CS 480/680

2/3/26

Lecture 8

Sol 2 – pick it up

Hw 3 - due tomorrow

Hw 4 - out –"–

NOT GRADED

LIV – posted.

Edits still t.b.d

$\boxed{\text{Q1 on Feb 12} \\ \text{starts 11:32 sharp}}$

Decision Trees

Neural nets

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

CART Greedy algorithm

Concepts

- Decision Region, Dec. Boundary

Training error, Test error
Expected error ↗

Variance, Bias

Loss functions - training/
test/expected loss

Max likelihood

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 $\leftarrow$

Predicting with a CART ✓

Some issues with CART $\leftarrow$

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

Some advantages of CART

- Natural and easy to interpret (if small)
- Can approximate any function (with enough leaves)

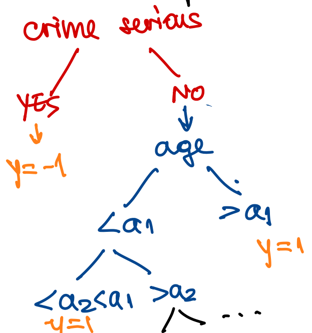
Flexible
- can classify / or predict any finite $\mathcal{D}$ with 0 error (= overfitting likely)
- can approximate any decision boundary with any precision

low bias
(high variance)

Ex 1. Predict recidivism

age
gender
crime
time served
...

transparent, objective



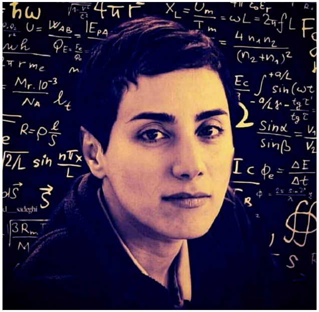
Some advantages of CART

?

- Natural and easy to interpret (if small) ← ONLY if features interpretable
- Can approximate any function (with enough leaves)

Ex2: Classify facial expression

x = image, $y = \{$ ☺ , ☹ , 😐 $\}$
happy, sad, neither



$X_L = \frac{U_m}{I_m} = \omega L = 2\pi f L$
$\frac{4 n_1 n_2}{(n_2+n_1)^2}$
R = ρ l/S
sadeghi

$x \in \mathbb{R}^{500 \times 500}$

→ D.Tree → y = 😐

• NOT interpretable
• NOT UNIQUE

HUGE Tree

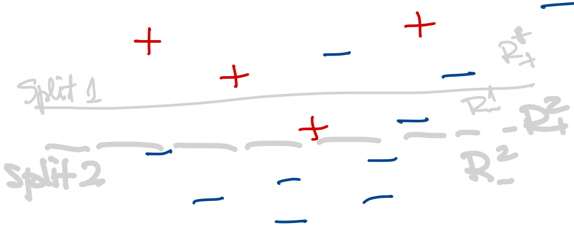
“Learning” a CART

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)

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.

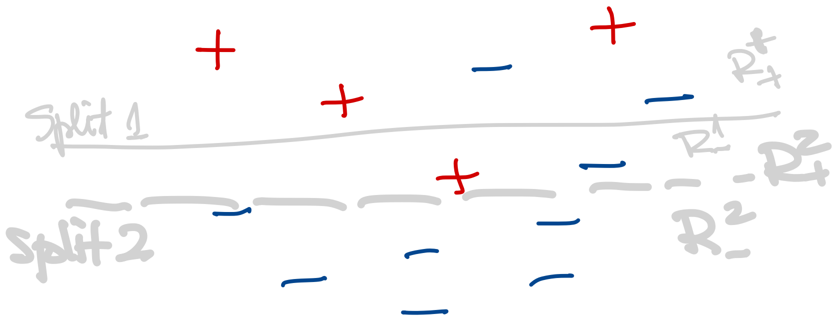


$\underline{\text{Init}}$ $R_0 = \mathcal{D}$, impurity
$I_0 = \underline{\underline{\text{one of}}}$ MS, Gini, Entropy
$\hat{y}_0 = -$
$p_0 = \frac{9}{13}$

$n = 13$
$n_+ = 4$

$I(R^1_+) + I(R^1_-) \overset{?}{\geq} I(R^2_+) + I(R^2_-)$

Recursively for all possible splits:
- choose split that minimizes Impurity

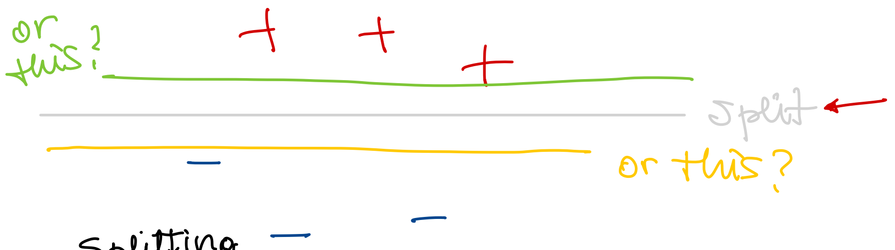


At leaf q with n_q points

```
for $j = 1:d$  (each dim)
    for $i = 1:n_q - 1$
        compute $I(q, \text{split } i, \text{dim} = j)$  ← $i$ points with smallest $x_j$'s
```

$(n_q - 1) \times d$ splits

STOP when all R_q have $I_q \leq$ threshold



$$\frac{\min_{+} x_j + \max_{-} x_j}{2}$$

halfway between $+$ $-$

Splitting

```
for leaf $g = 1 : n_{leaves}$
    for $j = 1:d$        dimension of $x$
        for $i = 1 : n_g - 1$        nodes of leaf $g$
            compute $\underline{\Delta I(g,j,i)}$
    split leaf $g^* = \arg\max_{g,j,i} \Delta I(g,j,i)$
```

$\Delta I(g,j,i) =$

$$= I_g - \left(I_{g'}(j,i) + I_{g''}(j,i) \right)$$

decrease in impurity

Purity

- Natural ways to set y_q based on the data, once the partition $\mathcal{T}$ has been fixed:
  - denote $Y_q = \{y^i \mid x^i \in R_q,\ i = 1:N\}$ the set of labels at a leaf R_q
  - Regression y_q = average of Y_q
  - Classification y_q = majority label of Y_q
- a leaf R_q is **pure** if all labels are the same, i.e. if $|Y_q| = 1$
- criteria for the **(im)purity** of a leaf R_q
  - Regression impurity = sample variance of Y_q
  - Classification let p_q be the frequency of y_q in Y_q

$$\text{impurity} = \begin{cases} \text{Misclassification error} & 1 - p_q \\ \text{Gini} & p_q(1-p_q) = \text{Var} \\ \text{Entropy} & -\left[p_q \ln p_q + (1-p_q)\ln(1-p_q) \right] \end{cases} \tag{2}$$

These measures generalize naturally to the multiclass setting.

$$I(p_0) = 1 - \frac{9}{13} = \frac{4}{13}$$

$$I(p_0) = \frac{4}{13}\frac{9}{13} = \frac{36}{13^2}$$

$$I(p_0) = \frac{4}{13} \cdot \ln\frac{13}{4} + \frac{9}{13}\ln\frac{13}{9}$$

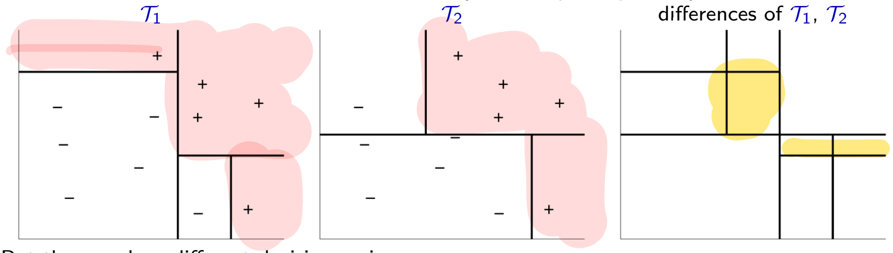
Predicting with a CART

Given new x

1. Find the unique leaf $R(x)$ so that $x \in R(x)$
2. Predict $\hat{y}$ based on the data in this leaf
   - **Regression**
     Predict $\hat{y}(x) = \mathrm{average}\{y^i \text{ with } x^i \in R(x)\}$
   - **Classification**
     Predict $\hat{y}(x) = \mathrm{majority}\{y^i \text{ with } x^i \in R(x)\}$

A decision tree over $\mathcal{D}$ is not unique

Same dataset $\mathcal{D}$, two different trees. Both classify the sample $\mathcal{D}$ perfectly.



But they produce different decision regions.

Lecture Notes IV – Neural Networks, Part 1

Marina Meilă
mmp@uwaterloo.ca

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

February 2, 2026

A little history

The single “neuron”

Two-layer Neural Networks
- Hidden layer options
- Output layer options

Multi-layer neural networks

Reading HTF Ch.: 11.3 Neural networks, Murphy Ch.: (16.5 neural nets), Bach Ch.: –, Deep Learning Book (Goodfellow, Bengio, Courville) 6.1-4, ResNet 7.6, ConvNet 9., Autoencoders 14.1, Dive Into Deep Learning 4.1-4.3.

A little history

First epoch – can computers mimic the brain?

- 40's-50's the first mathematical models of the neuron (McCullaugh & Pitts, Hebb)
- '58 the perceptron (Rosenblatt)
- First winter '69 "Perceptrons" (Minsky and Papert) – they can only classify linearly separable data (and von Neumann computers steal the show)

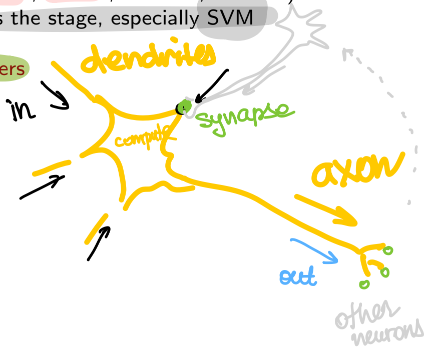
Revival in the 80's-'95 – let's use layers!

- '82 Hopfield associative net (contents adressable memory)
- '84 Boltzmann Machine (Ackley, Hinton, Sejnowski)
- autoencoders
- '86 backpropagation (Sejnowski, Rosenberg)
- Reinforcement learning (Shannon, Samuel, Barton, Sutton, Watkins, Tesauro)
- Second winter cca '95 Statistical learning takes the stage, especially SVM

Current epoch cca 2005 – more data, more layers

- Deep learning
- Generative models
- Attention
- LLMs

Neuron = unit



Brains vs. Computers

Computer (von Neumann)

- Electrical binary signals directed by gates
- Wiring is fixed
- Sequential and parallel computation
- Memory is retrieved by address
- **Fragile** (if a gate stops working, computer crashes). Also, "no friction" – a single instruction can stop the entire machine/program.

- **Brain**
- Electrical signals, units=neurons
- Wiring is dynamic, changes with brain development, experiences, learning
- Parallel (and some) sequential computation
- Memory is distributed
- **Robust** (when neuron/region dies, brain rewires itself to compensate). No On/Off master switch

Brains vs. Computers

Computer (von Neumann)

- Electrical binary signals directed by gates
- Wiring is fixed
- Sequential and parallel computation
- Memory is retrieved by address
- **Fragile** (if a gate stops working, computer crashes). Also, "no friction" – a single instruction can stop the entire machine/program.

- **Brain**
- Electrical signals, units=neurons
- Wiring is dynamic, changes with brain development, experiences, learning
- Parallel (and some) sequential computation
- Memory is distributed
- **Robust** (when neuron/region dies, brain rewires itself to compensate). No On/Off master switch

- **Neural network**
- Signals are numbers passed between units
- Network structure is fixed (and dense) but the learned weights allow "rewiring" during training
- Parallel (in layer) and sequential (feed forward/backward) computation
- Memory is distributed (in the weights)
- **Redundant/robust** (no single neuron can influence the output much

(Artificial) Neural Network (nn) unit

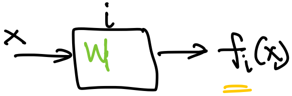
$x \in \mathbb{R}^d$

$w_i \in \mathbb{R}^{d+1}$

- For each **unit** i

$$y_i \equiv f_i(x) = \overset{\text{non-linear}}{\phi}\Big(\underbrace{\sum_j w_{ij}x_j + w_{i0}}_{\text{linear}}\Big) \qquad (1)$$

- **Weigth vector** w_i
  - w_{ij} = strength of the link from unit i to input j
  - $w_{ij} = 0$: no link
  - w_{ij} can be positive or negative
  - Sometimes we call the input vector $x = [x_{1:d}]$ input units



- **activation function** $\phi()$
  - must be non-linear (otherwise the unit is a linear transformation)
  - wanted: monotonically increasing, differentiable, gradient non-zero [1]

[1] More technically: ϕ can be any continuous, bounded and strictly increasing function on $\mathbb{R}$.

(Artificial) Neural Network (nn) unit

- For each **unit** i

$$y_i \equiv f_i(x) = \phi(\sum_j w_{ij}x_j + w_{i0}) \qquad (1)$$

- **Weigth vector** w_i
  - $w_{ij} =$ strength of the link from unit i to input j
  - $w_{ij} = 0$: no link
  - w_{ij} can be positive or negative
  - Sometimes we call the input vector $x = [x_{1:d}]$ input units
- **activation function** $\phi()$
  - must be non-linear (otherwise the unit is a linear transformation)
  - wanted: monotonically increasing, differentiable, gradient non-zero [1]

Notation ϕ is overloaded

- When we talk about nn in general: ϕ is any activation function
- When we do calculations with nn: ϕ is by default the logistic function (unless specified otherwise)

logistic or **sigmoid** function $\quad \phi(u) = \dfrac{1}{1+e^{-u}} \qquad (2)$

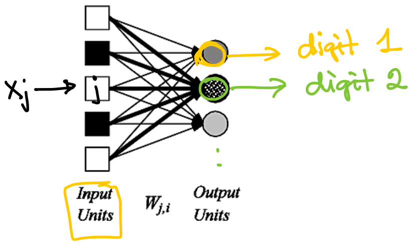
- When we do statistics or ML (but not nn): ϕ is the logistic function

- Exercise: compare $f_i(x)$ from (1) with $p(x) = Pr[y = 1 \,|\, x]$ from logistic regression.

[1] More technically: ϕ can be any continuous, bounded and strictly increasing function on $\mathbb{R}$.

Perceptron

- Single layer feed-forward network



$x_{1:d}$

UNIVERSITY OF WATERLOO