1/29/26

# Lecture 7

HW 1 —
[HW 2 —]
HW 3 out ←
Quiz 1 Feb 12

LIII CART

# Lecture II: Linear regression and classification. Loss functions

Marina Meilă
mmp@uwaterloo.ca

With Thanks to Pascal Poupart & Gautam Kamath
Cheriton School of Computer Science
University of Waterloo

January 12, 2026

## Predictors

- K-Nearest-Neighbor
- Linear – for regression
  – for classification
- Logistic regression ↗
- Perceptron, LDA
- Decision Trees (CART)

## Algorithms

LS Regression
Logistic Regression by
Gradient Ascent/Descent

## Concepts

- Decision Region, Dec. Boundary
- Training error, Test error
  Expected error ↗
- Variance, Bias
- Loss functions – training/
  test/expected loss
- Max Likelihood

Linear predictors generalities ✓

Loss functions ✓

Least squares linear regression ✓

- Linear regresssion as minimizing $L_{LS}$
- Linear regresssion as maximizing likelihood
- Linear Discriminant Analysis (LDA)
- QDA (Quadratic Discriminant Analysis)
- Logistic Regression ←
- The PERCEPTRON algorithm

**Reading** HTF Ch.: 2.1–5,2.9, 7.1–4 bias-variance tradeoff, Murphy Ch.: 1., 8.6[1], Bach Ch.:

[1] Neither textbook is close to these notes except in a few places; take them as alternative perspectives or related reading

# Logistic Regression

Fitting a linear predictor for classification, another approach.
Let $f(x) = \beta^T x$ model the **log odds** of class 1

$$f(X) = \frac{P(Y=1|X)}{P(Y=-1|X)} \tag{31}$$

Then

- $\hat{y} = 1$ iff $P(Y=1|X) > P(Y=-1|X)$
  - just like in the previous case! so what's the difference?
  - Answer: We don't assume each class is Gaussian, so we are in a more general situation than LDA
- What is $p(x) = P(Y=1|X=x)$ under our linear model?

$$\ln \frac{p}{1-p} = f, \quad \frac{p}{1-p} = e^f, \quad p = \frac{e^f}{1+e^f} \quad 1-p = \frac{1}{1+e^f} \tag{32}$$

An alternative "symmetric" expression for $p, 1-p$ is

$$p = \frac{e^{f/2}}{e^{f/2}+e^{-f/2}}, \quad 1-p = \frac{e^{-f/2}}{e^{f/2}+e^{-f/2}}. \tag{33}$$

# Estimating the parameters by Max Likelihood

- Denote $y_* = (1-y)/2 \in \{0,1\}$
- The likelihood of a data point is $P_{Y|X}(y|x) = \frac{e^{y_* f(x)}}{1+e^{f(x)}}$
- The log-likelihood is $l(\beta;x) = y_* f(x) - \ln(1+e^{f(x)})$
- $\frac{\partial l}{\partial f} = y_* - \frac{e^f}{1+e^f} = y_* - \frac{1}{1+e^{-f}}$
  This is a scalar, and $\mathrm{sgn}\frac{\partial l}{\partial f} = y$
- We have also $\frac{\partial f(x)}{\partial \beta} = x$
- Now, the gradient of $l$ w.r.t the parameter vector $\beta$ is

$$\frac{\partial l}{\partial \beta} = \frac{\partial l}{\partial f}\frac{\partial f}{\partial \beta} = \left(y_* - \frac{1}{1+e^{-f(x)}}\right)x \quad (34)$$

Interpretation: The infinitezimal change of $\beta$ to increase log-likelihood for a single data point is along the direction of $x$, with the sign of $y$

## Gradient of $\ell$ w.r.t $\beta$

$n=1$

$$\ell = y_* f(x) - \ln(1+e^{f(x)})$$

$$\frac{\partial \ell}{\partial f} = y_* - \frac{e^f}{1+e^f} \in \mathbb{R}$$

$\frac{e^f}{1+e^f}$: $P[y=+]$

$$\frac{\partial f}{\partial \beta} = \nabla f = \nabla_\beta (x^T\beta) = x \in \mathbb{R}^d$$

$$\nabla \ell \equiv \frac{\partial \ell}{\partial \beta} = \left(y_* - \frac{e^f}{1+e^f}\right) x = y \cdot w \cdot x$$

$y$: $\pm 1$

$y=+$ $y_*=1$ $\Rightarrow$ $\underbrace{1 - P[y^i=1|x^i]}_{w^i}$

$y=-$ $y_*=0$ $\Rightarrow$ $-P[y=1] = -(\underbrace{1-P[y_*=0]}_{w^i})$

## Step 4. (next lecture) Gradient ascent

$$\beta \leftarrow \beta + \eta \frac{\partial \ell}{\partial \beta}$$

$\eta > 0$ "step size"

## Step 3. Gradient

## Logistic Regression

Model $f(x) = \beta^T x$

predict $\hat{y} = +1$ iff $\geq 0$

$n>1$ **Training** $\mathcal{D} = \{(x^i, y^i), i=1:n\}$

$$\ell(\beta) = \sum_{i=1}^{n} [\, y_*^i f(x^i) - \ln(1+e^{f(x^i)})] \leftarrow \max_\beta \text{ (concave)}$$

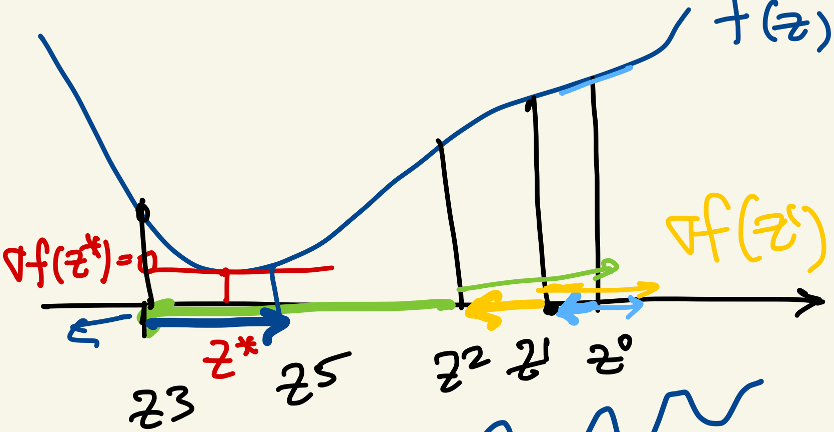
$y^i \in \pm 1 \Leftrightarrow y_*^i \in \{0,1\}$

**Loss** $L^{train}_{logit} = -\frac{1}{n} \ell(\beta) \leftarrow \min_\beta$ (convex) no local minima

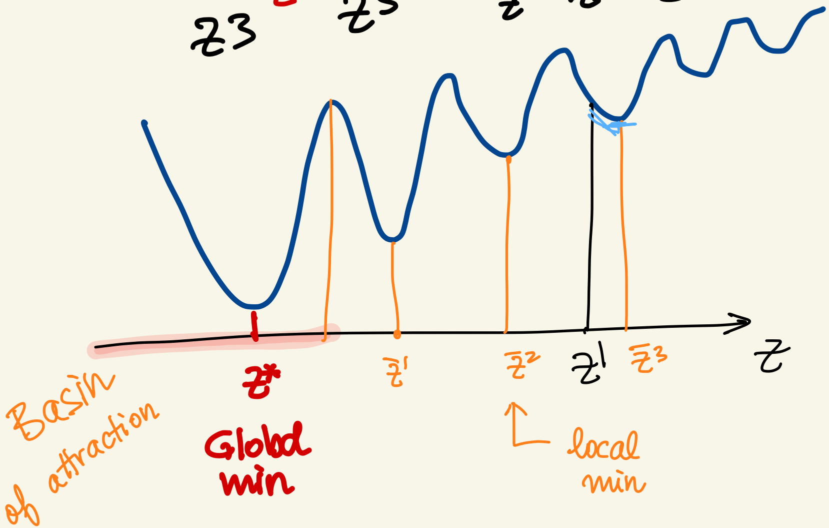
**∇Loss** $$\nabla L^{train}_{logit} = -\frac{1}{n} \sum_{i=1}^{n} \left( y_*^i - \frac{e^{f(x^i)}}{1+e^{f(x^i)}} \right) \cdot x^i$$

$y^i w_i \in \mathbb{R}$

$\nabla f$ = direction of fastest ↗

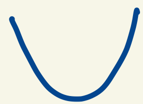



$z^1, z^2, \dots z^* \longrightarrow z^*$

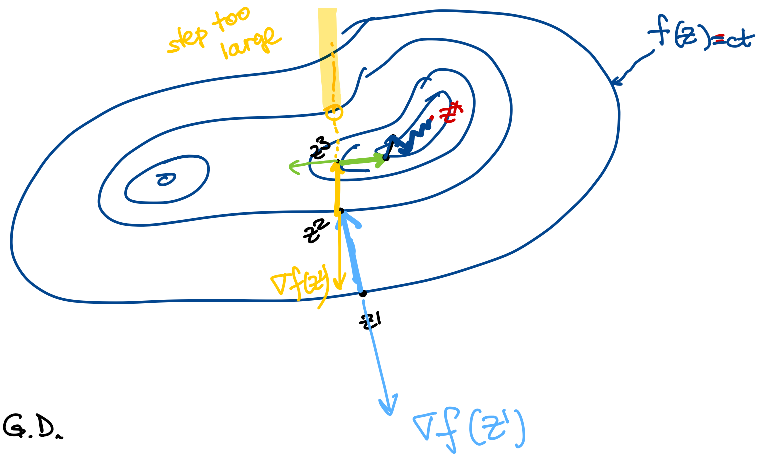



Grad. descent = Steepest descent

$f'' \geq 0 \Rightarrow f$ convex



$\left[\frac{\partial^2 f}{\partial z_i \partial z_j}\right] = H$ Hessian $\quad H \succeq 0 \Rightarrow f$ convex




G.D.

**Init** $z^0$

for $t = 1, 2, \dots$

$$z^t \leftarrow z^{t-1} - \eta \nabla f(z^{t-1})$$

step size

(compute $f(z^{t-1})$) has it increased?

## Gradient Ascent for $\max_\beta \ell(\beta)$

$$n\,\ell(\beta) = -L^{train}_{logit}$$

$$n\,\nabla\ell(\beta) = -\nabla L^{train}_{logit}$$

for $t$ compute $\ell(\beta^{t-1}), \nabla\ell(\beta^{t-1})$

$$\beta^t \leftarrow \beta^{t-1} + \eta n \nabla \ell(\beta^{t-1})$$

## Gradient Descent for Logistic Regression

"$L$" $\leftarrow L^{train}_{logit}$

**In** data $\mathcal{D}$, $\eta$, tol

**Init** $\beta^0 = 0$

**Do** for $t = 1, 2, \dots$

compute $L(\beta^{t-1})$, $\nabla L(\beta^{t-1})$

$$\beta^t \leftarrow \beta^{t-1} - \eta \nabla L(\beta^{t-1})$$

until $\dfrac{|L(\beta^{t-1}) - L(\beta^t)|}{L(\beta^{t-1})} < tol$

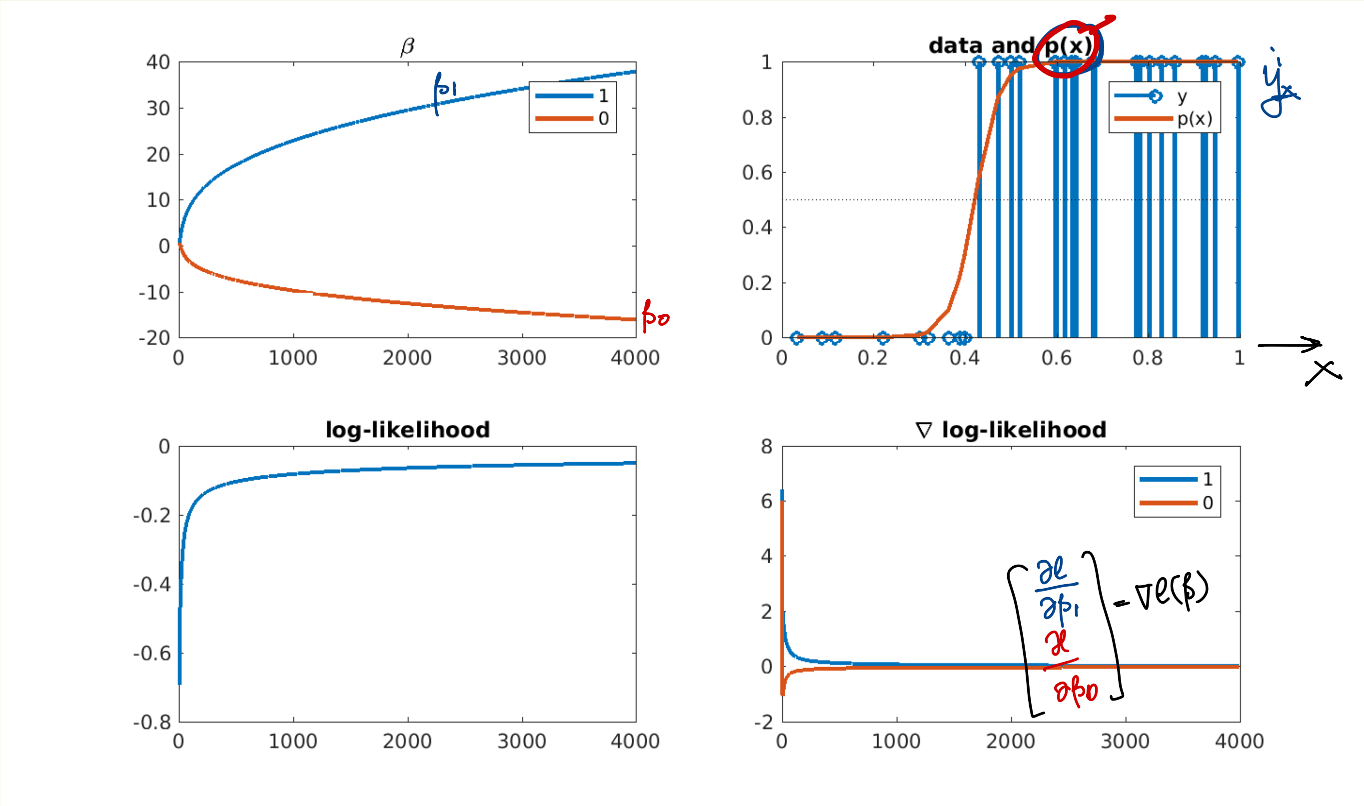
**Out** $\beta^t, L(\beta^t)$

$tol = 10^{-3}, \dots 10^{-8}$

$$f(x) = \beta_1 x + \beta_0$$

$$p_x = \frac{e^f}{1+e^f}$$

after training



# Estimating the parameters by Max Likelihood

- Denote $y_* = (1-y)/2 \in \{0,1\}$
- The likelihood of a data point is $P_{Y|X}(y|x) = \frac{e^{y_* f(x)}}{1+e^{f(x)}}$
- The log-likelihood is $l(\beta; x) = y_* f(x) - \ln(1+e^{f(x)})$
- $\frac{\partial l}{\partial f} = y_* - \frac{e^f}{1+e^f} = y_* - \frac{1}{1+e^{-f}}$
  This is a scalar, and $\operatorname{sgn}\frac{\partial l}{\partial f} = y$
- We have also $\frac{\partial f(x)}{\partial \beta} = x$
- Now, the gradient of $l$ w.r.t the parameter vector $\beta$ is

$$\frac{\partial l}{\partial \beta} = \frac{\partial l}{\partial f}\frac{\partial f}{\partial \beta} = \left(y_* - \frac{1}{1+e^{-f(x)}}\right) x \qquad (34)$$

  Interpretation: The infinitezimal change of $\beta$ to increase log-likelihood for a single data point is along the direction of $x$, with the sign of $y$
- Log-likelihood of the data set $\mathcal{D}$

$$l(\beta; \mathcal{D}) = \frac{1}{N}\sum_{i=1}^{d} l(\beta; (x^i, y^i)) \qquad (35)$$

- The optimal $\beta$ maximizes $l(\beta; \mathcal{D})$ and therefore

$$\frac{\partial l(\beta; \mathcal{D})}{\partial \beta} = \frac{1}{N}\sum_{i=1}^{d}\left(y_*^i - \frac{1}{1+e^{-f(x^i)}}\right) x^i = 0 \qquad (36)$$

- Unfortunately, (36) does not have a closed form solution!
  We maximize the (log)likelihood by iterative methods (e.g. gradient ascent) to obtain the $\beta$ of the classifier.

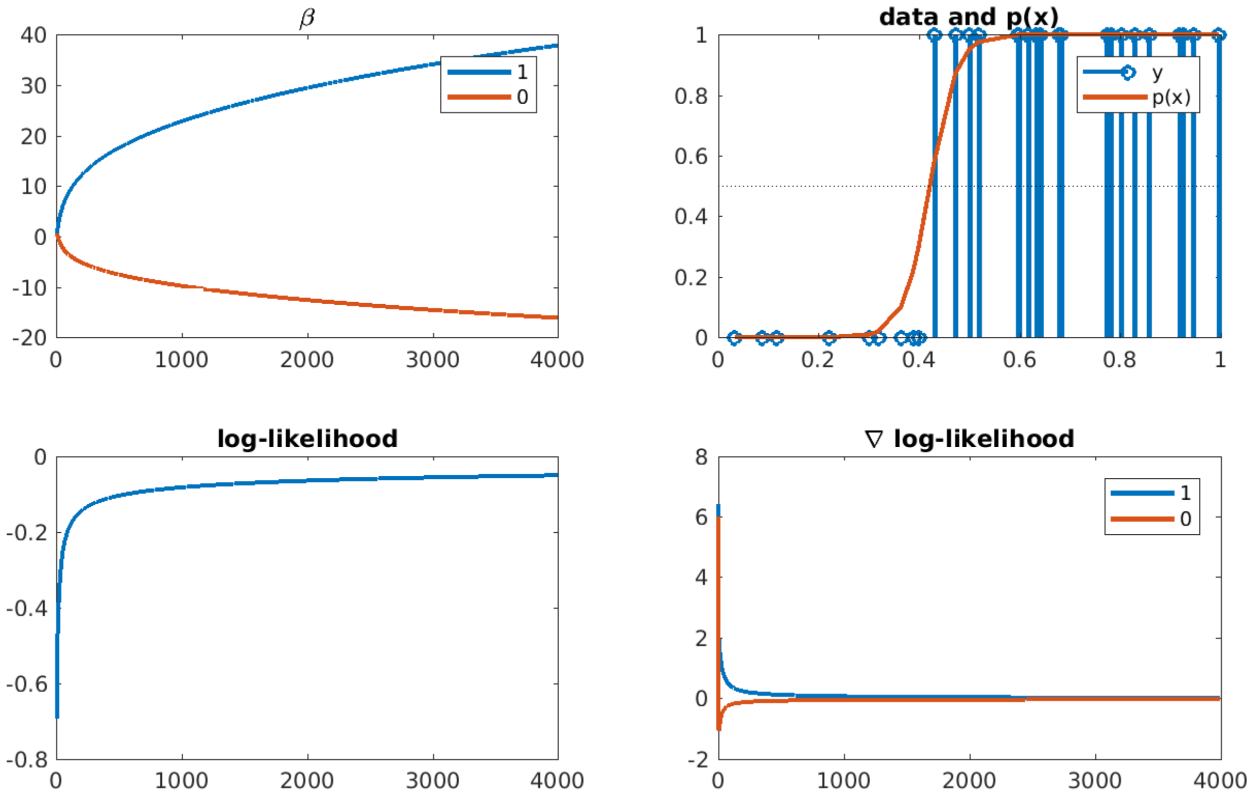
# The gradient – an alternative formula

- We use the original $y$ values instead of $y_*$
- Note that

$$P_{Y|X}(y|x) = \frac{1}{1+e^{-yf(x)}} = \phi(yf(x)) \tag{37}$$

- with $\phi' = \phi(1-\phi)$
- Then, $\frac{\partial \ln P_{Y|X}(y|x)}{\partial f} = \frac{\partial \ln \phi(yf)}{\partial f} = \frac{y\phi(yf)(1-\phi(yf))}{\phi(yf)} = y(1-\phi(yf))$
- The gradient of the log-likelihood of the data is now

$$\frac{\partial l(\beta;\mathcal{D})}{\partial \beta} = \frac{1}{N}\sum_{i=1}^{d}\left(1-\underbrace{\phi(e^{yf(x^i)})}_{P_{Y|X}(y_i|x^i,\beta)}\right)y_i x^i \tag{38}$$



β
1
0
data and p(x)
y
p(x)
log-likelihood
∇ log-likelihood
1
0

# Lecture III: Classification and Decision Trees (CART)

Marina Meilă
mmp@uwaterloo.ca

With Thanks to Pascal Poupart & Gautam Kamath
Cheriton School of Computer Science
University of Waterloo

January 27, 2026

Classification and regression tree(s) (CART)

Learnin a CART

Predicting with a CART

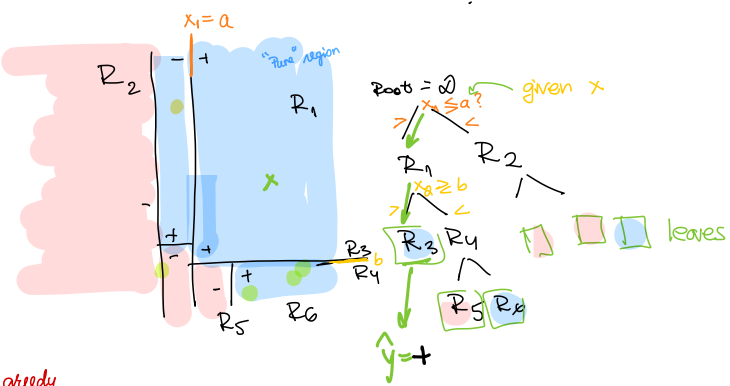
Some issues with CART

**Reading** HTF Ch.: 9.2 CART, Murphy Ch.: 16.2.1–4 CART, Bach Ch.:

# Classification and regression trees (CART)

- A **classification tree** or (**decision tree**) is built recursively by splitting the data with hyperplanes parallel to the coordinate axes.
  - At each split, try to separate $+$ examples from $-$ examples as well as possible.
  - Eventually, all the regions will be "pure", i.e. will contain examples from one class only.
- Classification trees can be used in multiway classification as well (there we try to create a pure region on at least one side of the split)
- A **regression tree** uses the same principle for regression
  here we try to obtain regions where the outputs are nearly the same

# Classification Tree (Decision Tree)




Training – greedy
– recursive

Prediction – recursive down the tree

# Hierarchical partitions

- a **hierarchical partition** $\mathcal{T}$ of $\mathbb{R}^d$ is a set of regions $\{R_q\}$, so that $\mathbb{R}^d = \bigcup_q R_q$ and between any two $R_q, R_{q'}$ we have either

$$R_q \cap R_{q'} = \emptyset, \text{ or } R_q \subset R_{q'} \text{ or } R_{q'} \subset R_q. \tag{1}$$

  (we include the boundariy between 2 regions $R_q, R_{q'}$ arbitrarily in a single one of them)
- In a CART, the partitions are usually chosen to be axis-aligned, i.e. $R_q = \{x \,|\, \pm x_{j_1} \text{"} > \text{"} \tau_1, \pm x_{j_2} \text{"} > \text{"} \tau_2, \ldots \pm x_{j_l} \text{"} > \text{"} \tau_l\}$ where " $>$ " stands for one of $>$ or $\geq$, so that the union of all regions covers $\mathbb{R}^d$.
- The number of inequalities $l$ defining the region is called the *level* of the region.
- $R_q$ is a **leaf** of $\mathcal{T}$ iff there is no other $R_{q'}$ included in it.

## Example (**A hierarchical partition with 3 levels over $\mathbb{R}^2$**)

Level 1: $R_1 = \{x \,|\, x_2 > 3\}$,
$R_2 = \{x \,|\, x_2 \leq 3\}$

Level 2: $R_3 = \{x \,|\, x_2 > 3,\ x_1 \geq -2\}$,
$R_4 = \{x \,|\, x_2 > 3,\ x_1 < -2\}$,
$R_5 = \{x \,|\, x_2 \leq 3, x_1 > 0\}$,
$R_6 = \{x \,|\, x_2 \leq 3, x_1 \leq 0\}$

Level 3: $R_7 = \{x \,|\, x_2 > 3,\ x_1 \geq -2, x_1 < 4\}$,
$R_8 = \{x \,|\, x_2 > 3,\ x_1 \geq 4\}$,
$R_9 = \{x \,|\, x_2 < 3, x_1 \geq 1\}$
$R_{10} = \{x \,|\, x_2 \leq 3, x_1 \leq 0, x_2 > -1\}$,
$R_{11} = \{x \,|\, x_2 \leq -1, x_1 \leq 0\}$,
$R_{12} = \{x \,|\, x_2 < 3, x_1 > 0, x_1 < 1\}$

The leaves are $R_4, R_7, \ldots R_{12}$. Not all leaves are at the same level; for example $R_4$ is at level 2.

# "Learning" a CART

A standard algorithm for building a decision tree works recursively in top-down fashion.

**Input** Training set $\mathcal{D}$ of size $n$
**Initialize** with a tree with only one region, that contains all the data
Repeat until all leaves are pure (or until desired purity is attained in all leaves)

1.
2. Find the "optimal" split over all leaves $R_q$ and all possible splits of $R_q$.
   "Optimal" is defined in terms on purity (e.g split the least pure leaf, find the split that makes one of the new leaves pure)
3. Perform the "optimal" split and add the two new leaves to the tree

This is a greedy algorithm. Sometimes, trees obtained this way are pruned back to smaller sizes.