-based Random Forests

1. ψ: Select regions within a patch
   → Random size and position

2. φ: Select / compute pixel value
   → Random, data type dependent operator
   → HS signature: e.g. min/max power
   → Pol. cov. matrix: e.g. min/max pol. entropy

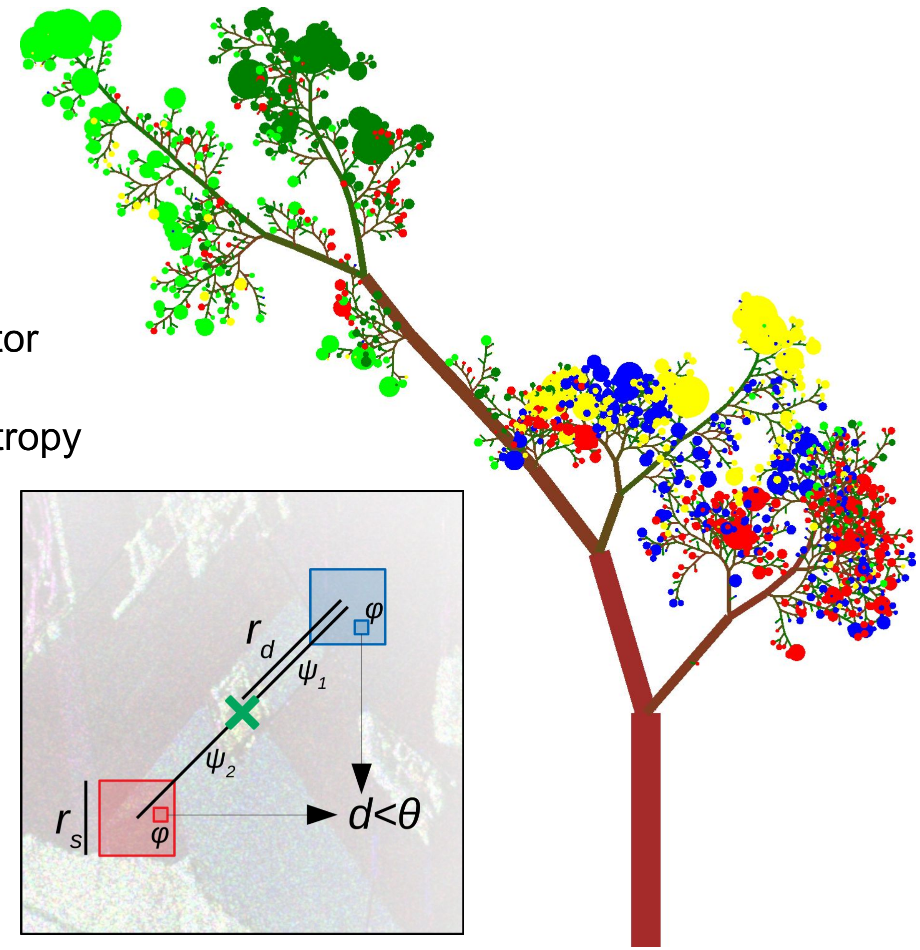


Summary: Projection-based Random Forests

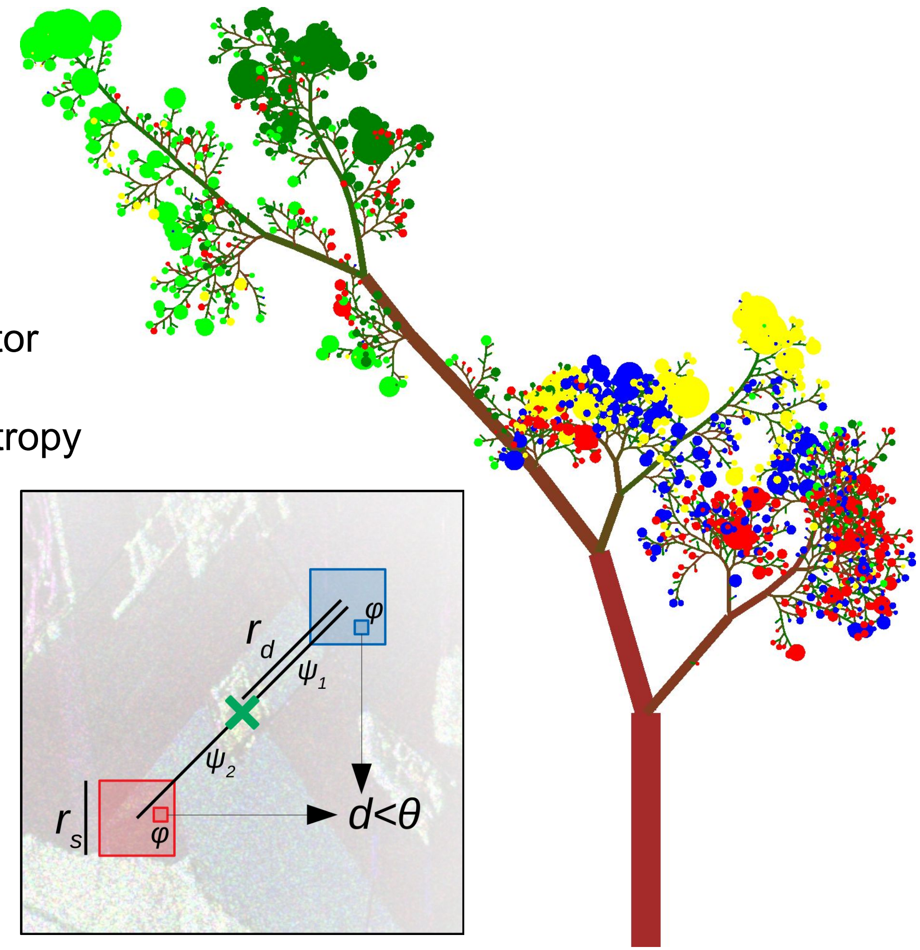
1. ψ: Select regions within a patch
   → Random size and position

2. φ: Select / compute pixel value
   → Random, data type dependent operator
   → HS signature: e.g. min/max power
   → Pol. cov. matrix: e.g. min/max pol. entropy

3. d: Apply distance measure
   → Randomly selected
   → Data type dependent
   → HS signature: e.g. cosine similarity
   → Pol. cov. matrix: e.g. Bartlett distance



Summary: Projection-based Random Forests



1. ψ: Select regions within a patch
   → Random size and position

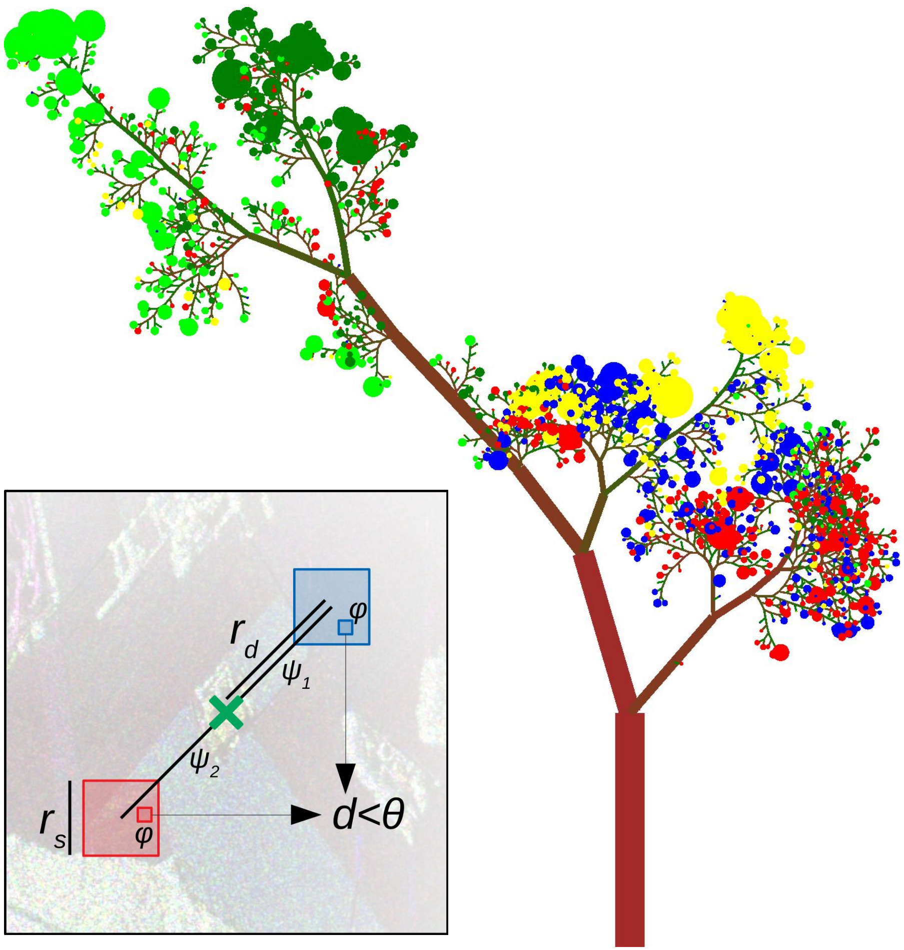
2. φ: Select / compute pixel value
   → Random, data type dependent operator
   → HS signature: e.g. min/max power
   → Pol. cov. matrix: e.g. min/max pol. entropy

3. d: Apply distance measure
   → Randomly selected
   → Data type dependent
   → HS signature: e.g. cosine similarity
   → Pol. cov. matrix: e.g. Bartlett distance

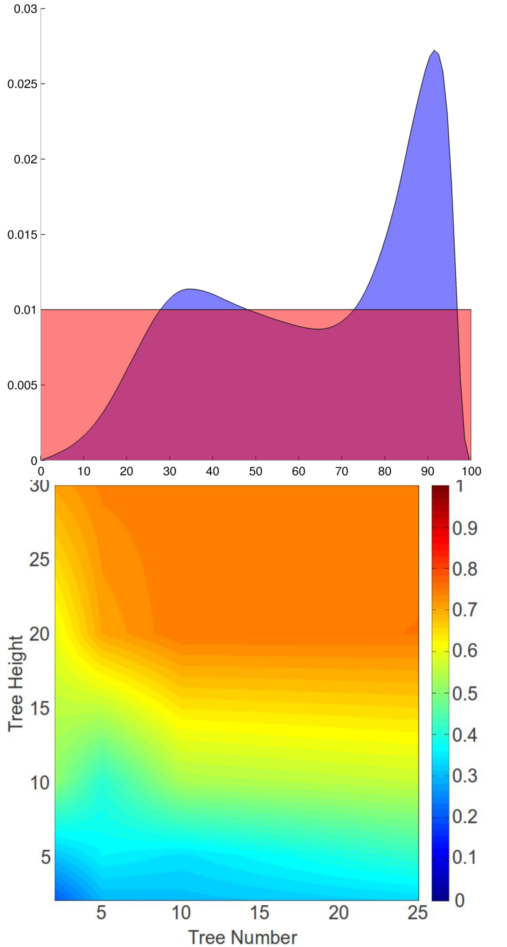
4. Compare to scalar (split threshold)

Summary: Projection-based Random Forests

- Can be directly applied to any kind of data
- Learns features directly from the data
- Project local patches into scalars
- Direct connection between scale of the projection and access to context

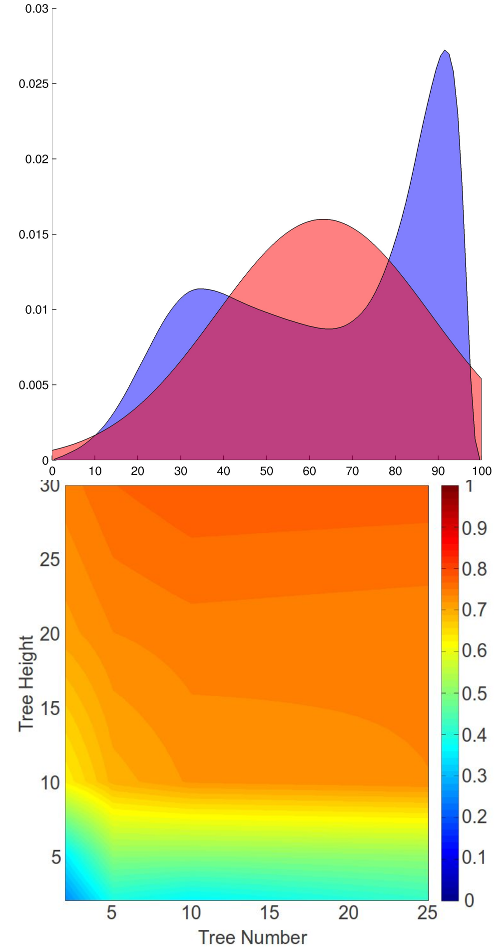


Random Forests - Split point selection - Unsupervised



Uniform sampled

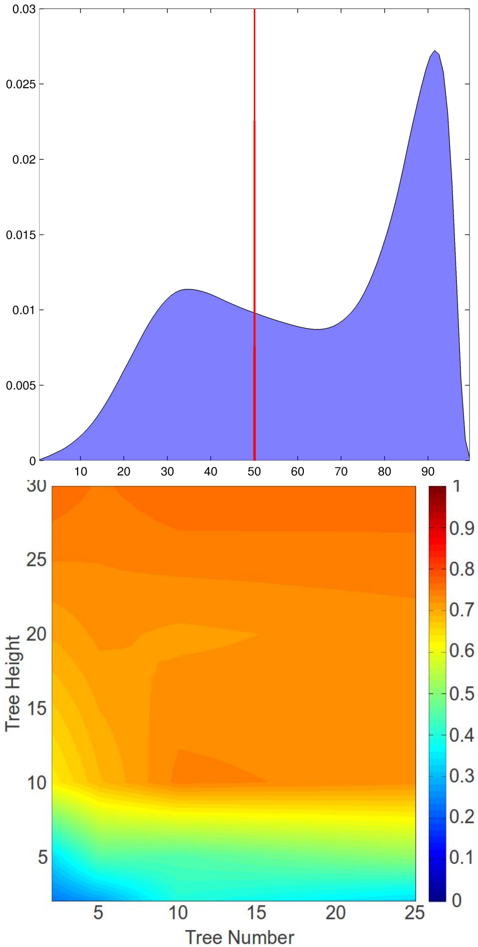
$$\theta \sim U(\min(\hat{D}), \max(\hat{D}))$$



Gaussian sampled

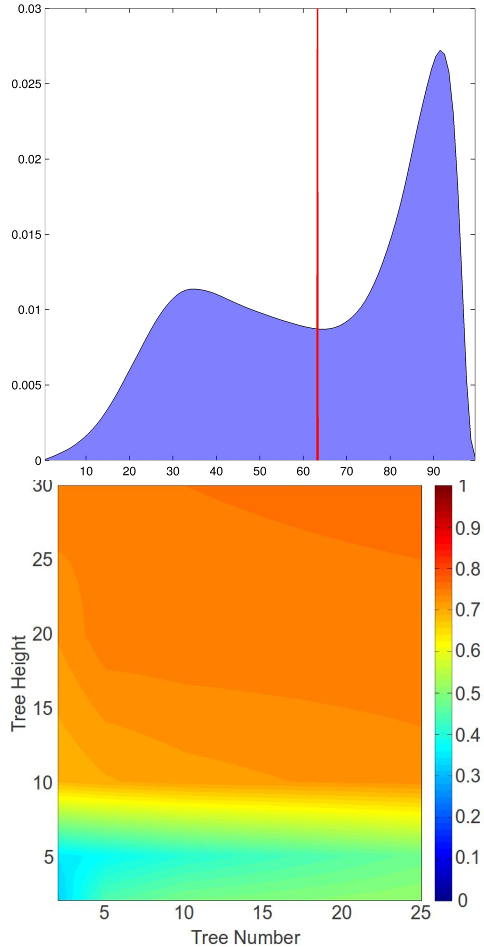
$$\theta \sim N(\mu_{\hat{D}}, \sigma_{\hat{D}})$$

Random Forests - Split point selection - Unsupervised



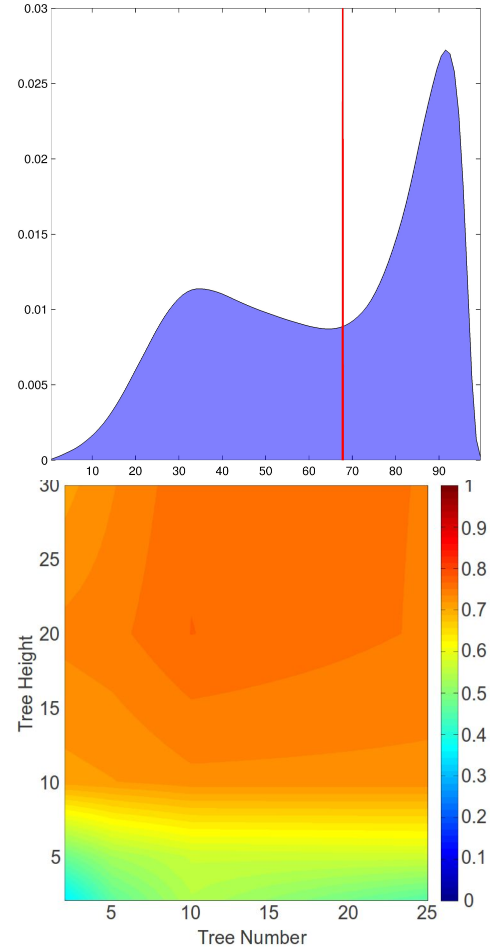
Interval center

$$\theta = \frac{\min(\hat{D}) + \max(\hat{D})}{2}$$



Mean value

$$\theta = \frac{1}{|\hat{D}|} \sum_{\hat{x} \in \hat{D}} \hat{x}$$



Median value

$$\theta = median(\hat{D})$$

DLR

Random Forests - Split point selection - Supervised

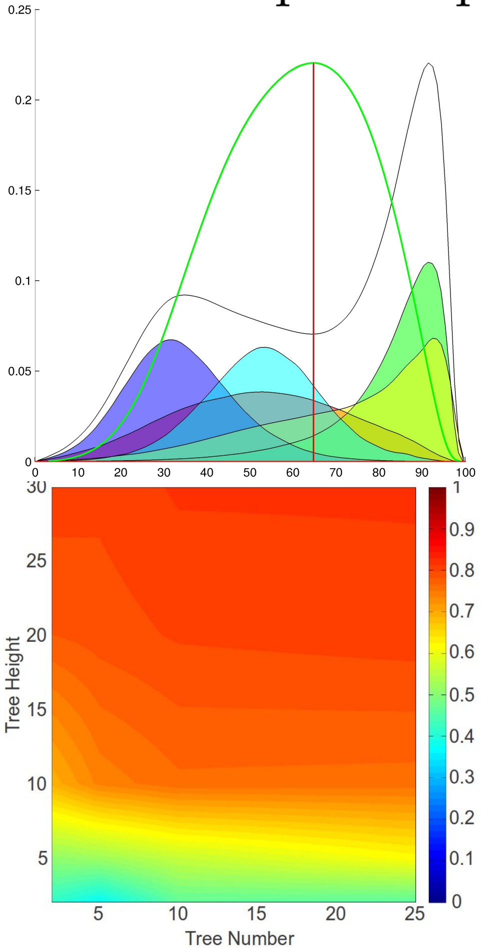
$$\text{Max. drop of impurity: } \theta = \arg\min_{\hat{\theta}} \left[I(n) - P_L I(n_L) - P_R I(n_R) \right]$$

n … Set of samples in current node
$n_{L/R}$ … Set of samples in left / right child node
$P_{L/R}$ … Fraction of samples that are in left / right child node
I … A measure of impurity

→ Find a test function that splits the data into two subsets that are as “pure” as possible regarding the class distribution (i.e. contain only samples of a single class in the best case)

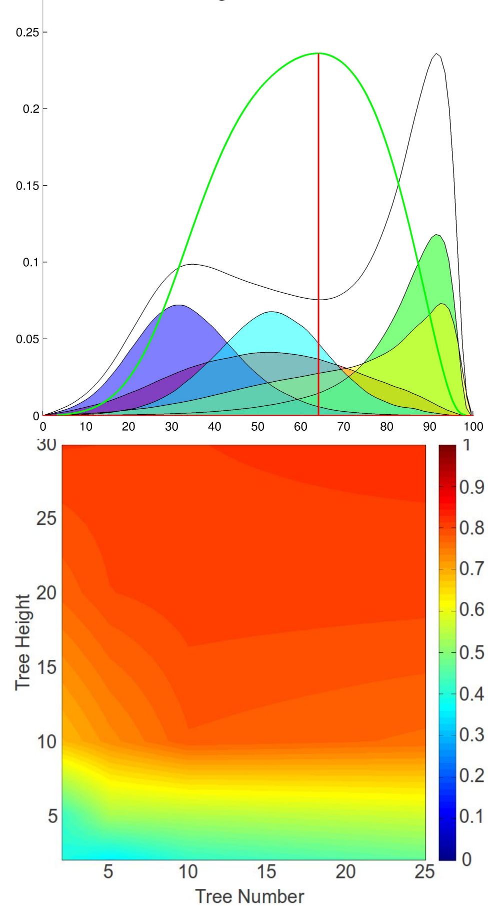
Random Forests - Split point selection - Supervised

Max. drop of impurity: $\theta = \arg\min_{\hat{\theta}} \left[I(n) - P_L I(n_L) - P_R I(n_R) \right]$



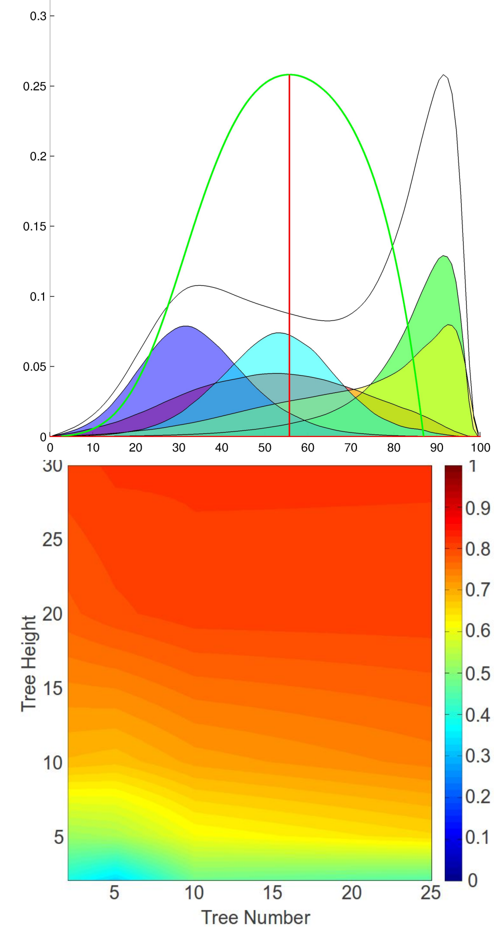
Entropy

$$I(n) = -\sum_c P(c|n) \log(P(c|n))$$



Gini

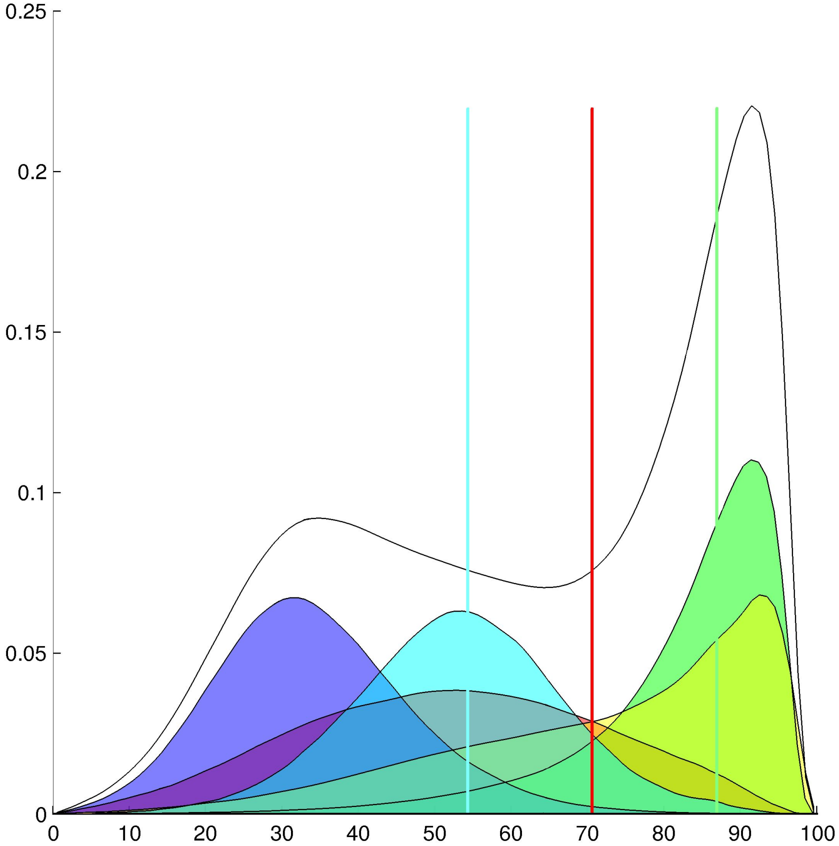
$$I(n) = 1 - \sum_c P(c|n)^2$$



Misclassification

$$I(n) = 1 - \max_c P(c|n)$$

Random Forests - Split point selection



- Other possibilities available
  → Intervals, structured label spaces,
  inter-class split
- Need for computational efficiency since selection is performed thousand to million times during training
- Avoid exhaustive search

Random Forests - Key questions

- Why randomization?
  → How to achieve a diverse and strong ensemble?
- What kind of node tests?
  → For images, for other data spaces than R^n
- **How to select node tests?**
  → How to measure good tests?
- What kind of target variables?
  → More than a single class label?
- How to limit model capacity (tree height, tree number)?
  → The more the better? What about overfitting?
- How to fuse tree decisions?
  → Whom to trust?
- How to interpret results?
  → Tree properties and visualization.

Random Forests - Node optimization

- Generate m split candidates
  → “Traditionally”: $m = \sqrt{d}$, where d is data dimension
  → “Modern” approaches: $m \approx 10^5$
  → Usually even $m = 2$ leads to performance increase
  → Trade-off between high performance and high correlation

Random Forests - Node optimization

- Generate m split candidates
  → "Traditionally": $m = \sqrt{d}$, where d is data dimension
  → "Modern" approaches: $m \approx 10^5$
  → Usually even $m = 2$ leads to performance increase
  → Trade-off between high performance and high correlation

- Select best split, reject all others

Random Forests - Node optimization

- Generate m split candidates
  → "Traditionally": $m = \sqrt{d}$, where d is data dimension
  → "Modern" approaches: $m \approx 10^5$
  → Usually even $m = 2$ leads to performance increase
  → Trade-off between high performance and high correlation

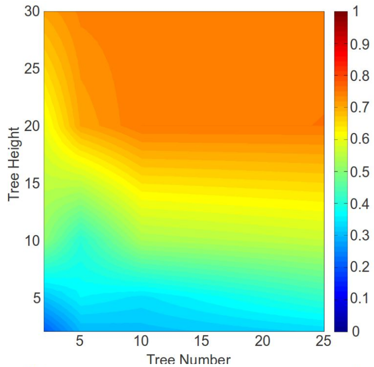
- Select best split, reject all others

- Measure optimality of a split
  → Classification: "Purity" of child nodes (e.g. Gini, entropy, etc.)
  → Regression: e.g. variance
  → In general: How much better is the estimation of the child nodes (as a weighted average) than parent nodes?

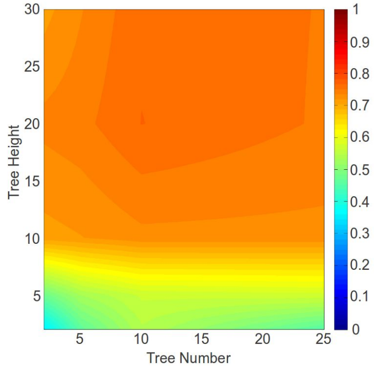
DLR

Random Forests - Node optimization

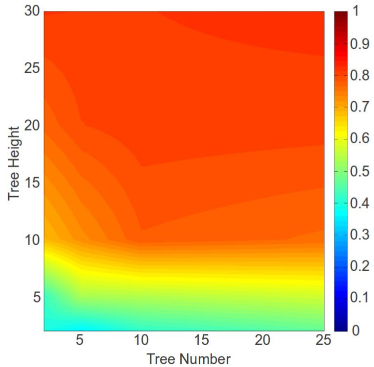
Uniform:



Median:



Gini:



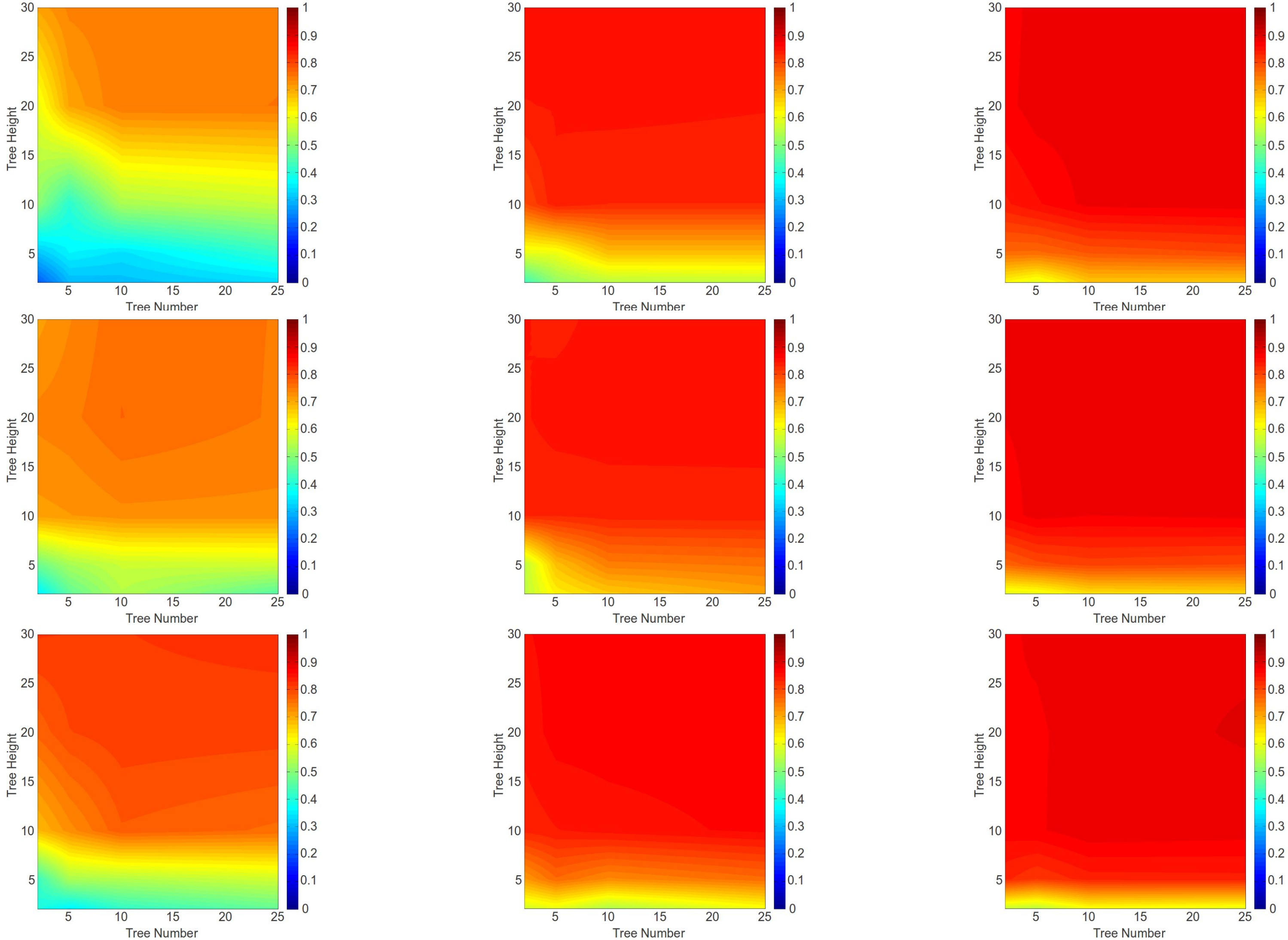
m=1

Random Forests - Node optimization

Uniform:

Median:

Gini:



m=1 m=10 m=100

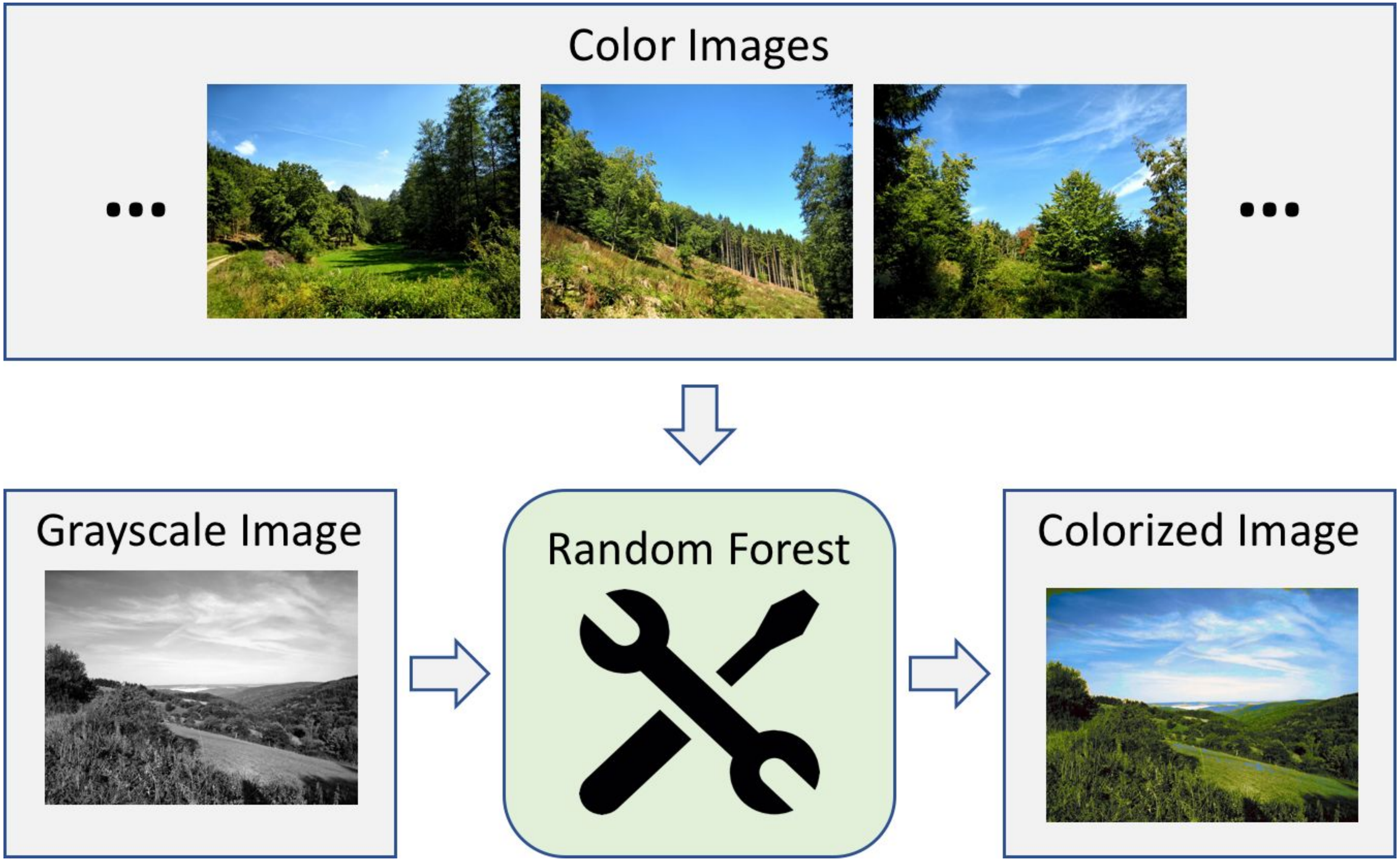
DLR

Random Forests - Key questions

- Why randomization?
  → How to achieve a diverse and strong ensemble?
- What kind of node tests?
  → For images, for other data spaces than R^n
- How to select node tests?
  → How to measure good tests?
- **What kind of target variables?**
  → More than a single class label?
- How to limit model capacity (tree height, tree number)?
  → The more the better? What about overfitting?
- How to fuse tree decisions?
  → Whom to trust?
- How to interpret results?
  → Tree properties and visualization.

Random Forests - Regression: Colorization

Sensor to sensor transcoding, e.g. grayscale to color

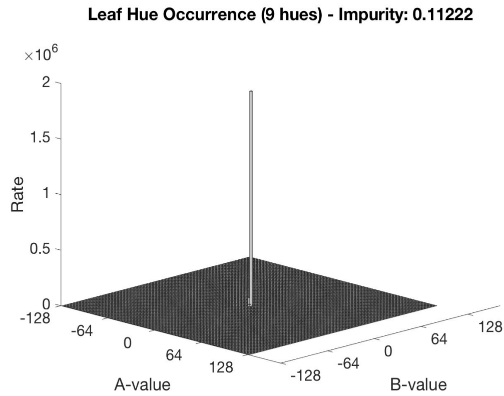


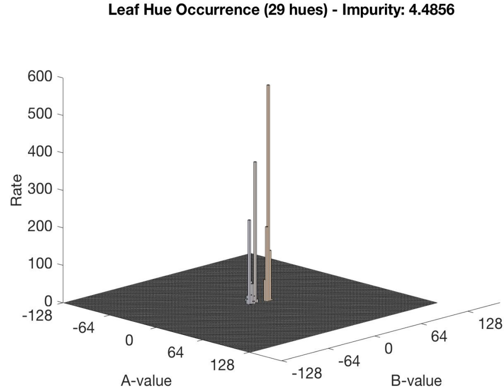
Random Forests - Regression: Colorization

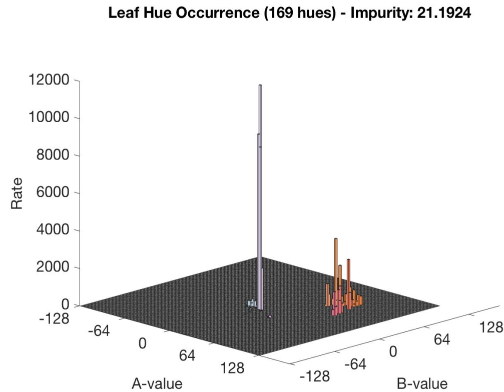
- Data given as intensity image

- Target is *(a, b)* chrominance vector of the *Lab* color space
  - → Leaf information are 2D histograms
  - → Combined by averaging
  - → Final result is the *(a,b)* vector with highest probability
  - → Given intensity will serve as luminance *L*

- Node optimization: Minimize variance
  - → Create child nodes with “pure” colors







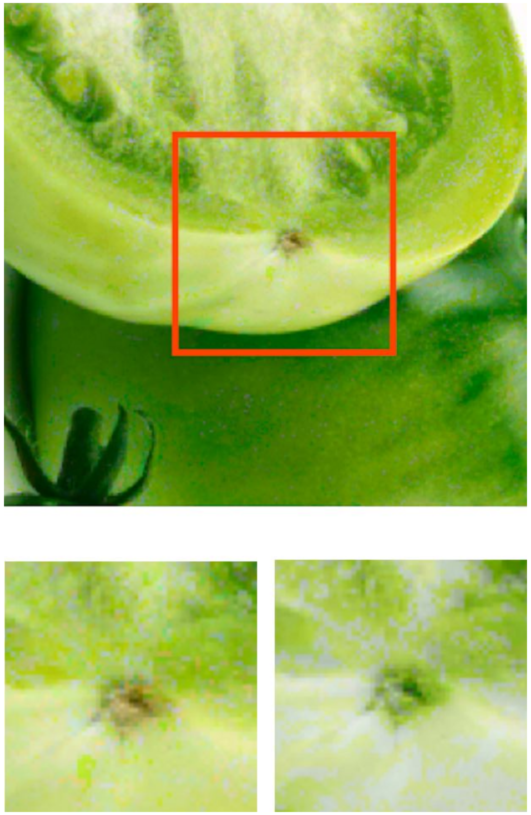
Random Forests - Regression: Colorization

- Data given as intensity image

- Target is *(a, b)* chrominance vector of the *Lab* color space
  - → Leaf information are 2D histograms
  - → Combined by averaging
  - → Final result is the *(a,b)* vector with highest probability
  - → Given intensity will serve as luminance *L*

- Node optimization: Minimize variance
  - → Create child nodes with "pure" colors

- Unbalanced data requires implicit data rebalancing
  - → Use weighted sums (variance, histograms) where the weight is inversely proportional to occurrence.



Random Forests - Regression: Colorization

Reference & Input

Random Forests - Regression: Colorization

Results (RF trained on a few topic-specific images)

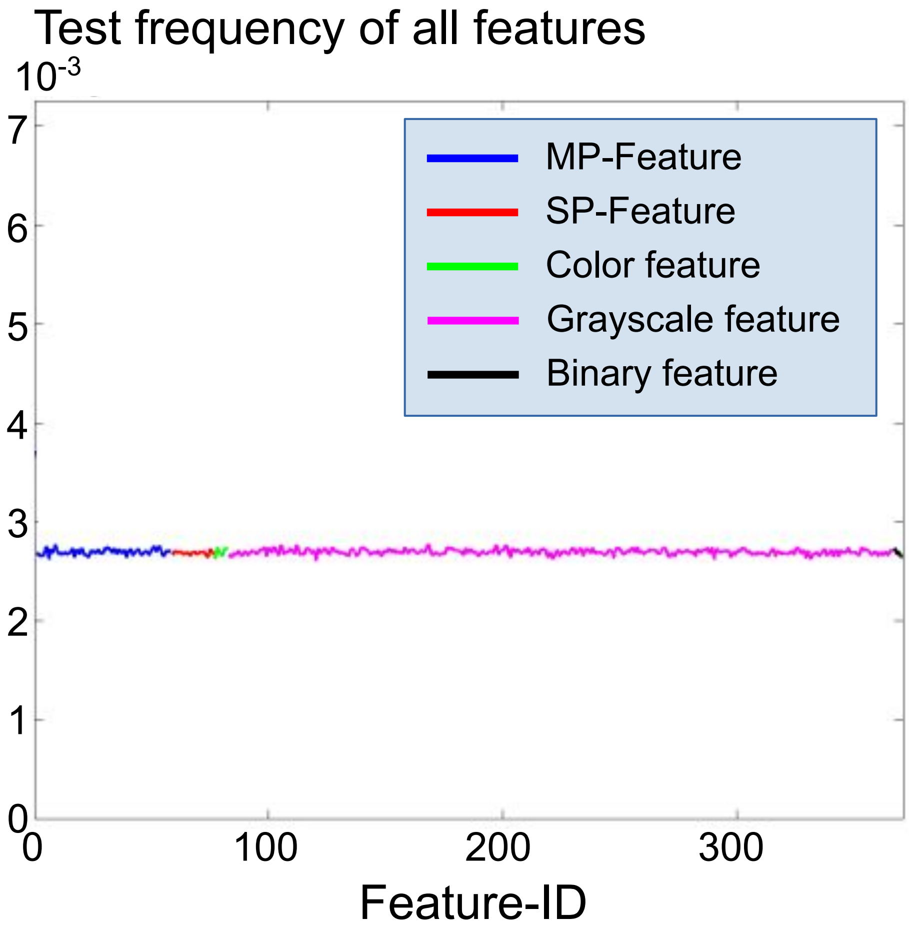
Random Forests - Regression: Colorization

DL results (ConvNet trained on large image database)

Random Forests - Key questions

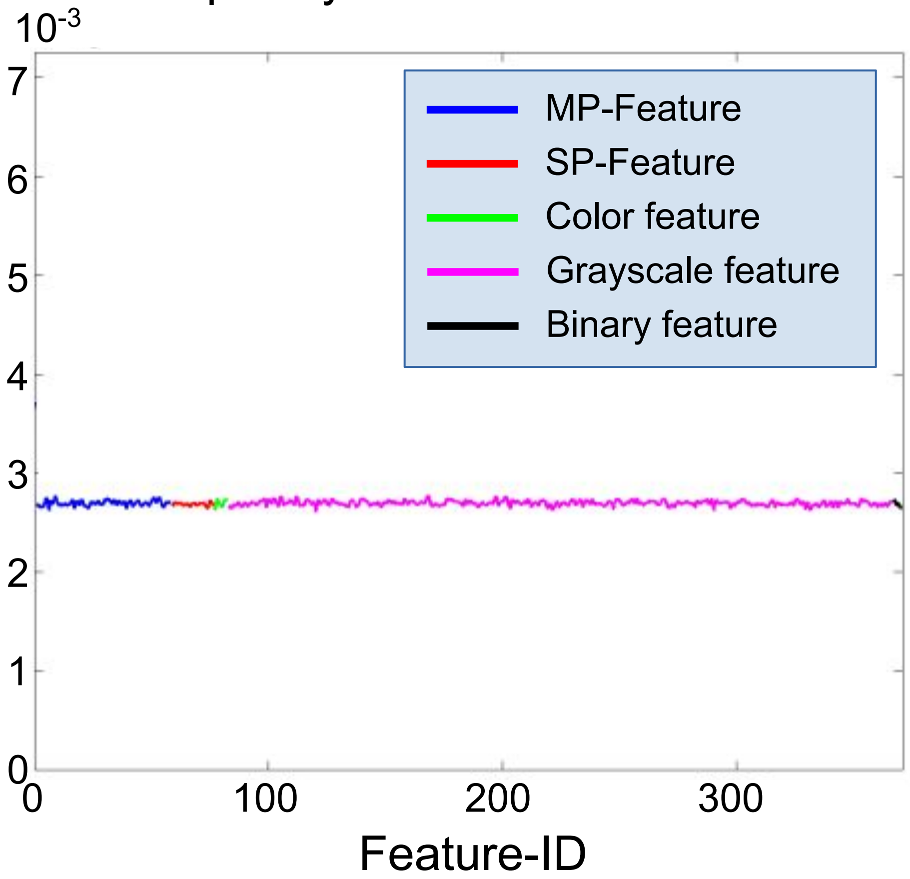
- Why randomization?
  → How to achieve a diverse and strong ensemble?
- What kind of node tests?
  → For images, for other data spaces than R^n
- How to select node tests?
  → How to measure good tests?
- What kind of target variables?
  → More than a single class label?
- How to limit model capacity (tree height, tree number)?
  → The more the better? What about overfitting?
- How to fuse tree decisions?
  → Whom to trust?
- **How to interpret results?**
  → Tree properties and visualization.

Random Forests – Interpretation

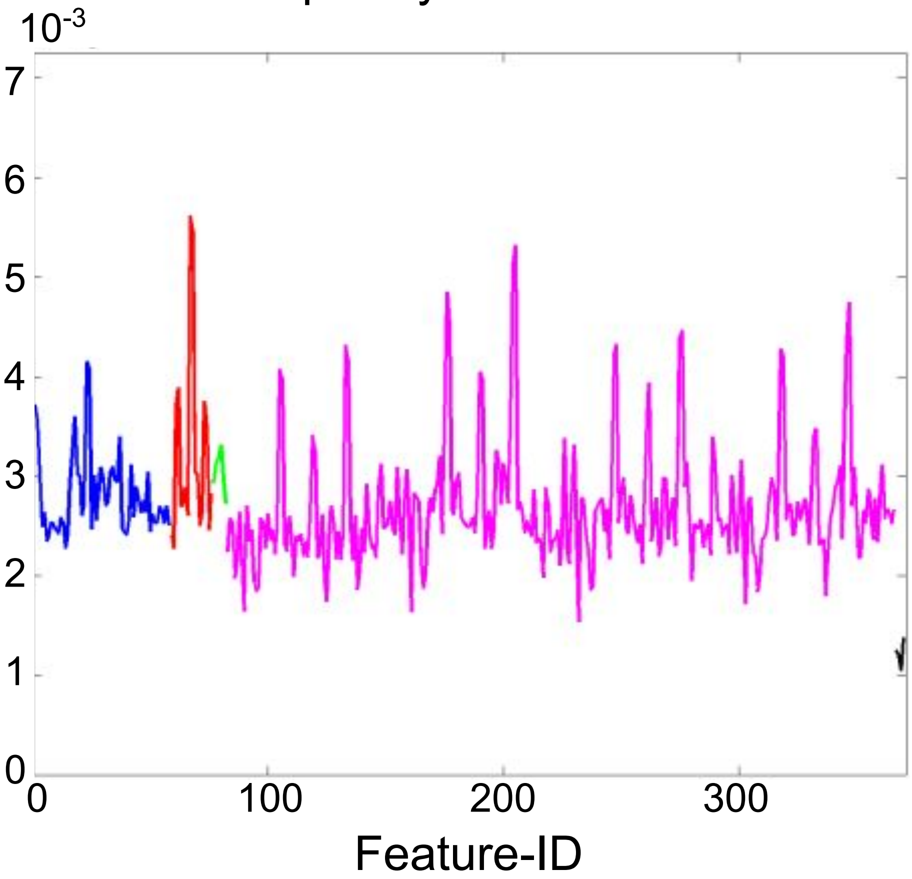


Random Forests – Interpretation

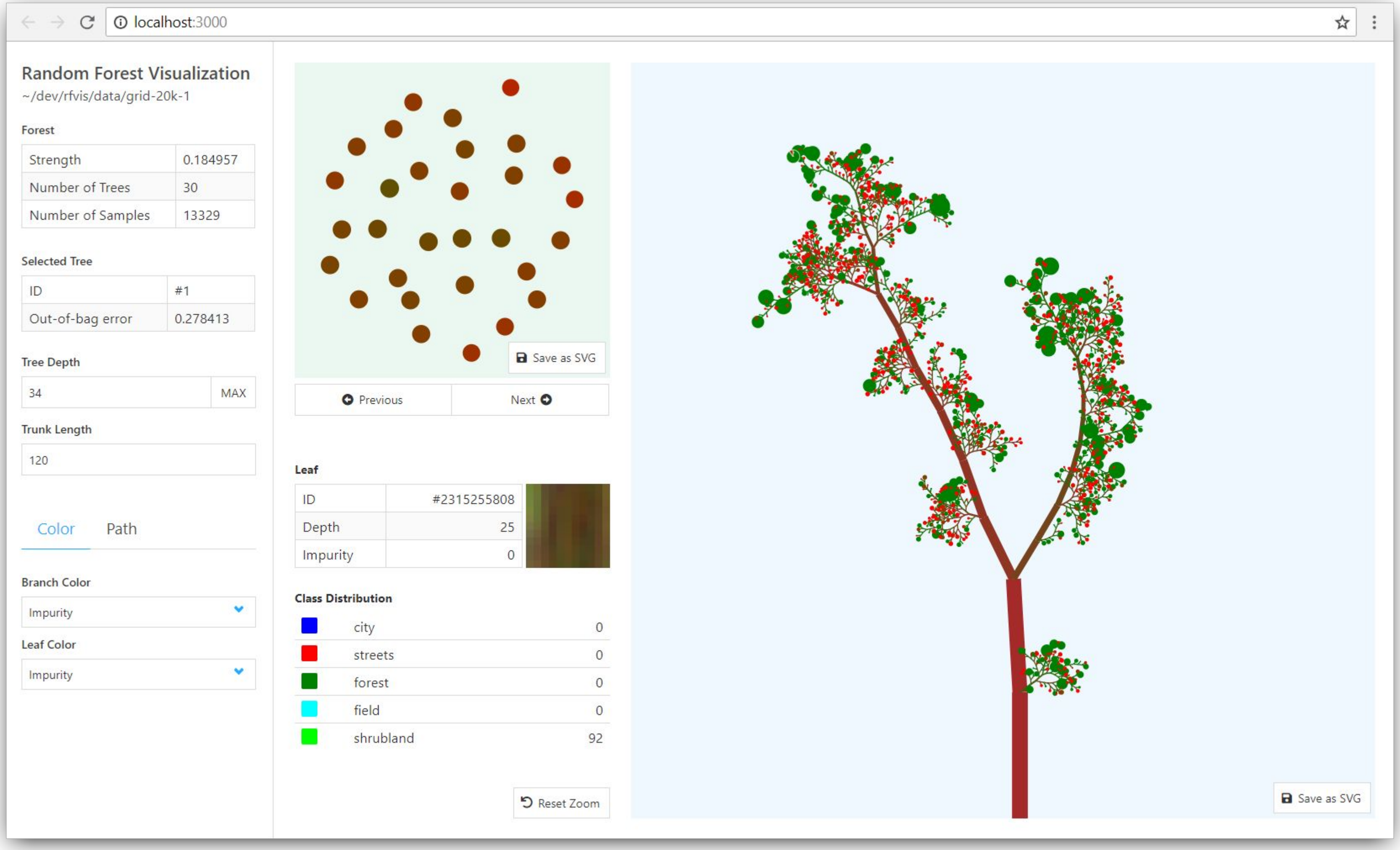
Test frequency of all features



Selection frequency of all features

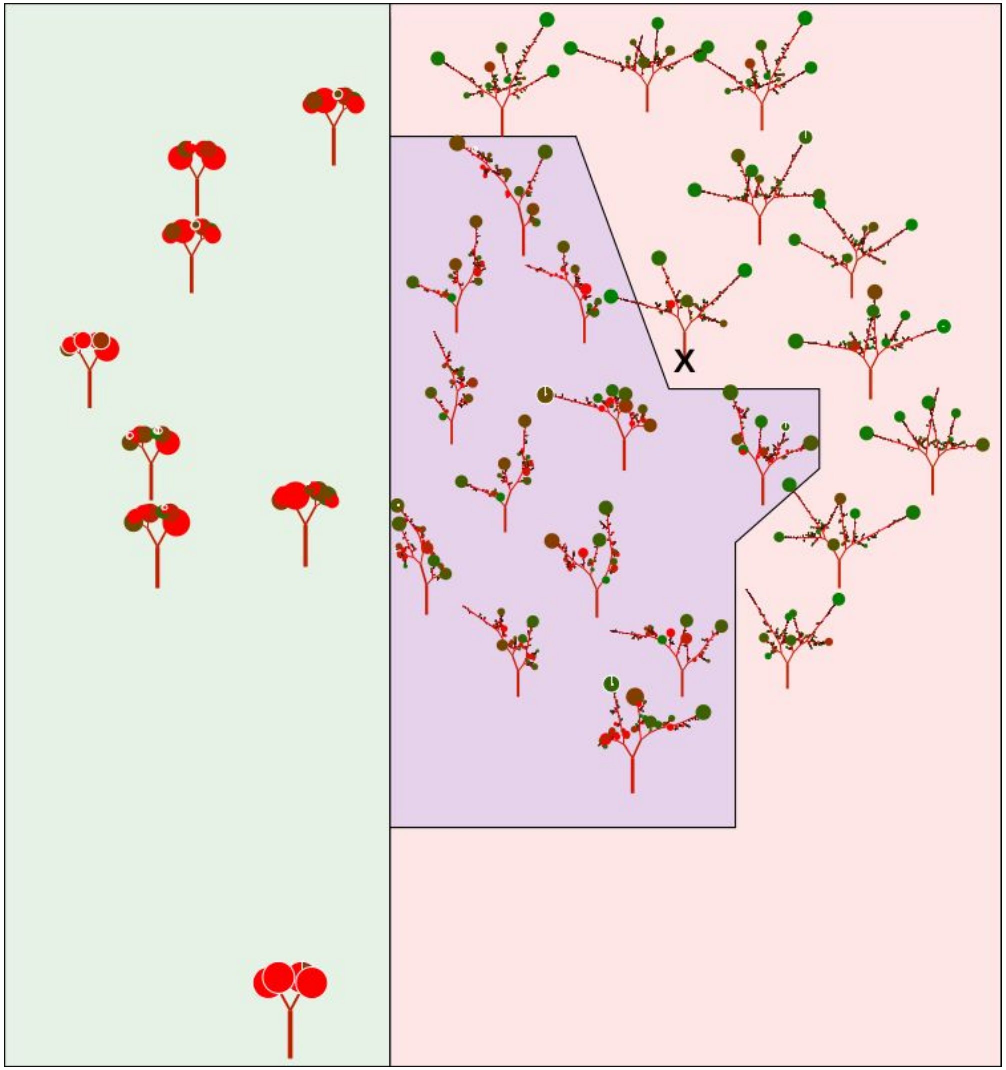


Random Forests – Interpretation: Visualization



Colorful Trees: Visualizing Random Forests for Analysis and Interpretation,
R. Hänsch, P. Wiesner, S. Wendler, O. Hellwich, IEEE Winter Conf. on Applications of Computer Vision, 2019

Random Forests – Interpretation: Forest Overview

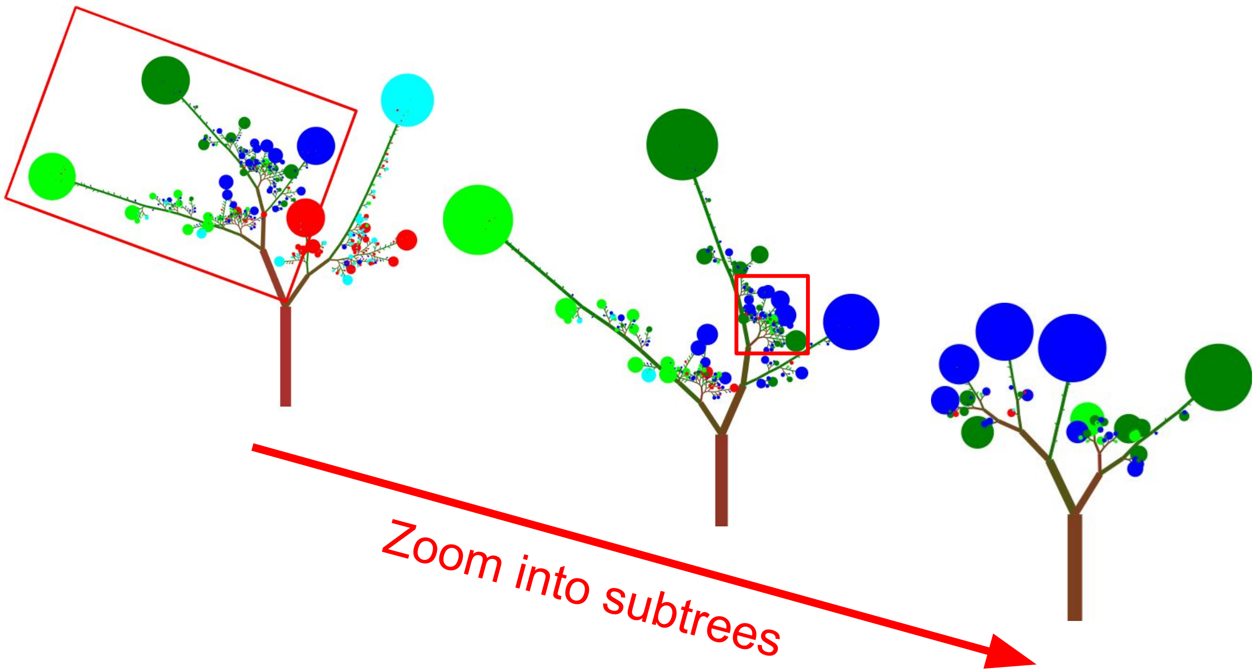


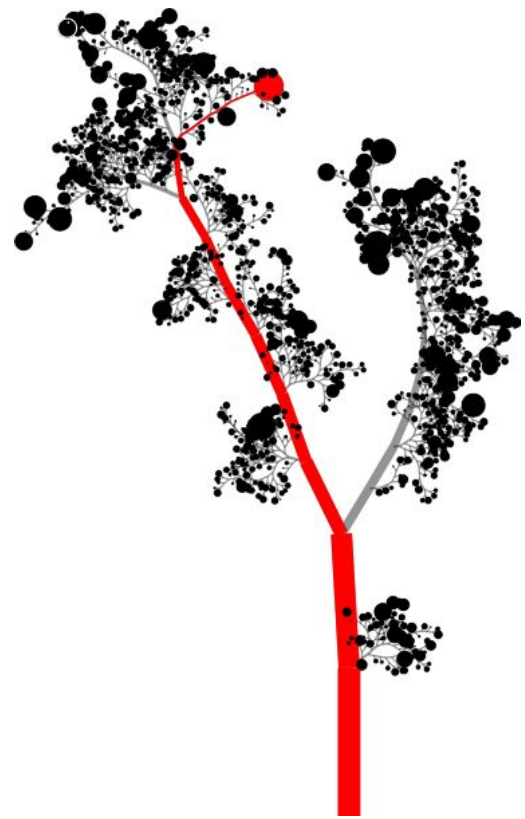
- Arangement of trees in 2D space represents correlation of their decisions

- Trees with similar structure are in spatial proximity (high correlation)

- Allows a fast assessment of individual tree strength as well as tree similarity

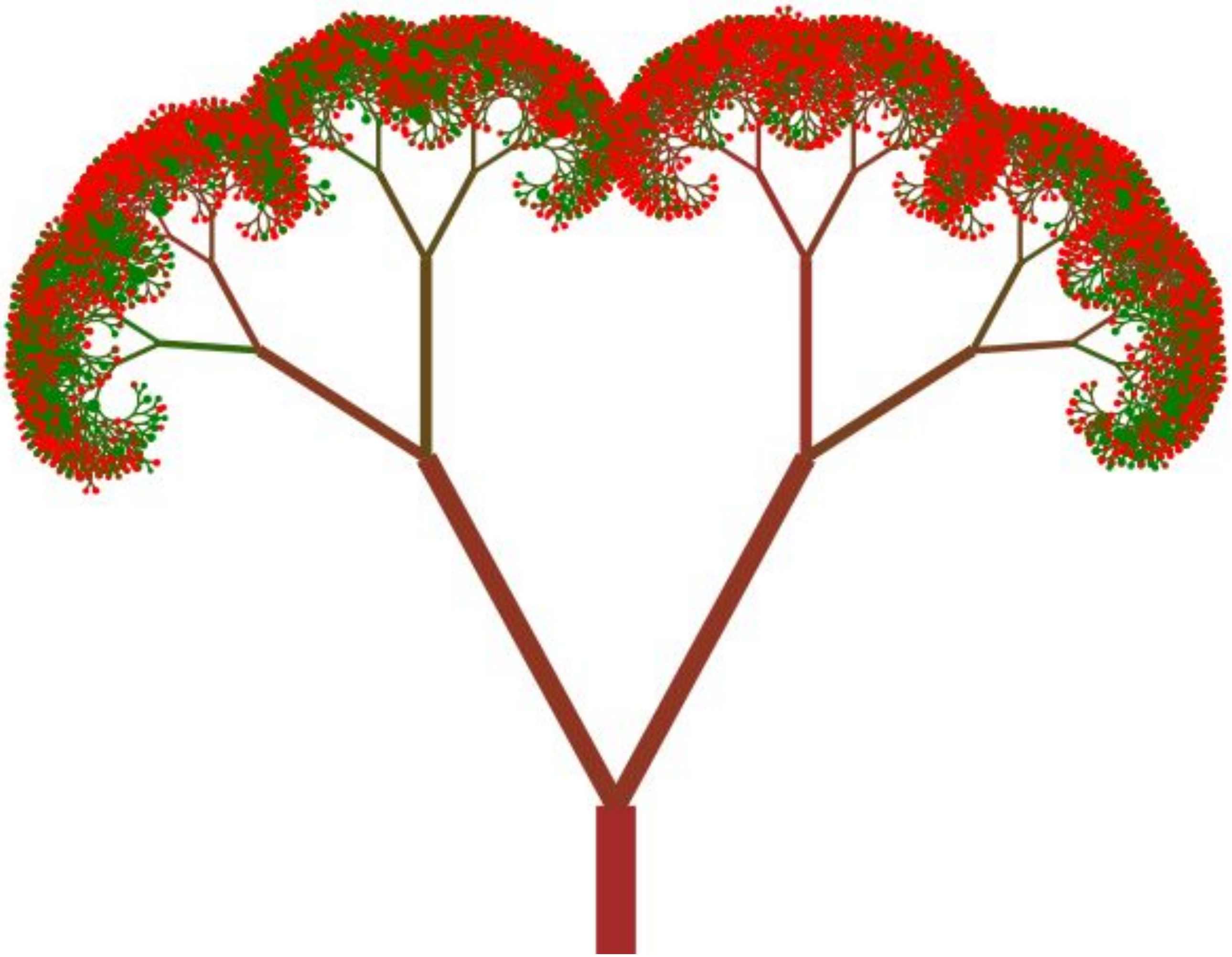
Random Forests – Interpretation: Detailed analysis



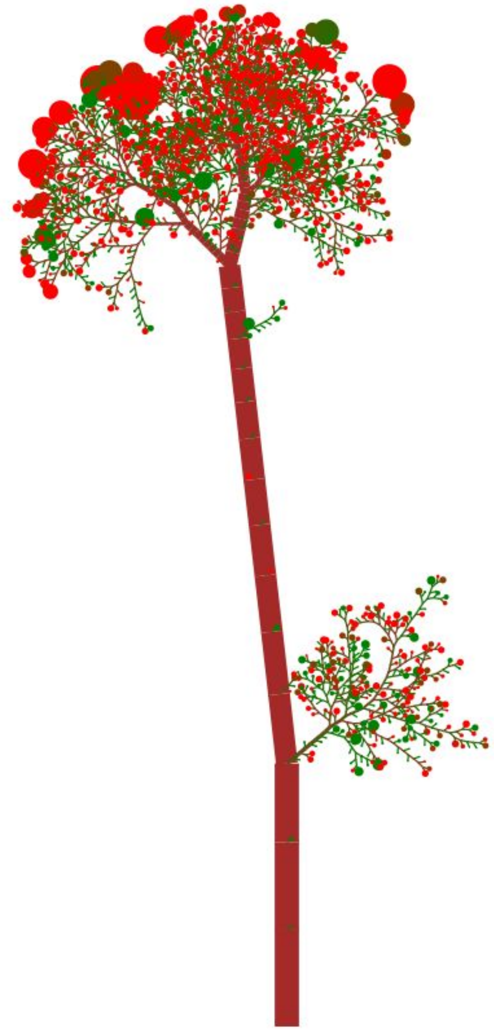


Tracking of the path of indivi-dual samples through the tree

Random Forests – Interpretation: Tree Topology

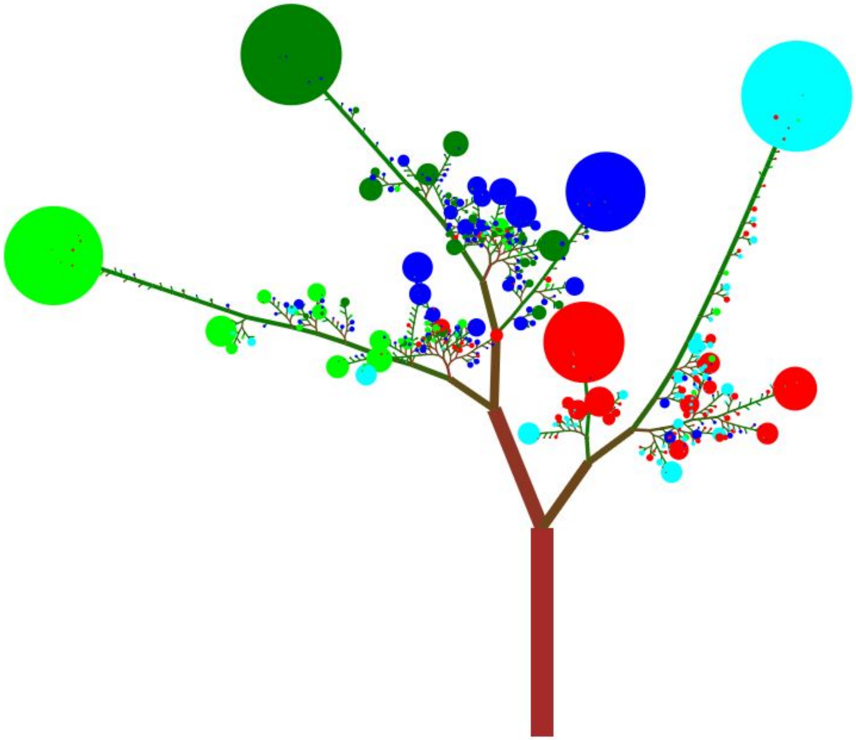


Threshold as median

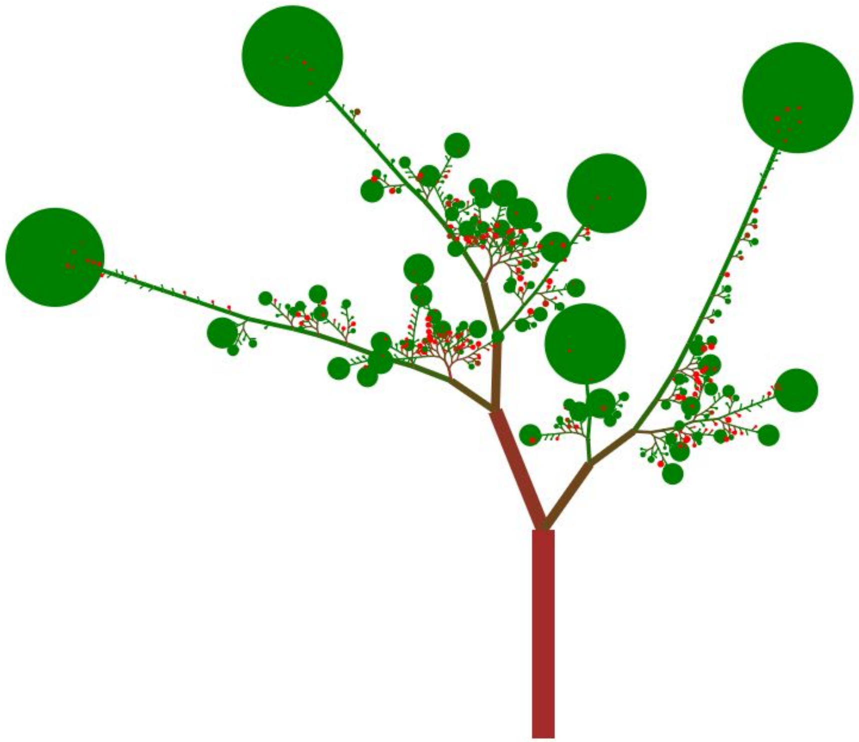


Uniformly sampled threshold

Random Forests – Interpretation: Leaf information



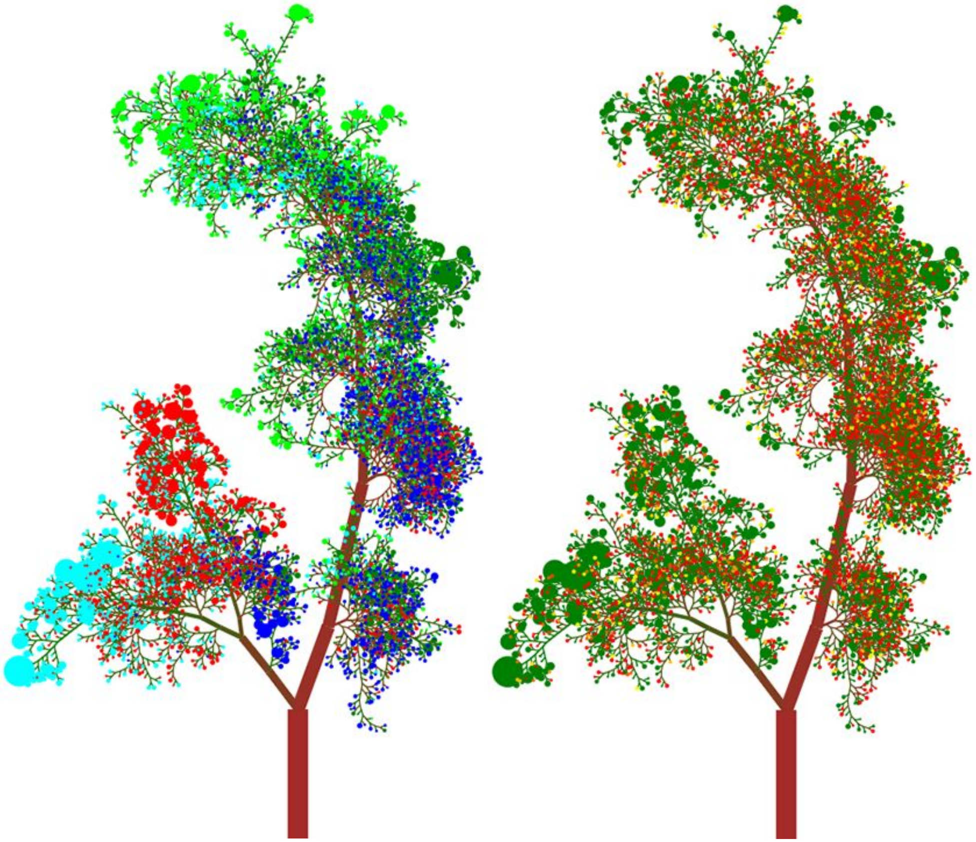
Class assignment



„Purity“, e.g. entropy of the class posterior

Threshold via grid-search (highly optimized)

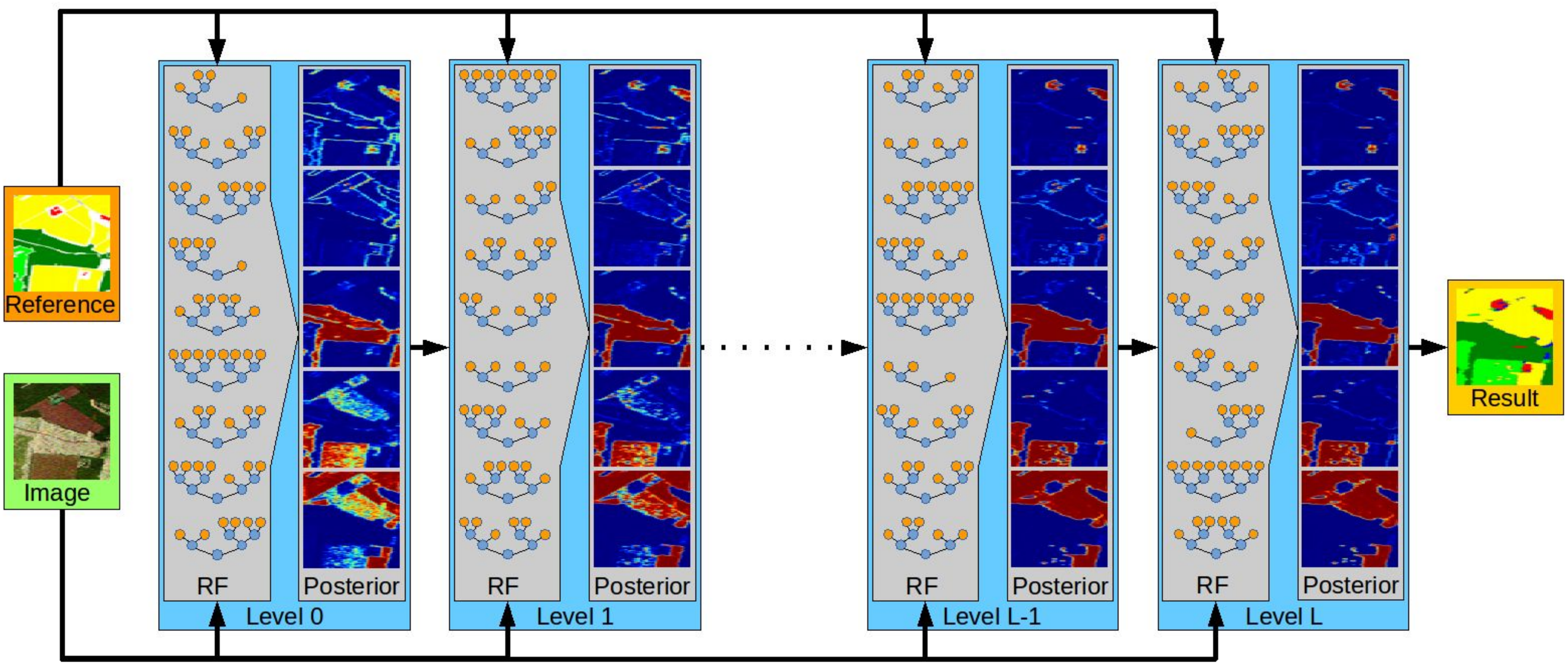
Random Forests – Interpretation: Consolidation nodes



Threshold via grid-search
(weakly optimized)

Height-limited rendering

Random Forests – Advanced Concepts: Stacked RF



Classification of PolSAR Images by Stacked Random Forests,
R. Hänsch, O. Hellwich, ISPRS International Journal of Geo-Information, 2018

Random Forests – Advanced Concepts: Stacked RF

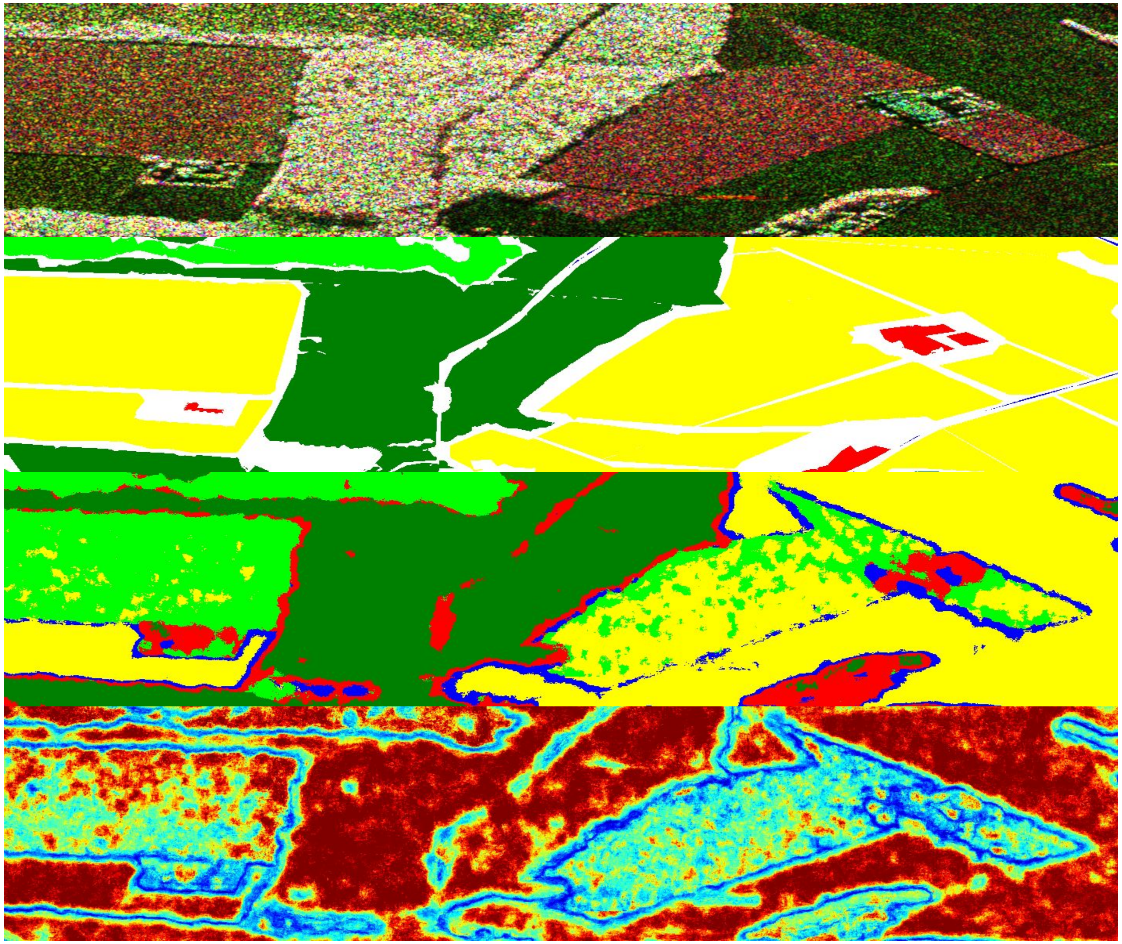


Image detail

Reference

Estimate

Uncertainty

Level 1

BA = 86.8%

Random Forests – Advanced Concepts: Stacked RF

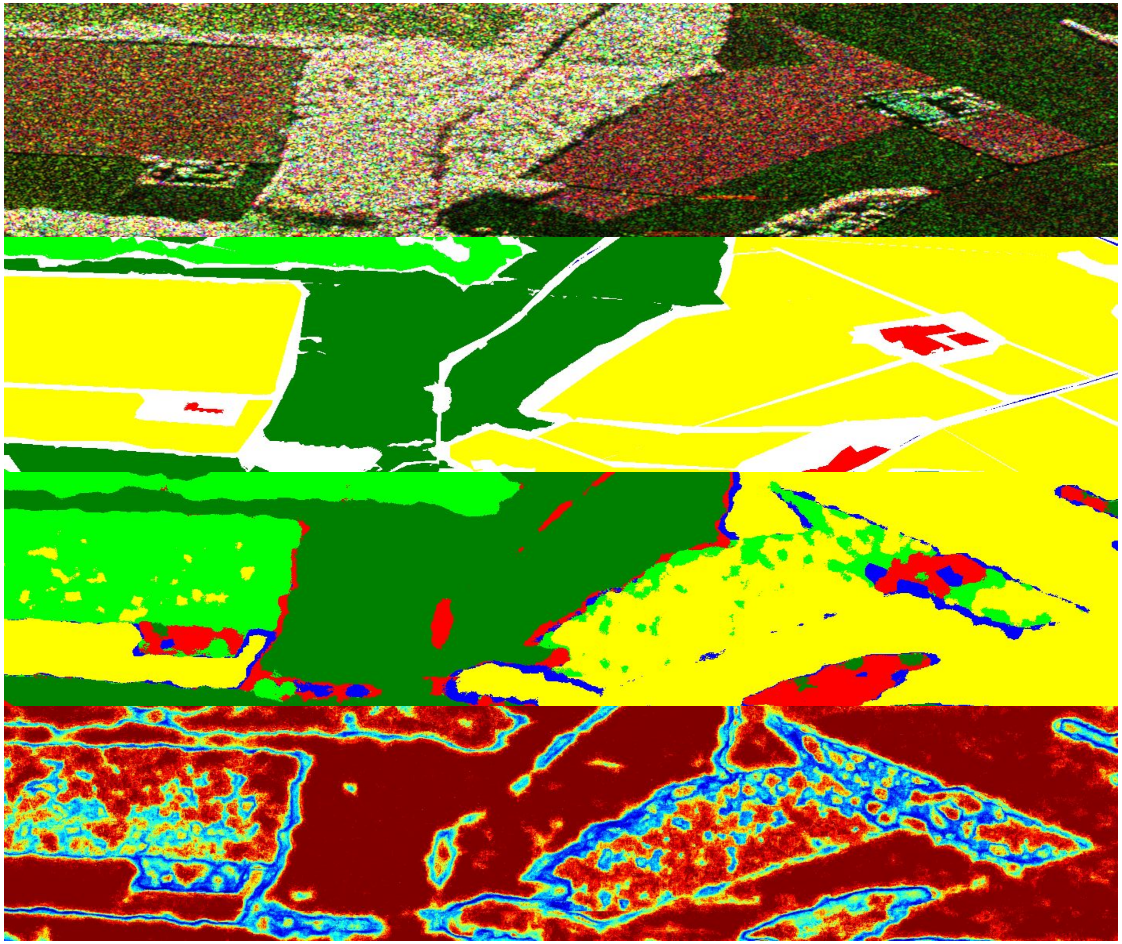


Image detail

Reference

Estimate

Uncertainty

Level 2

BA = 88.6%

Random Forests – Advanced Concepts: Stacked RF

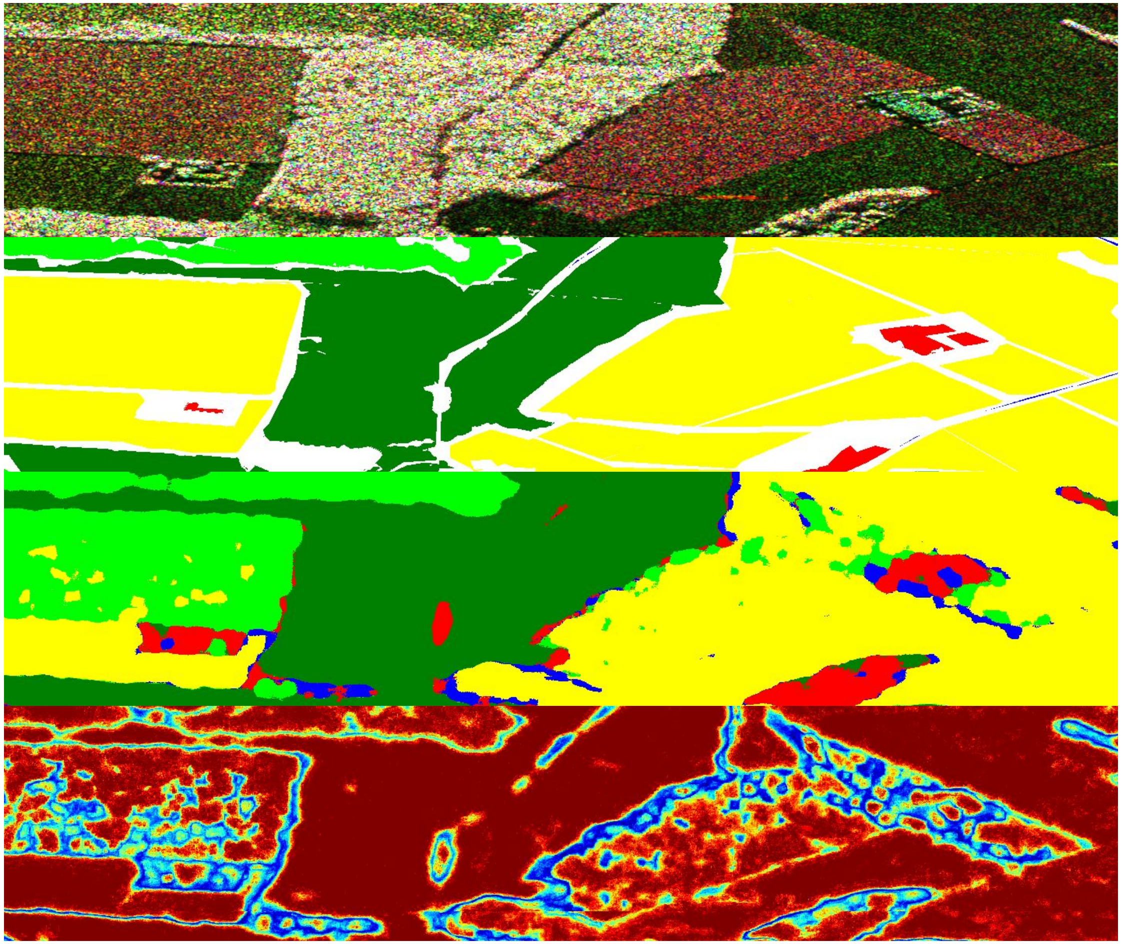


Image detail

Reference

Estimate

Uncertainty

Level 3

BA = 89.5%

Random Forests – Advanced Concepts: Stacked RF

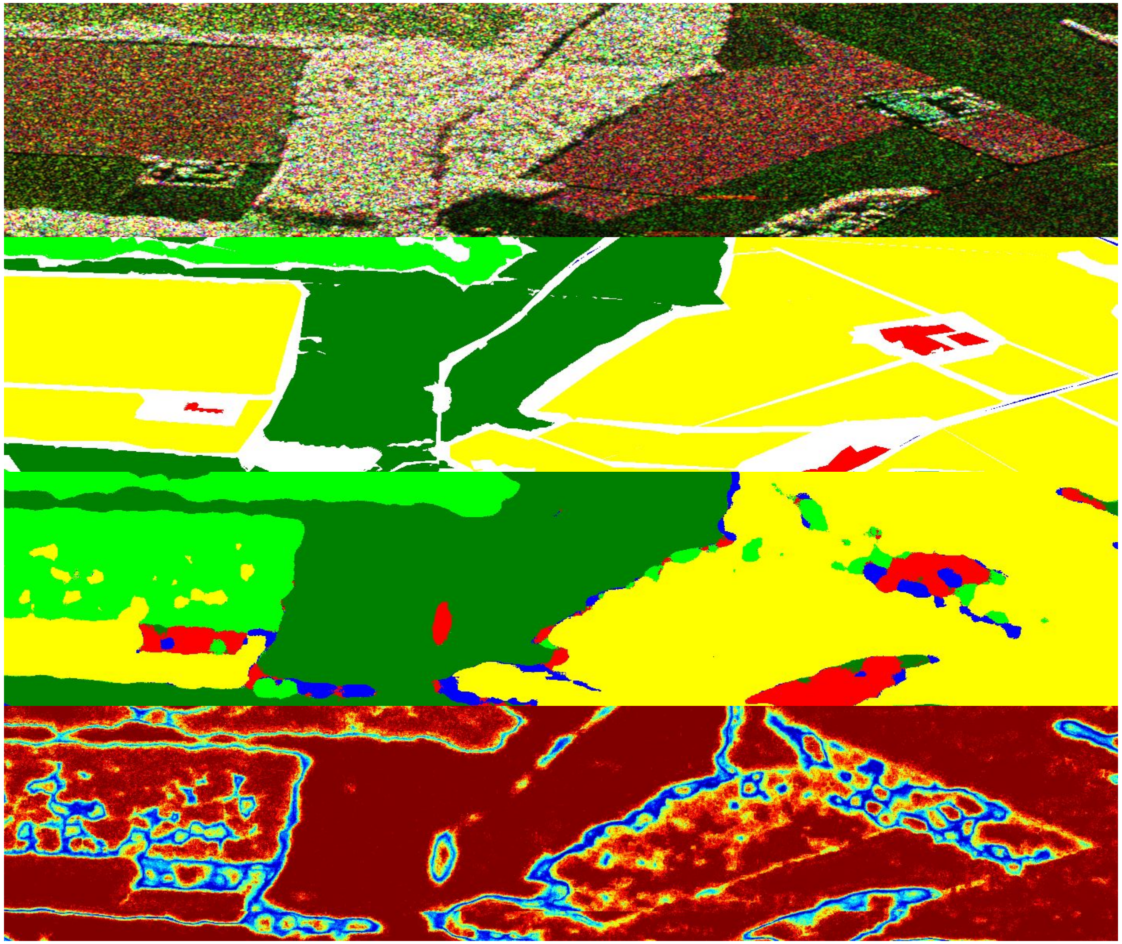


Image detail

Reference

Estimate

Uncertainty

Level 4

BA = 90.0%

Random Forests – Advanced Concepts: Stacked RF

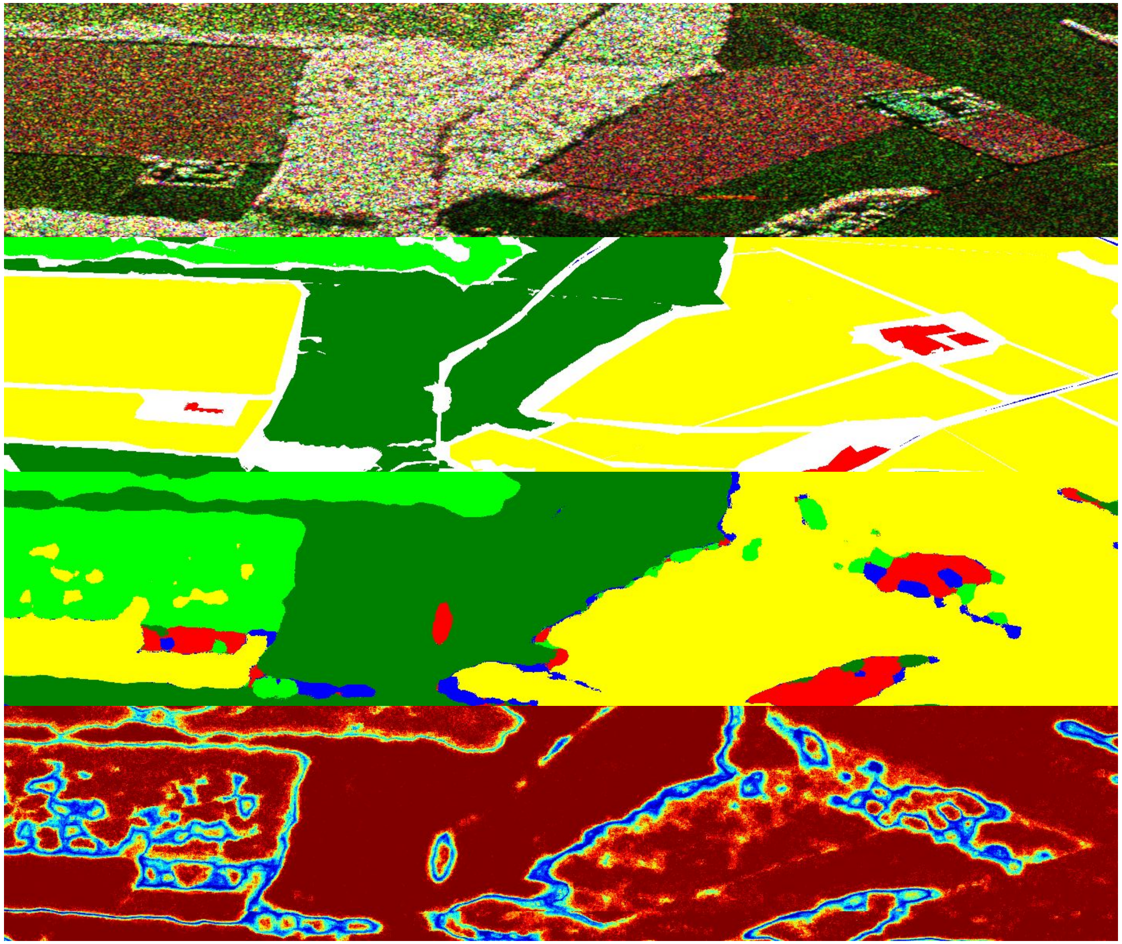


Image detail

Reference

Estimate

Uncertainty

Level 5

BA = 90.4%

Random Forests – Advanced Concepts: Stacked RF

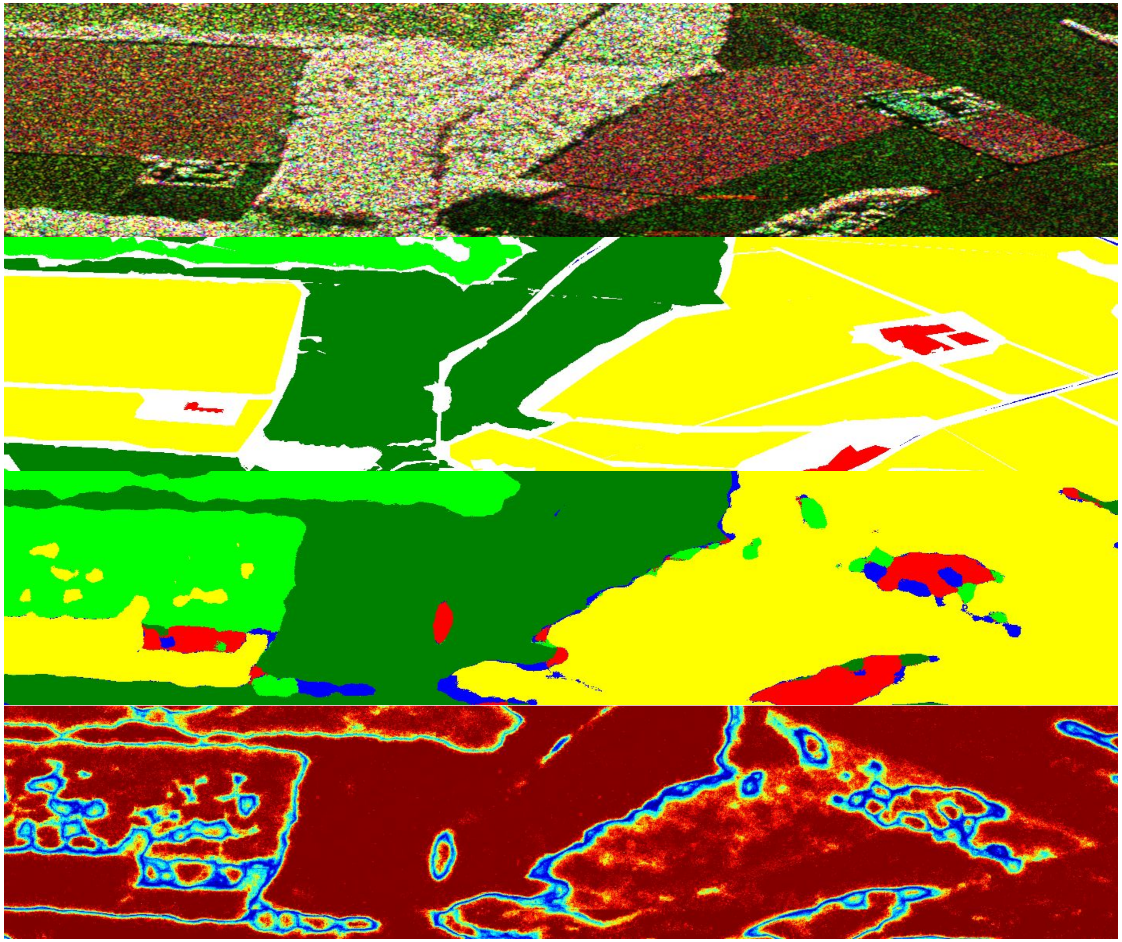


Image detail

Reference

Estimate

Uncertainty

Level 6

BA = 90.5%

Random Forests – Advanced Concepts: Stacked RF

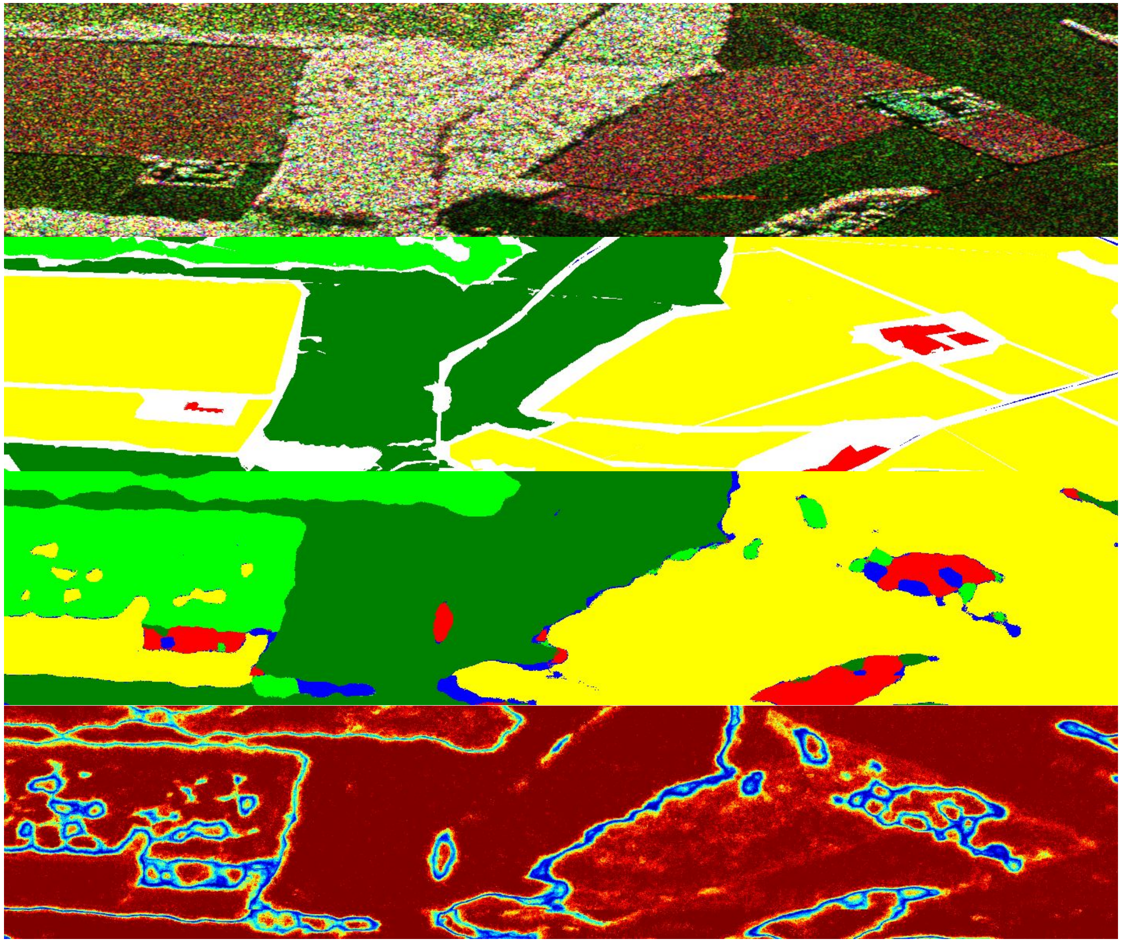


Image detail

Reference

Estimate

Uncertainty

Level 7

BA = 90.4%

Random Forests – Advanced Concepts: Stacked RF

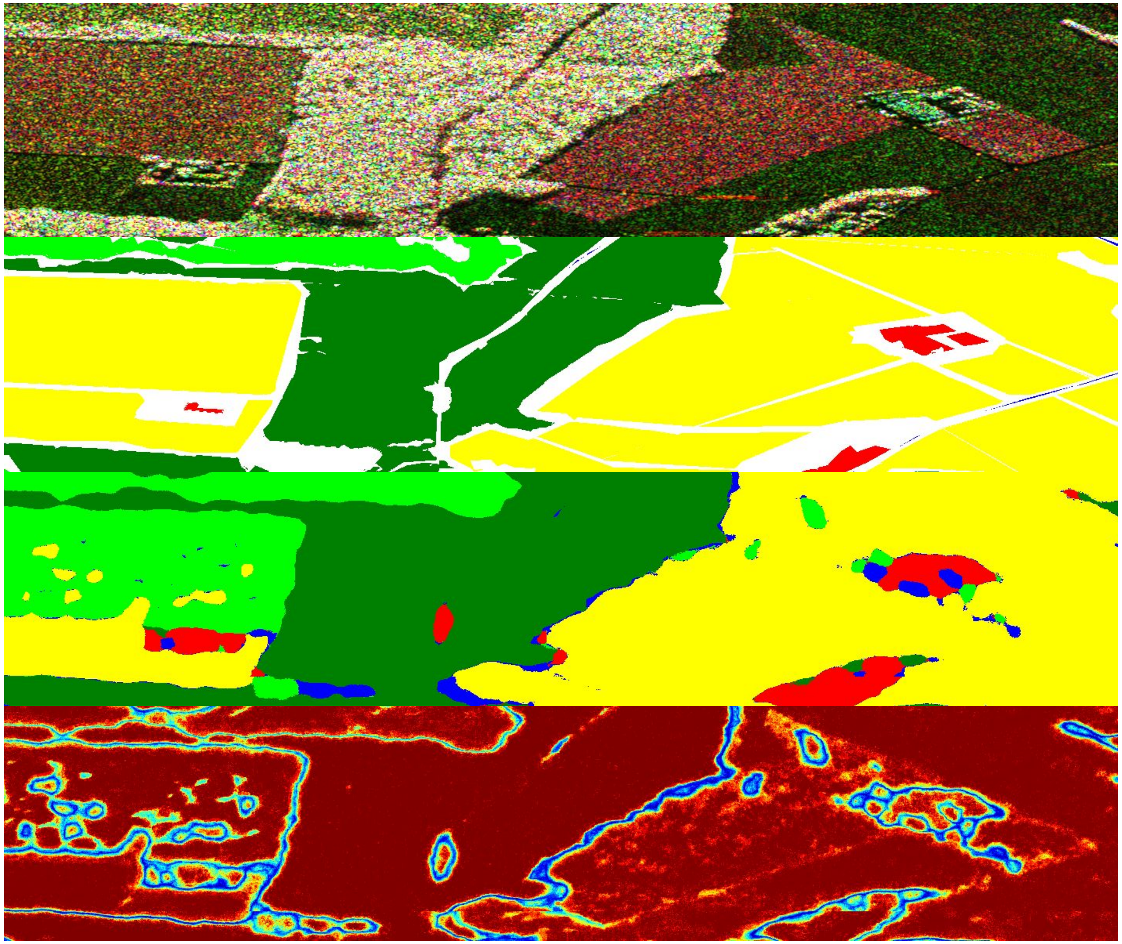


Image detail

Reference

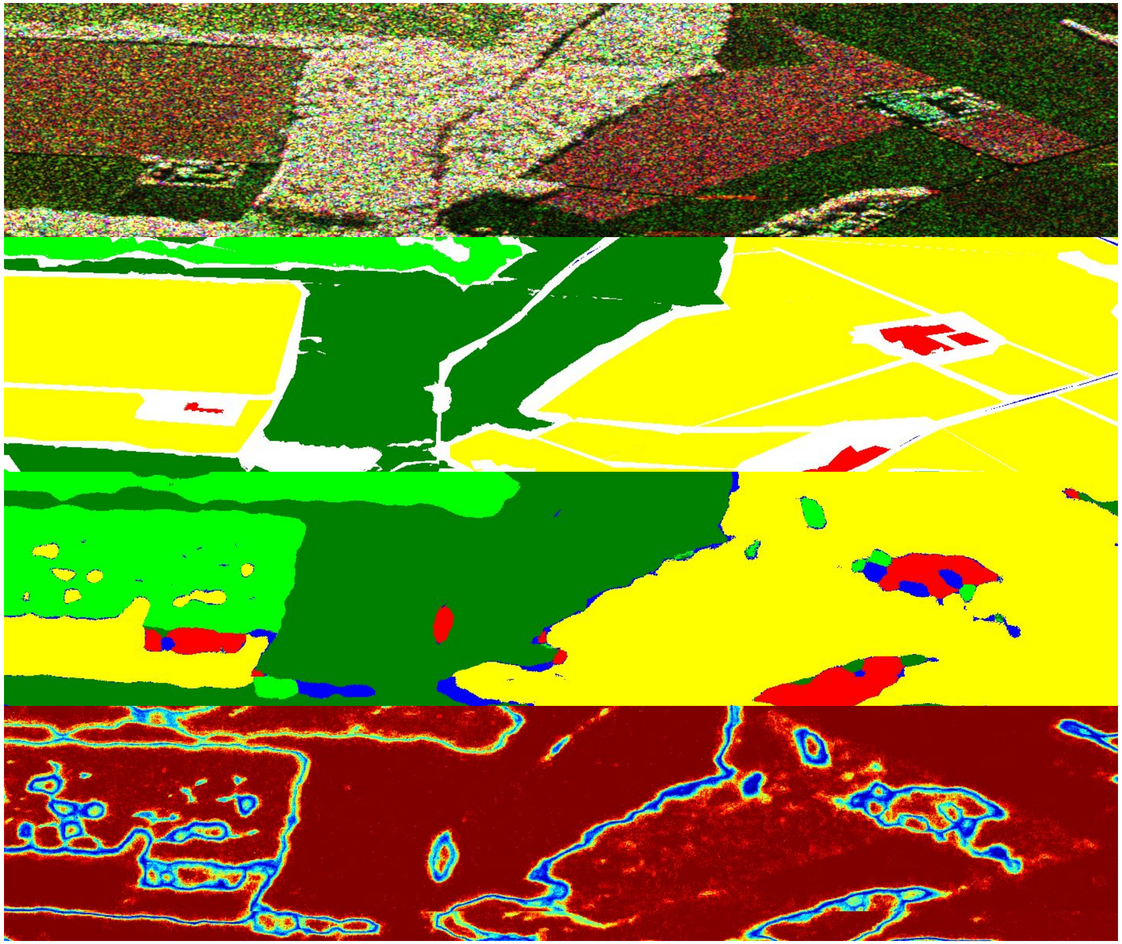
Estimate

Uncertainty

Level 8

BA = 90.5%

Random Forests – Advanced Concepts: Stacked RF



Level 9

BA = 90.6%

Random Forests – Advanced Concepts: Stacked RF

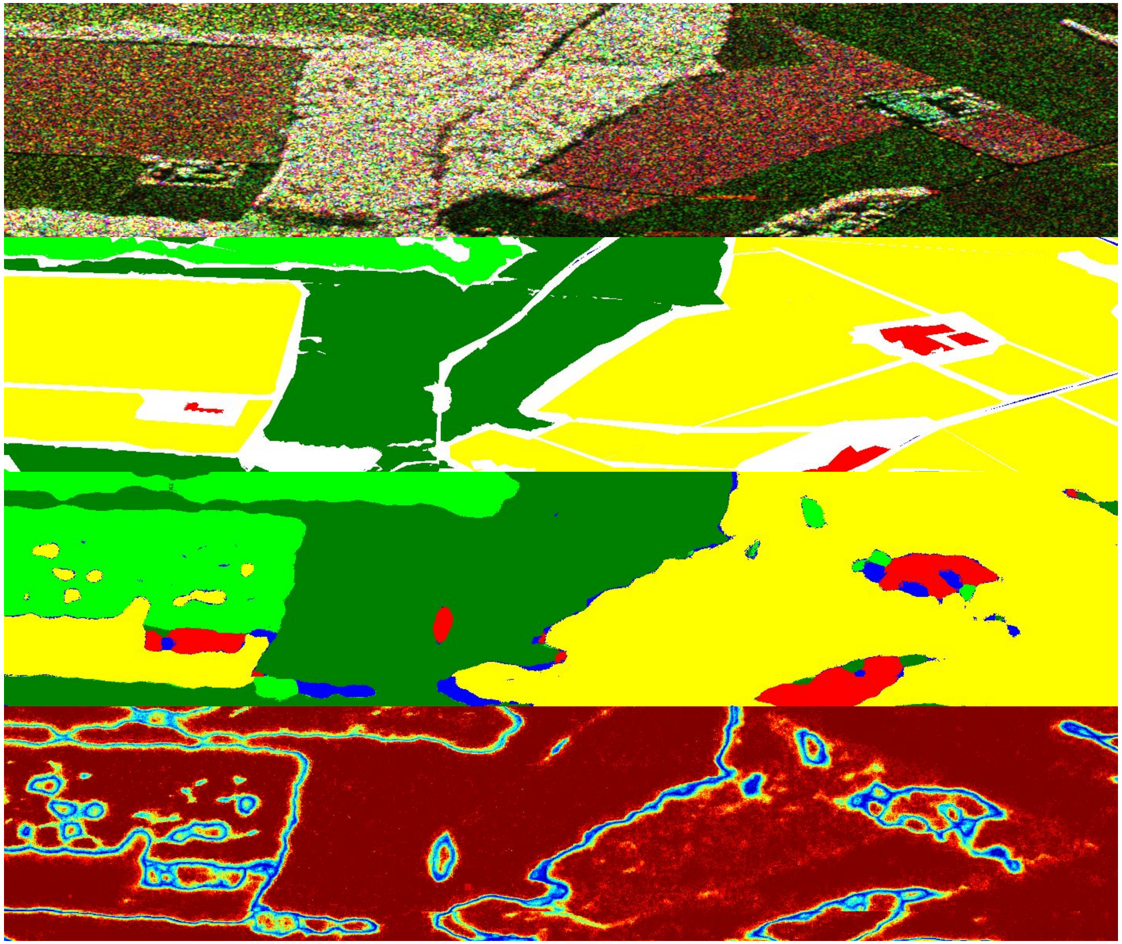


Image detail

Reference

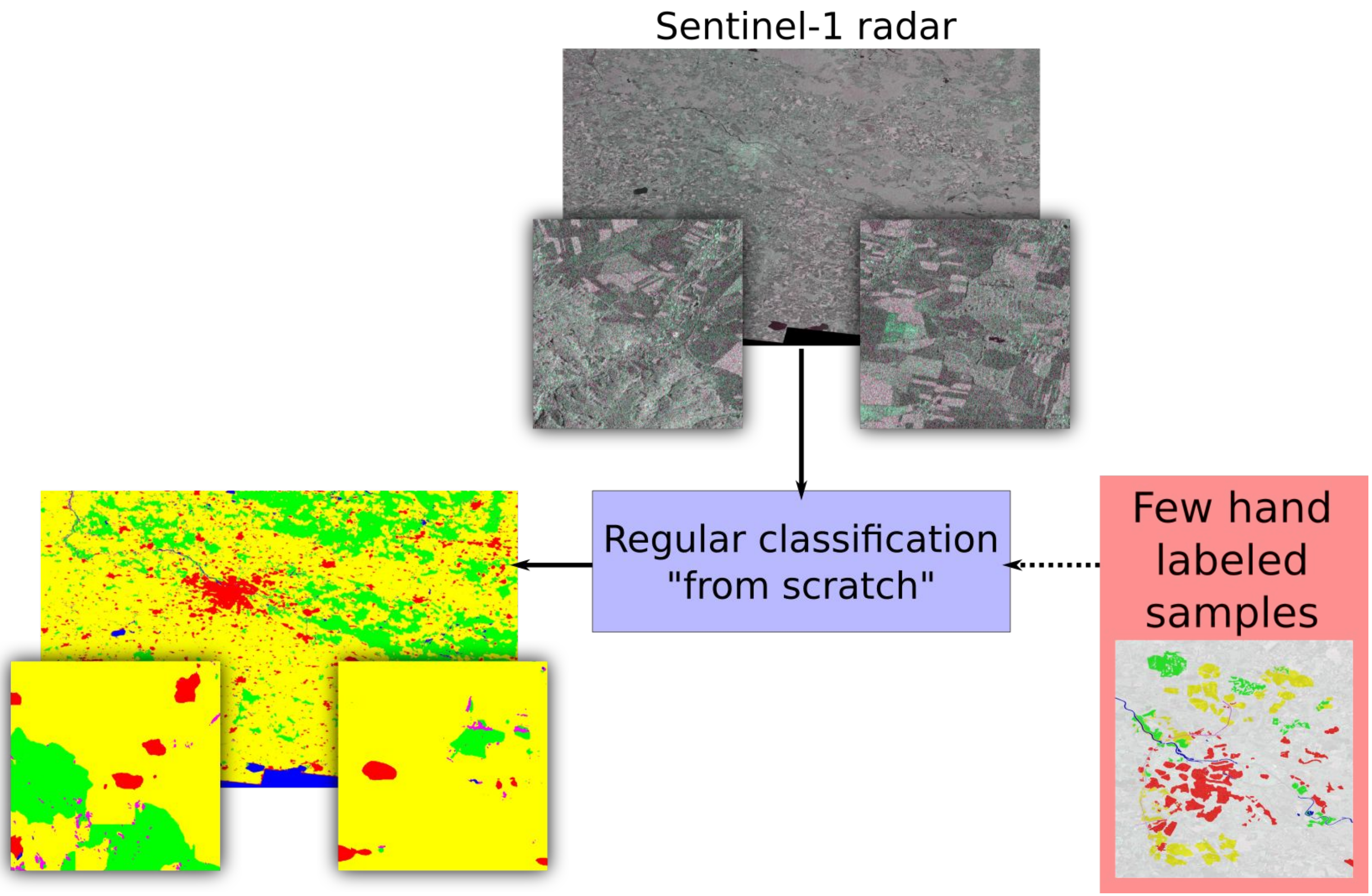
Estimate

Uncertainty

Level 10

BA = 90.7%

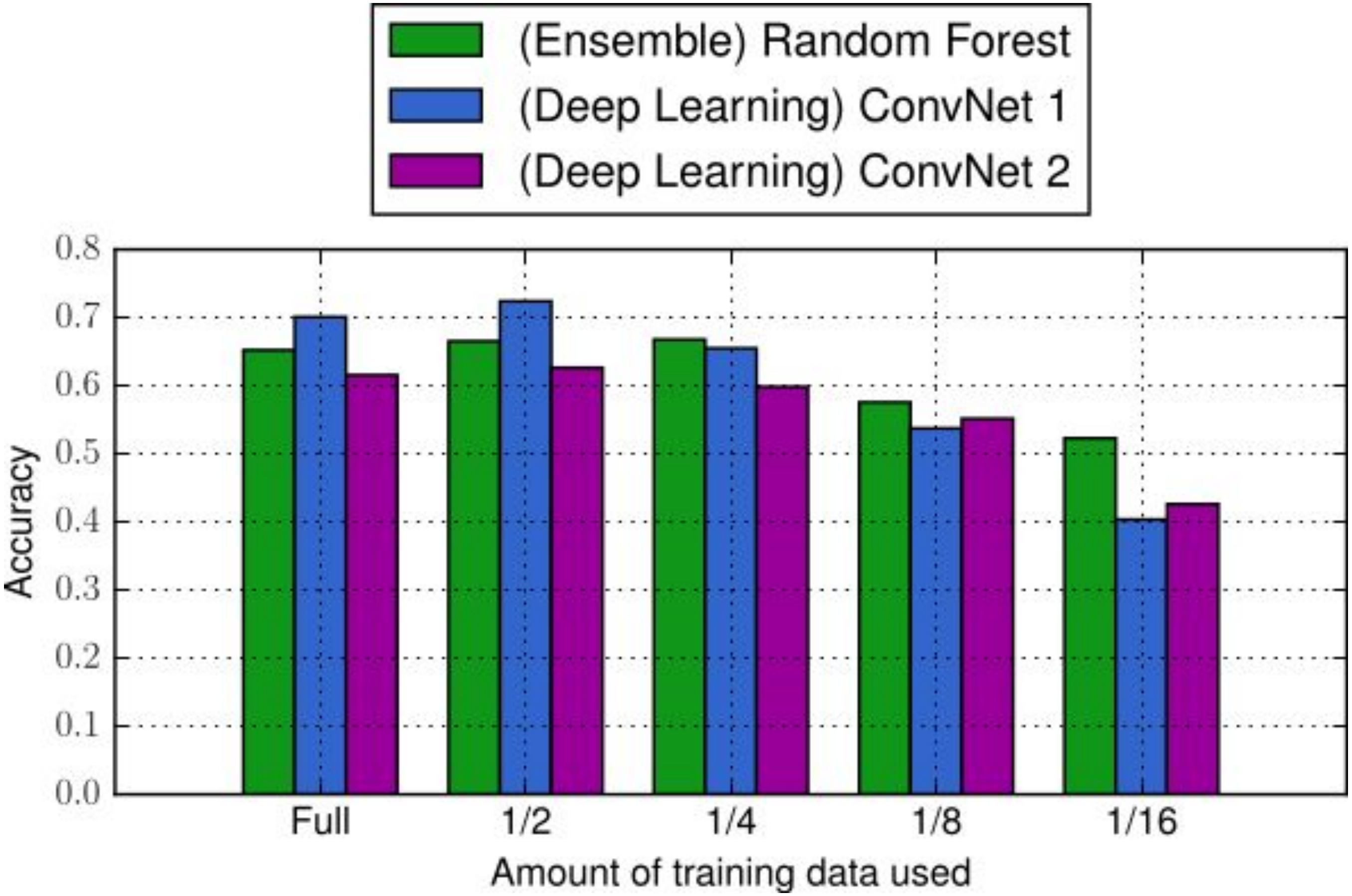
But what about Deep Learning?



Exploiting GAN-Based SAR to Optical Image Transcoding for Improved Classification via Deep Learning
A. Ley, O. D'Hondt, S. Valade, R. Hänsch, O. Hellwich, EUSAR 2018

DLR

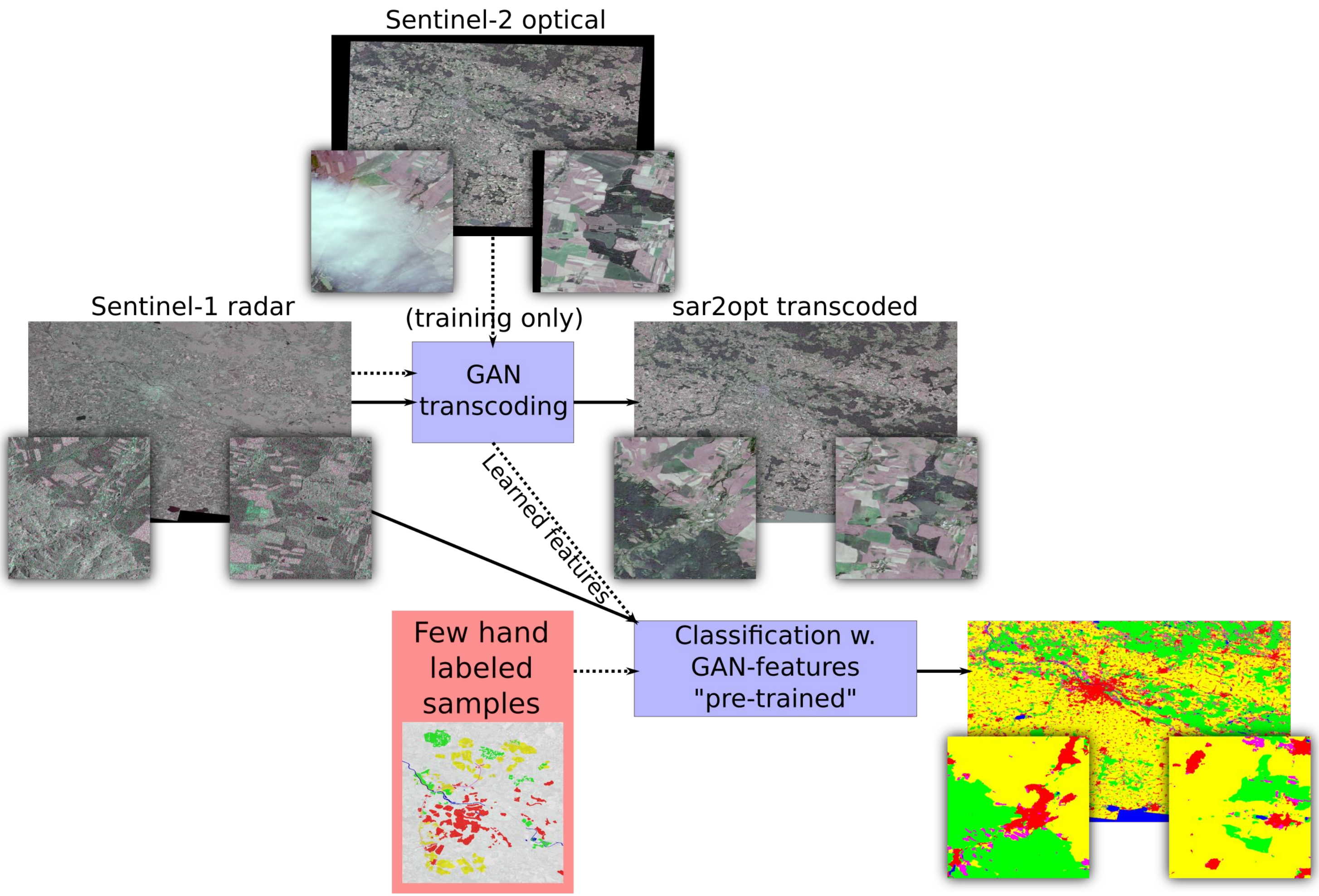
But what about Deep Learning?



Exploiting GAN-Based SAR to Optical Image Transcoding for Improved Classification via Deep Learning
A. Ley, O. D'Hondt, S. Valade, R. Hänsch, O. Hellwich, EUSAR 2018

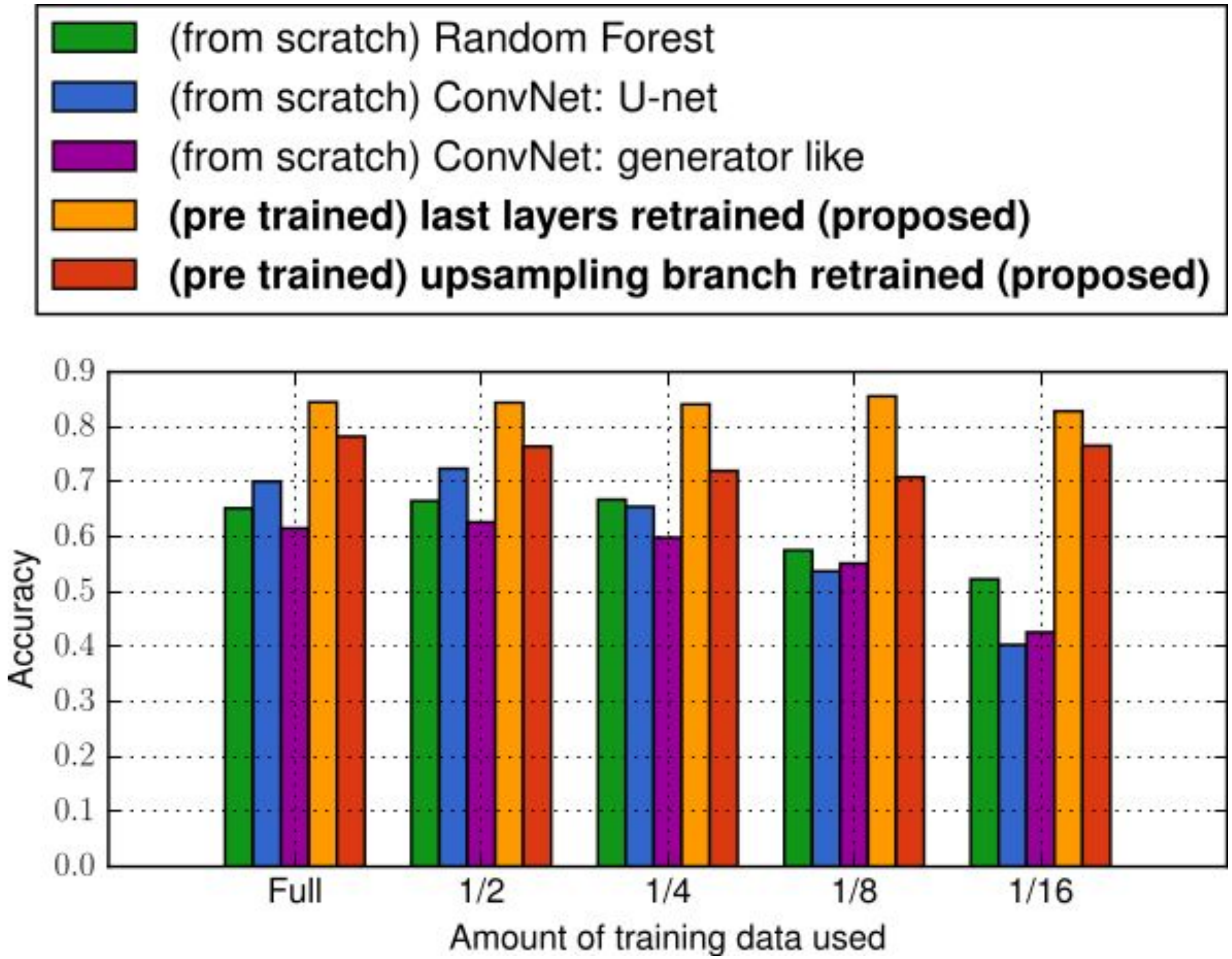
DLR

Self-supervised learning via transcoding



Exploiting GAN-Based SAR to Optical Image Transcoding for Improved Classification via Deep Learning
A. Ley, O. D'Hondt, S. Valade, R. Hänsch, O. Hellwich, EUSAR 2018

Self-supervised learning via transcoding



Exploiting GAN-Based SAR to Optical Image Transcoding for Improved Classification via Deep Learning
A. Ley, O. D'Hondt, S. Valade, R. Hänsch, O. Hellwich, EUSAR 2018

Conclusion

- Deep Learning works! Differentiable learning won’t go away for the next years.

- But (modern!) shallow learners are still of importance.

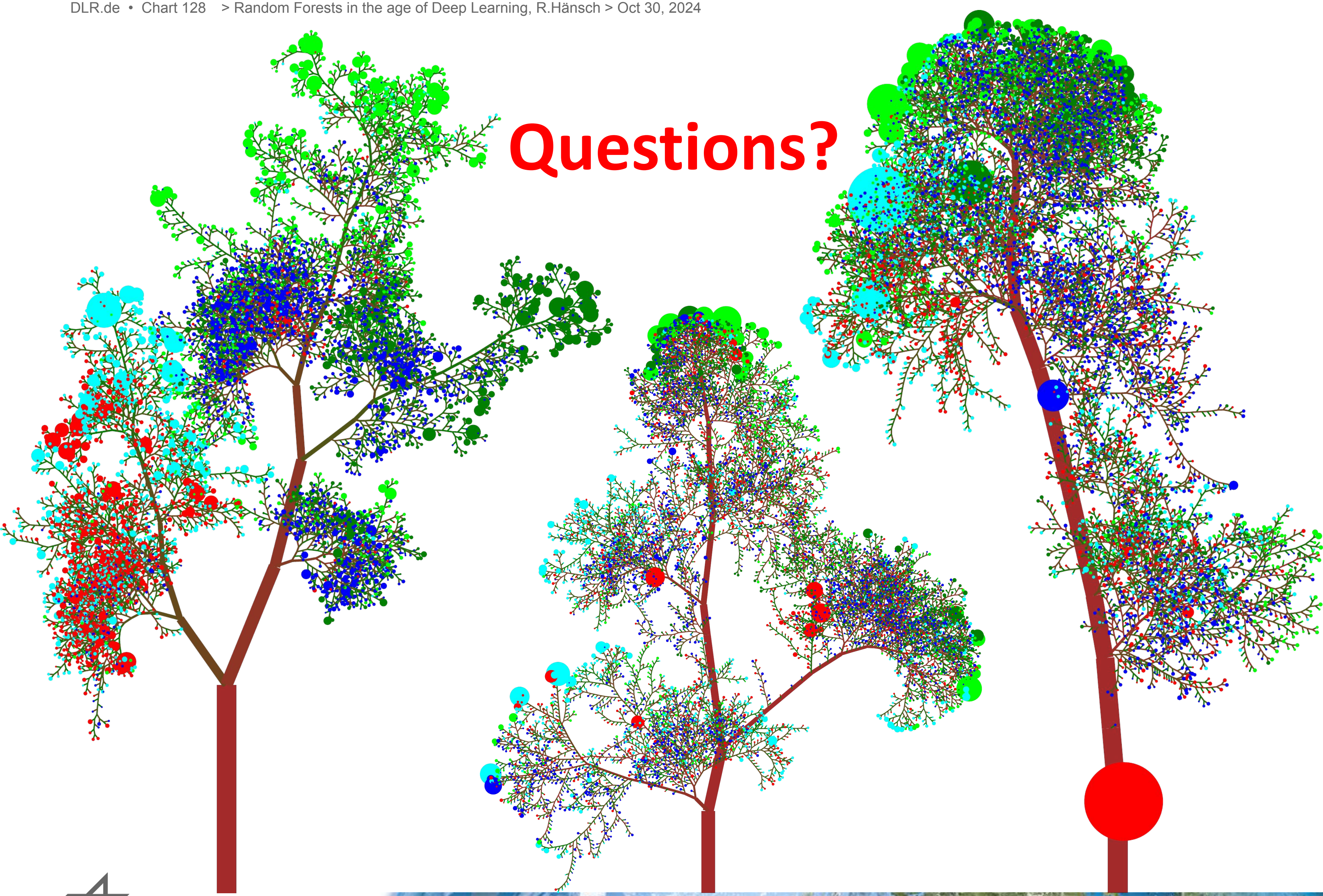
- They are competitive and sometimes even superior to deep learners.

- RF (and other shallow learners) scale less well with large datasets

- Decision trees are not differentiable (at least not in their vanilla version)

- Take home message: Use the right tool for the right job (in the right way).

Questions?



DLR