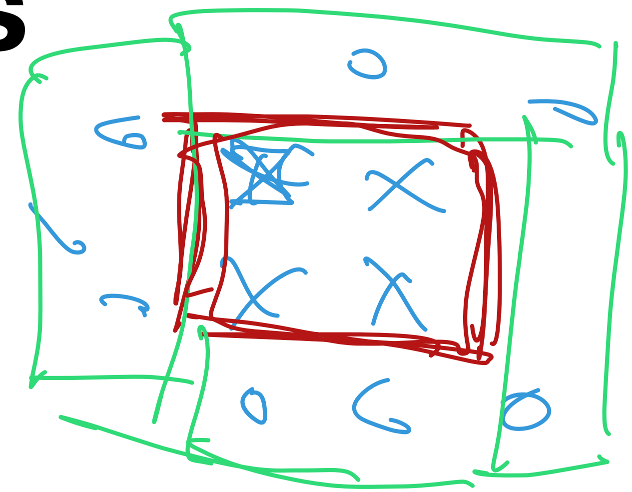
Decision Trees and Boosting

Lecture 22 - CS 689, Spring 2023

- This week
  - Decision Trees
  - Ensemble (additive) decision trees
    - Today: optimization with boosting
    - Thursday: Bayesian learning with MCMC
- Discrete parameters; no latent representations

Axis-parallel regions



- Feature thresholds

$$1\{x_j < t\}$$

$$T_j = \text{unique values of } x_{ij} \;\; \forall x_i \in D^{Tr}$$

- Conjunctive form

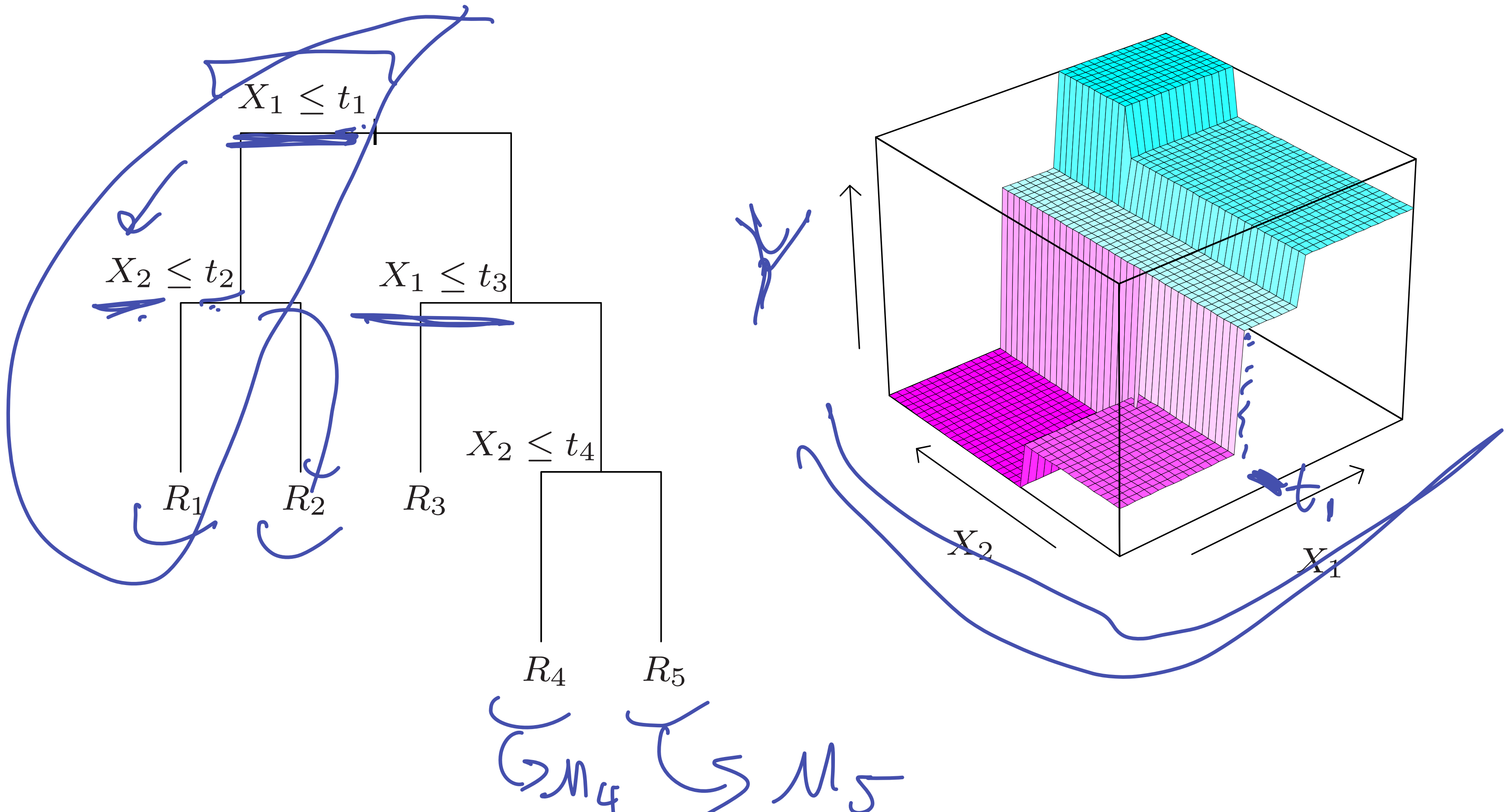
$$1\{x_1 < t\}\,1\{x_1 > s\}\,1\{x_2 < u\}\,1\{x_2 > v\}$$

New feature

Learning? Intractable

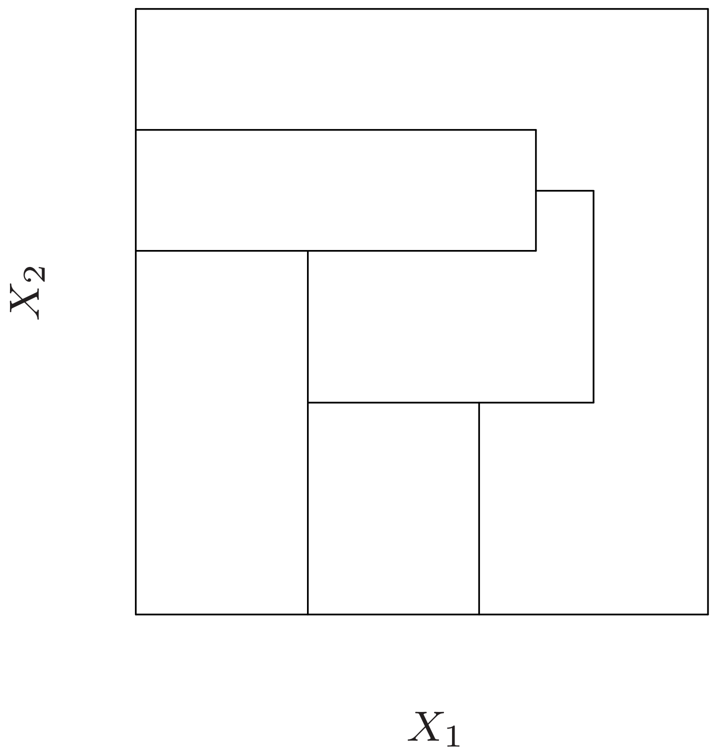
Recursive splitting

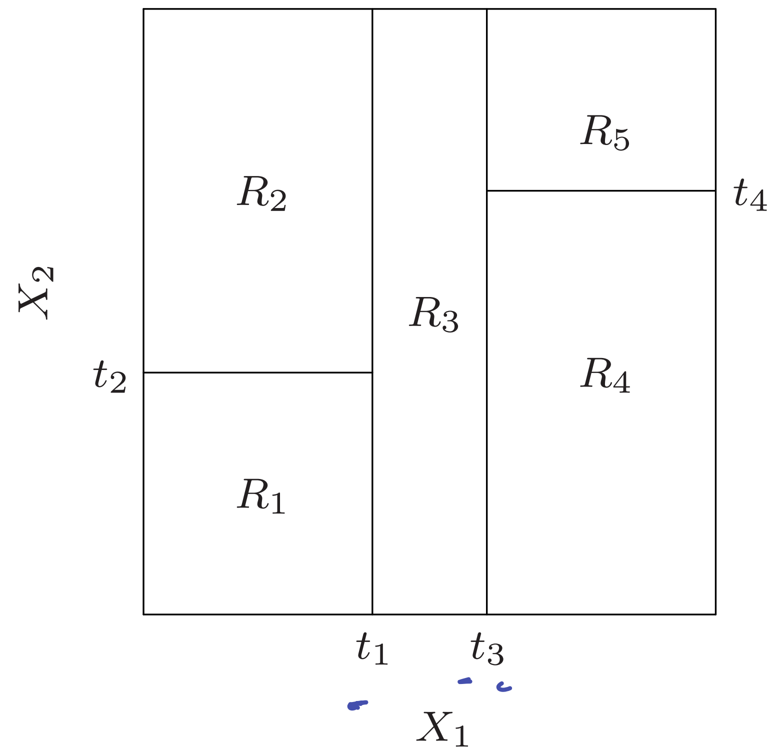
$$R_1 \iff (X_1 \le t_1) \wedge (X_2 \le t_2)$$



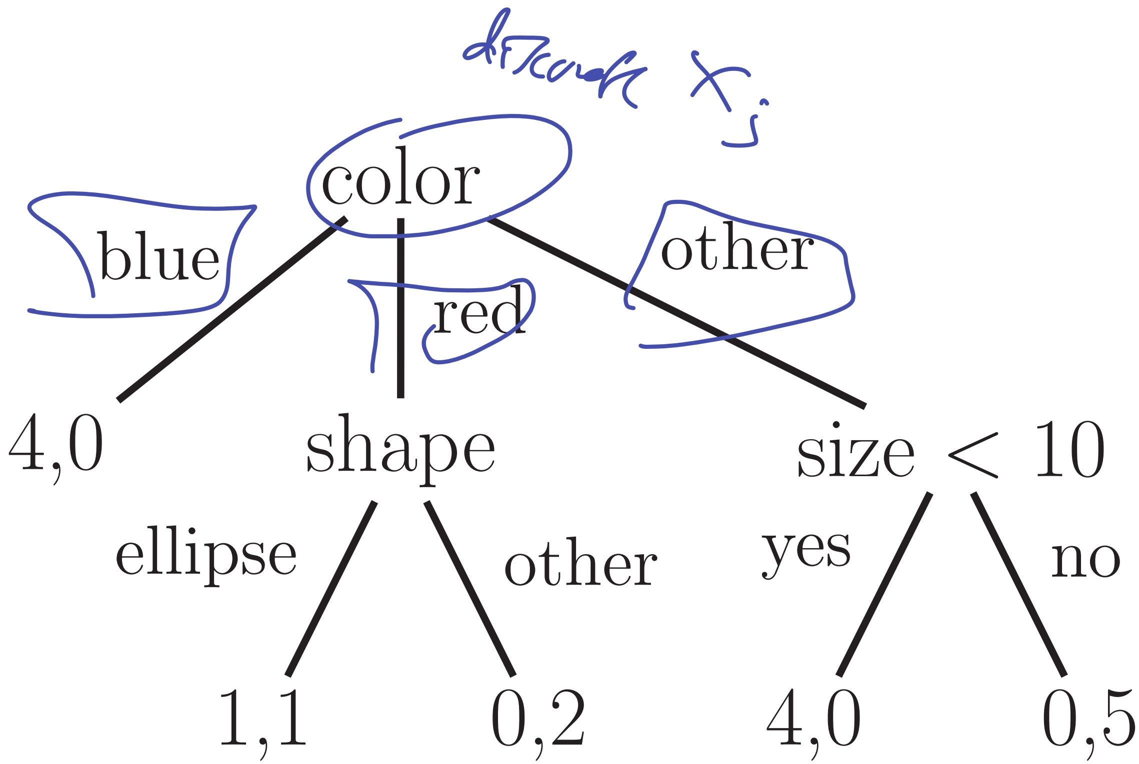
- From Hastie et al. 2009: *Elements of Statistical Learning*

Not representable
by DTree





- From Hastie et al. 2009: *Elements of Statistical Learning*


color
blue
red
other
4,0
shape
size < 10
ellipse
other
yes
no
1,1
0,2
4,0
0,5

Greedy decision tree learning

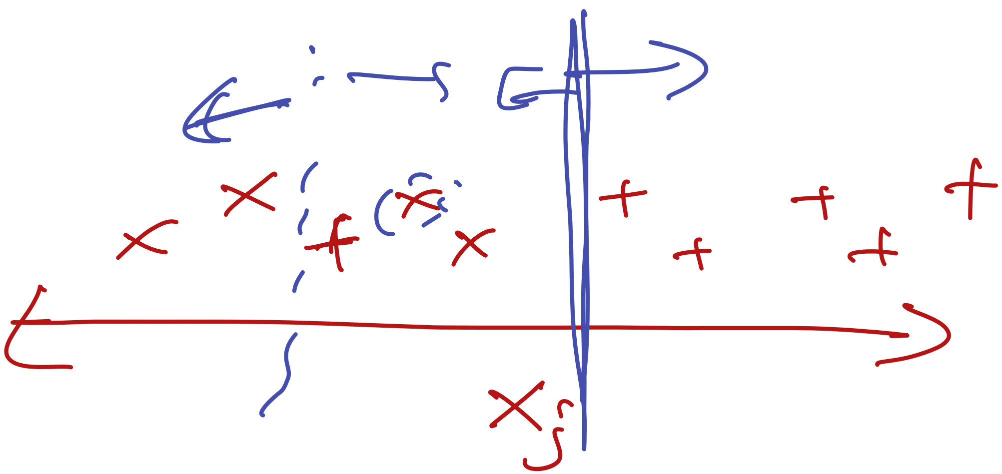
split(D): feature j, thresholds T_j

e.g. MSE

$$(j^*, t^*) = \arg \min_{j \in \{1,\ldots,D\}} \min_{t \in \mathcal{T}_j} \text{cost}(\{\mathbf{x}_i, y_i : x_{ij} \le t\}) + \text{cost}(\{\mathbf{x}_i, y_i : x_{ij} > t\})$$

dim.

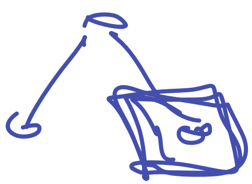
MSE : $\sum_{i \in D_L} (y_i - \bar{y}_{D_L})^2 + \sum_{i \in D_R} (y_i - \bar{y}_{D_R})^2$



Greedy decision tree learning

split(D): feature j, thresholds T_j

$$(j^*, t^*) = \arg \min_{j \in \{1,\ldots,D\}} \min_{t \in \mathcal{T}_j} \text{cost}(\{\mathbf{x}_i, y_i : x_{ij} \leq t\}) + \text{cost}(\{\mathbf{x}_i, y_i : x_{ij} > t\})$$



Algorithm 16.1: Recursive procedure to grow a classification/ regression tree

function fitTree(node, $\mathcal{D}$, depth) ;
node.prediction = mean($y_i : i \in \mathcal{D}$) // or class label distribution ;
$(j^*, t^*, \mathcal{D}_L, \mathcal{D}_R)$ = split($\mathcal{D}$);
if *not worthSplitting(depth, cost, $\mathcal{D}_L$, $\mathcal{D}_R$)* **then**
    return node
else
    node.test = $\lambda \mathbf{x}.x_{j^*} < t^*$ // anonymous function;
    node.left = fitTree(node, $\mathcal{D}_L$, depth+1);
    node.right = fitTree(node, $\mathcal{D}_R$, depth+1);
    return node;

maxdepth

Other details

- "Not worth splitting": overfitting control
    - Min # examples in node
    - Δ improvement to loss
- Max depth
- Pruning
- CART, C4.5, ...

Issues for single dtree learning

Pro

- Nonlinear
- Simple
- Interpretable (??)

Con

- Linear & gradual relationships
- Not axis-aligned
- Combinations?? Yes/No
- High variance / unstable wrt data perturbations

Generalized additive models

$$f(x) = \alpha + \sum_{m=1}^{M} f_m(x)$$

Basic nonlinear model

Ensemble models

Random forests

Heuristic ensemble of dtrees

M = num trees you want

For $m = 1 \ldots M$:

- Draw bootstrap sample $D^{(m)}$ of D:
  $i = 1 \ldots N$: $D_i^{(m)} \sim D^{\text{Train}}$
- Choose random subset of features $J^{(m)}$
- Fit tree f_m from $D^{(m)}, J^{(m)}$

Predict: $f(x) = \frac{1}{M} \sum_{m=1}^{M} f_m(x)$

Idea:
Average over Dtree instability

Works well!

Boosting

→ "weak learner"

- Use optimization ideas to formulate/explain iterative learning of successive decision trees

- Basic idea: learn to predict *residuals* from sum of previous trees

$$y \Rightarrow (y - \hat{y}) = r$$

Forward Stagewise Learning

Greedy learning for multiple dtrees

DTree   splits & preds

Init: $f_0(x) = \arg\min_{\gamma} \sum_i L(y_i, \phi(x_i; \gamma))$

CART or greedy recurs. splitter

DTree

for $m = 1 \ldots M$

$(\beta_m, \gamma_m) = \arg\min_{\beta, \gamma} \sum_i L(y_i, f_{m-1}(x_i) + \beta \phi(x_i; \gamma))$

$$f_m(x) = f_{m-1}(x) + \beta_m \phi(x; \gamma_m)$$

Better: $f_m(x) = f_{m-1}(x) + \nu \beta_m \phi(x; \gamma_m)$

$\in (0,1)$ step size!

L2Boosting

- Apply forward stagewise learning to squared error loss.

Gradient Boosting

Algorithm 16.4: Gradient boosting

Initialize $f_0(\mathbf{x}) = \text{argmin}_{\gamma} \sum_{i=1}^{N} L(y_i, \phi(\mathbf{x}_i; \gamma))$;
for $m = 1 : M$ **do**
  Compute the gradient residual using $r_{im} = -\left[\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}\right]_{f(\mathbf{x}_i)=f_{m-1}(\mathbf{x}_i)}$;
  Use the weak learner to compute $\boldsymbol{\gamma}_m$ which minimizes $\sum_{i=1}^{N}(r_{im} - \phi(\mathbf{x}_i; \boldsymbol{\gamma}_m))^2$;
  Update $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \nu\phi(\mathbf{x}; \boldsymbol{\gamma}_m)$;
Return $f(\mathbf{x}) = f_M(\mathbf{x})$