GHENT UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

E016350 - Artificial Intelligence

Lecture 4

Machine learning

Learning with Nonlinear Features and Decision Trees

Aleksandra Pizurica

Ghent University

Spring 2024

Outline

1. Multiclass classification
2. Learning with nonlinear features
3. Decision trees

[R&N], Chapter 19

The slides are based on: S. Russel and P. Norvig: *Artificial Intelligence: A Modern Approach*, (Fourth Ed.), http://aima.cs.berkeley.edu/; D. Klein & P. Abbeel: CS188 Artificial Intelligence (Berkeley) and M. Charikar & Koyejo: CS221 Artificial Intelligence: Principles and Techniques (Stanford).

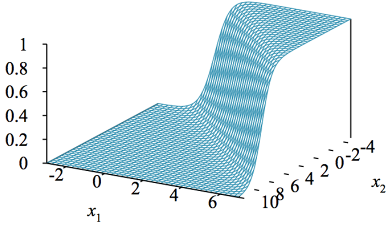
Outline

Logistic regression - Reminder

We consider binary classification: to each input data point $\mathbf{x} \in \mathbb{R}^d$ we assign a class label $y \in \{0, 1\}$.

Let $g(z)$ denote the logistic (sigmoid) function:

$$g(z) = Logistic(z) = \frac{1}{1 + e^{-z}}$$



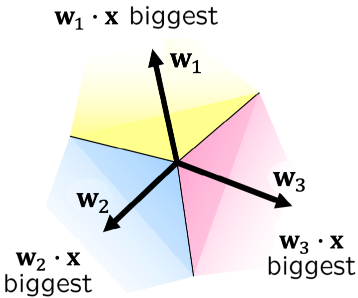
For some weight vector $\mathbf{w} \in \mathbb{R}^d$, the hypothesis

$$h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$

can be interpreted as the probability that $\mathbf{x}$ belongs to class 1, i.e., the probability that $y = 1$.

Multiclass Logistic Regression

- Multi-class linear classification
  - A weight vector for each class: $\mathbf{w}_y$
  - Score (activation) of a class y : $\mathbf{w}_y \cdot \mathbf{x}$
  - Prediction "the highest score wins": $\arg\max_y \mathbf{w}_y \cdot \mathbf{x}$



- How to turn the scores into probabilities?

$$\underbrace{z_1, z_2, z_3}_{\text{original activations}} \rightarrow \underbrace{\frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}}, \frac{e^{z_2}}{e^{z_1}+e^{z_2}+e^{z_3}}, \frac{e^{z_3}}{e^{z_1}+e^{z_2}+e^{z_3}}}_{\text{softmax activations}}$$

- In general, for K classes:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$$

Finding the best weights

Maximum likelihood estimation

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \arg\max_{\mathbf{w}} \prod_{i=1}^{N} P(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w})$$

with

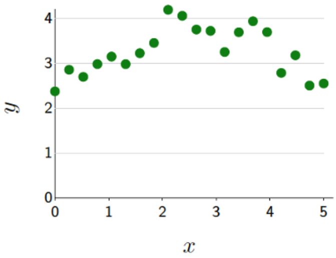
$$P(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}) = \frac{e^{\mathbf{w}_{y^{(i)}} \cdot \mathbf{x}^{(i)}}}{\sum_{y} e^{\mathbf{w}_{y^{(i)}} \cdot \mathbf{x}^{(i)}}}$$

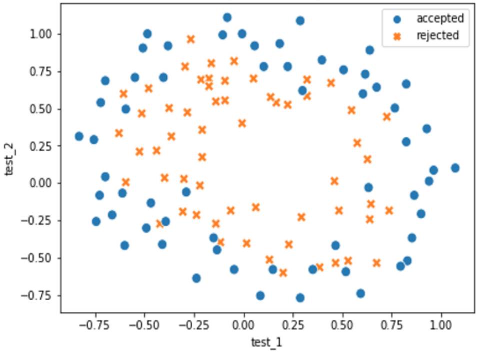
(softmax activations serve as probabilities)

Outline

More complex data

So far we analysed linear regression and linear classification (with logistic regression).
– But in many cases data show nonlinear trends / nonlinear separability.





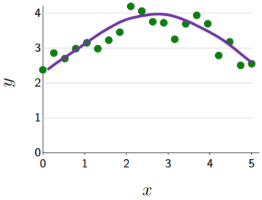
Can we handle such more complex data with the machinery of linear predictors?

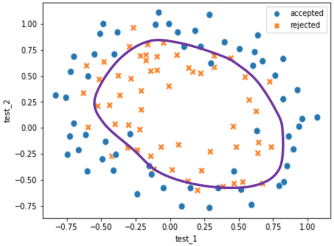
Linear predictors with nonlinear features

Idea: extract a vector of nonlinear features $\phi(x)$ from input x (no matter its dimension) and feed $\phi(x)$ to a linear predictor. It's going to be non-linear in x!

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w} \cdot \phi(x)$$

$$h_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \phi(x)}}$$

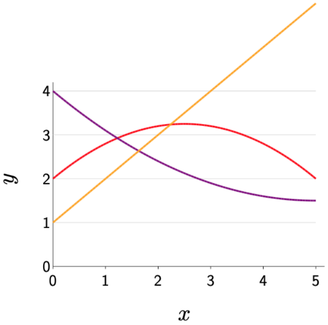




Quadratic predictors

$\phi(x) = [1, x, x^2]$ (Example: $\phi(3) = [1, 3, 9]$)

$$h_{\mathbf{w}}(x) = [2, 1, -0.2] \cdot \phi(x)$$
$$h_{\mathbf{w}}(x) = [4, -1, 0.1] \cdot \phi(x)$$
$$h_{\mathbf{w}}(x) = [1, 1, 0] \cdot \phi(x)$$



Hypothesis space:

$$\mathcal{H} = \{h_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x) : \mathbf{w} \in \mathbb{R}^3\}$$

Non-linear predictors just by changing ϕ !

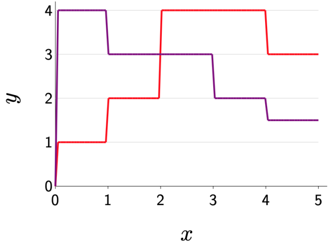
Slide credit: M. Charikar and Koyejo: Artificial Intelligence: Principles and Techniques (Stanford)

Piece-wise constant predictors

$$\phi(x) = [\mathbf{1}[0 < x \leq 1], \mathbf{1}[1 < x \leq 2], \mathbf{1}[2 < x \leq 3], \mathbf{1}[3 < x \leq 4], \mathbf{1}[4 < x \leq 5]]$$

(Example: $\phi(2.3) = [0, 0, 1, 0, 0]$)

$$h_{\mathbf{w}}(x) = [1, 2, 4, 4, 3] \cdot \phi(x)$$
$$h_{\mathbf{w}}(x) = [4, 3, 3, 2, 1.5] \cdot \phi(x)$$



Hypothesis space:

$$\mathcal{H} = \{h_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x) : \mathbf{w} \in \mathbb{R}^5\}$$

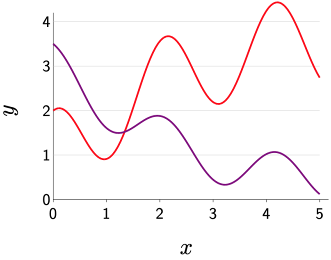
Expressive non-linear predictors by partitioning the input space.

Slide credit: M. Charikar and Koyejo: Artificial Intelligence: Principles and Techniques (Stanford)

Predictors with periodicity structure

$\phi(x) = [1, x, x^2, \cos(3x)]$

(Example: $\phi(2) = [1, 2, 4, 0.96]$)



$h_{\mathbf{w}}(x) = [1, 1, -0.1, 1] \cdot \phi(x)$

$h_{\mathbf{w}}(x) = [3, -1, 0.1, 0.5] \cdot \phi(x)$

Hypothesis space:

$\mathcal{H} = \{h_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x) : \mathbf{w} \in \mathbb{R}^4\}$

Just throw in any features you want!

Slide credit: M. Charikar and Koyejo: Artificial Intelligence: Principles and Techniques (Stanford)

Quadratic classifiers

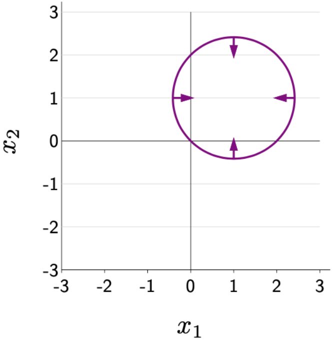
$$\phi(x) = [x_1, x_2, x_1^2 + x_2^2]$$

A linear classifier with a hard threshold:

$$h_{\mathbf{w}}(\mathbf{x}) = Threshold\Big([2, 2, -1]) \cdot \phi(x)\Big)$$

is now equivalent to:

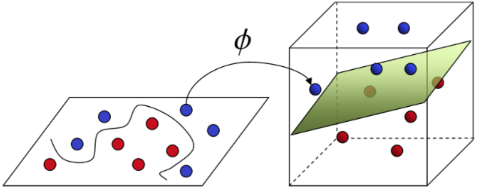
$$h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1 & (x_1 - 1)^2 + (x_2 - 1)^2 \leq 2 \\ 0 & \text{otherwise} \end{cases}$$



Decision boundary is a circle.

Slide credit: M. Charikar and Koyejo: Artificial Intelligence: Principles and Techniques (Stanford)

In essence, ...



A general concept (kernel trick in SVM)

– We'll return to it later

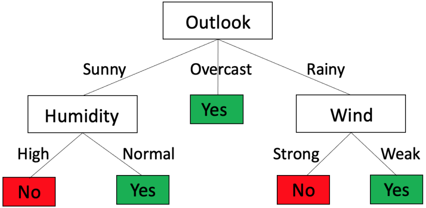
Picture credit: Drew Wilimitis: The Kernel Trick in Support Vector Classification

Outline

Decision trees

- Decision trees are able to learn complex, nonlinear relationships between variables, using a series of simple, **intuitive** decision rules.
- Easy to undersand and interpret. Require little or no data preparation.
- Widely used in today's machine learning approaches.

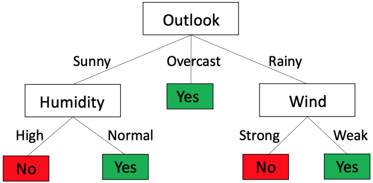
Example: Should I play tennis today?



A simple idea: start with one test, and depending on its outcome decide what the next test will be. Continue until a decision is reached.

Interpretation of a decision tree

Like any supervised ML approach, a decision tree is learned from $(\mathbf{x}, y) \in \mathcal{D}_{train}$, where $\mathbf{x}$ are the values of some features (or attributes) $\mathbf{X}$ and y is the output label.



- Internal nodes test a feature X_i

  In this tree: $X_1 = Outlook$, $X_2 = Humidity$, $X_3 = Wind$
- Branching is determined by the feature value

  E.g. $x_3 = wind \in \{Strong, Weak\}$
- Leaf nodes are outputs (predictions):
  - numerical (regression tree); categorical (classification tree)
  - tuple-valued variable (multi-target trees) or $P(y|\mathbf{x})$ (probability estimation trees)

Case study: "Restaurant domain"

Decide whether to wait for a table in a restaurant depending on the following attributes (R&N):

1. Alternate (Alt): Is there a suitable alternative restaurant nearby?
2. Bar (Bar): Is there a comfortable bar area in the restaurant, where I can wait?
3. Fri/Sat (Fri): True on Fridays/Saturdays
4. Hungry (Hun): Are we hungry?
5. Patrons (Pat): How many people are in the restaurant ($None$, $Some$ or $Full$)
6. Price ($Price$): the restaurant's price range (\$, \$\$, \$\$\$)
7. Raining ($Rain$): Is it raining outside?
8. Reservation (Res): Did we make a reservation?
9. Type ($Type$): the kind of restaurant (French, Italian, Thai or burger)
10. WaitEstimate (Est): the wait time estimated by the host (0-10, 10-30, 30-60, or>60 min)

Decision trees

Examples for the restaurant domain R&N, table 19.2 (adapted notation)

	Input Attributes										Output
Example	Alt	Bar	Fri	Hun	Pat	$Price$	$Rain$	Res	$Type$	Est	*WillWait*
1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
10	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
11	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
12	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

Each raw is an example $(\mathbf{x}^{(i)}, y^{(i)})$, where the output $y^{(i)}$ is true (T) or false (F).

Decision trees

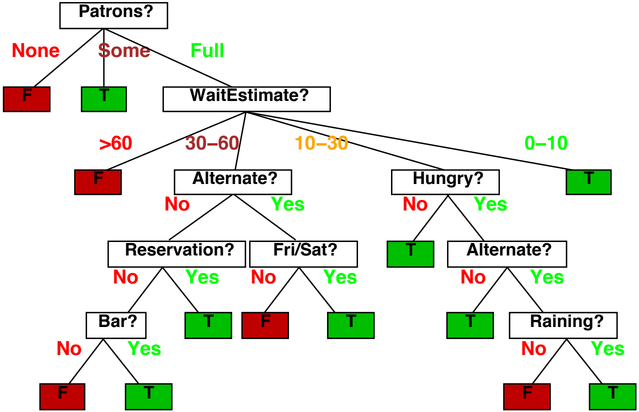
Examples for the restaurant domain R&N, table 19.2 (adapted notation)

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
1	T	F	F	T	Some	$$$	F	T	French	0–10	T
2	T	F	F	T	Full	$	F	F	Thai	30–60	F
3	F	T	F	F	Some	$	F	F	Burger	0–10	T
4	T	F	T	T	Full	$	F	F	Thai	10–30	T
$(\mathbf{x}^{(5)}, y^{(5)})$	T	F	T	F	Full	$$$	F	T	French	>60	F
6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
7	F	T	F	F	None	$	T	F	Burger	0–10	F
8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
9	F	T	T	F	Full	$	T	F	Burger	>60	F
10	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
11	F	F	F	F	None	$	F	F	Thai	0–10	F
12	T	T	T	T	Full	$	F	F	Burger	30–60	T

Each raw is an example $(\mathbf{x}^{(i)}, y^{(i)})$, where the output $y^{(i)}$ is true (T) or false (F).

Decision trees

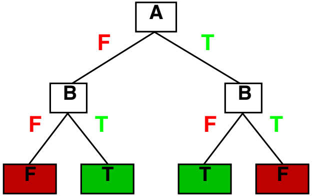
One possible representation for hypotheses
E.g., here is the "true" tree for deciding whether to wait:



Expressiveness

Decision trees can express any function of the input attributes.
E.g., for Boolean functions, truth table row → path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless f nondeterministic in $\mathbf{x}$) but it probably won't generalize to new examples

We prefer to find more **compact** decision trees

Expressiveness cont'd

How many distinct decision trees with n Boolean attributes??

$=$ number of Boolean functions

$=$ number of distinct truth tables with 2^n rows $= 2^{2^n}$

E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 ($\approx 10^{19}$) trees

With 10 Boolean attributes there are about 10^{308} trees

More expressive hypothesis space

– increases chance that target function can be expressed :)

– increases number of hypotheses consistent w/ training set

$\Longrightarrow$ may get worse predictions :(

Decision tree learning: Idea

Aim: find a small tree consistent with the training examples

Idea: (recursively) choose "most significant" attribute as root of (sub)tree:

- Start with the whole training set and an empty decision tree
- Pick a feature that gives the best split
- Split on that feature and recurse on sub-partitions

Decision tree learning algorithm

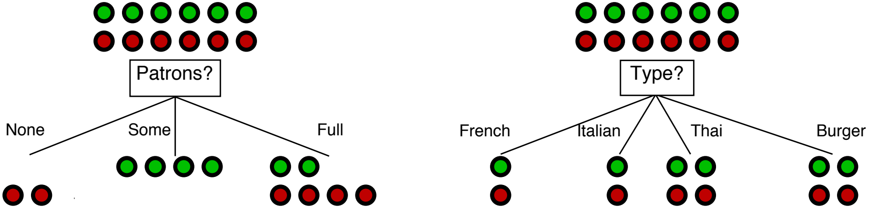
```
function LEARN-DECISION-TREE(examples, attributes, parent_examples) returns a tree

  if examples is empty then return PLURALITY-VALUE(parent_examples)
  else if all examples have the same classification then return the classification
  else if attributes is empty then return PLURALITY-VALUE(examples)
  else
     A ← argmax_{a ∈ attributes} IMPORTANCE(a, examples)
     tree ← a new decision tree with root test A
     for each value v of A do
        exs ← {e : e ∈ examples and e.A = v}
        subtree ← LEARN-DECISION-TREE(exs, attributes − A, examples)
        add a branch to tree with label (A = v) and subtree subtree
     return tree
```

The function IMPORTANCE measures the importance of attributes (as explained next). The PLURALITY-VALUE function selects the most common output value among a set of examples, breaking ties randomly.

Choosing attribute tests

Idea: a good (=**important**) attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



Patrons? is a better choice – gives **information** about the classification

Information gain

- Information answers questions
- The more clueless we are about the answer initially, the more information is contained in the answer
- 1 bit $=$ answer to Boolean question with prior $\langle 0.5, 0.5 \rangle$
- Information in an answer when prior is $\langle P_1, \ldots, P_n \rangle$ is

$$H(\langle P_1, \ldots, P_n \rangle) = \sum_{i=1}^{n} -P_i \log_2 P_i$$

(also called entropy of the prior)

Information gain, cont'd

Suppose we have p positive and n negative examples at the root

$\implies H(\langle p/(p+n), n/(p+n)\rangle)$ bits needed to classify a new example

E.g., for 12 restaurant examples, $p = n = 6$ so we need 1 bit

An attribute splits the examples E into subsets E_i, each of which (we hope) needs less information to complete the classification

Let E_i have p_i positive and n_i negative examples

$\implies H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$ bits needed to classify a new example

$\implies$ **expected** number of bits per example over all branches is

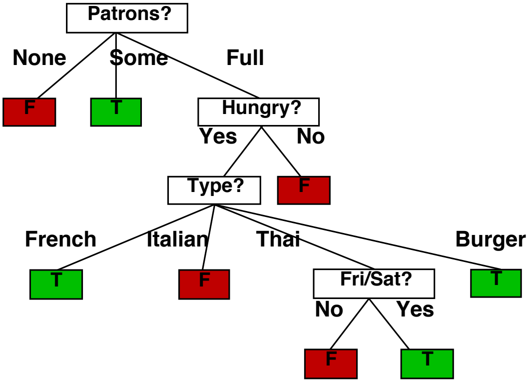
$$\sum_i \frac{p_i + n_i}{p+n} H\left(\left\langle \frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i} \right\rangle\right)$$

For $Patrons?$, this is 0.459 bits, for $Type$ this is (still) 1 bit

$\implies$ choose the attribute that minimizes the remaining information needed

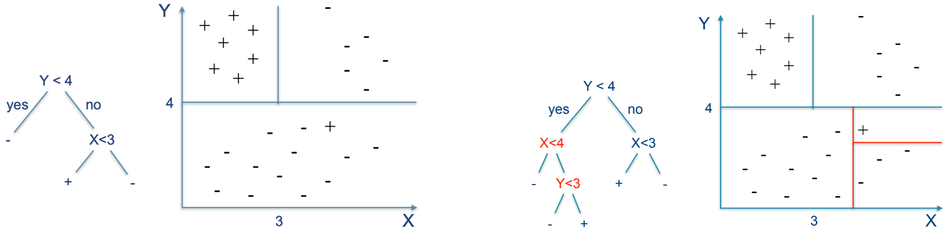
Information gain cont'd

Decision tree learned from the 12 examples:



Substantially simpler than "true" tree — a more complex hypothesis isn't justified by small amount of data

Some considerations



Left: a small tree fits the training data almost perfectly. It can be grown to fit perfectly (right), but a relatively large area to the right will then be predicted positive, while the data contains very little evidence for this.

Next lesson

- Perceptron
- Neural networks