Selected classification techniques

T Padma Ragaleena

National Institute of Science Education and Research
Bhubaneswar

6 July 2021

The problem of classification

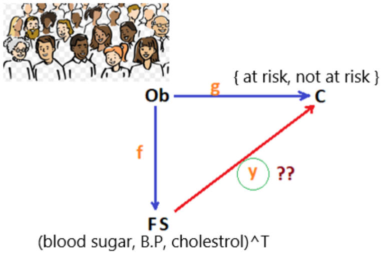
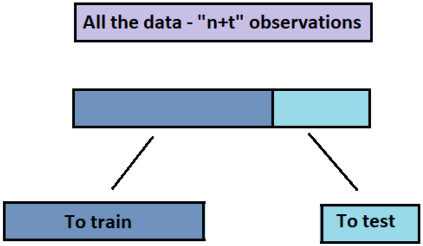


Figure: Classification problem

- Ob = set of objects that need to classified into k different classes.
- FS = Feature space = All possible d-dimensional vector of features extracted for each object.
- C = set of all possible classes i.e. $C = \{C_1, C_2, \cdots, C_k\}$
- Function g is the ideal classifier - unknown.
- Function f outputs or extracts features for each object in Ob.
- We wish to find an approximation of function g, denoted by y, based on the **data we have**.

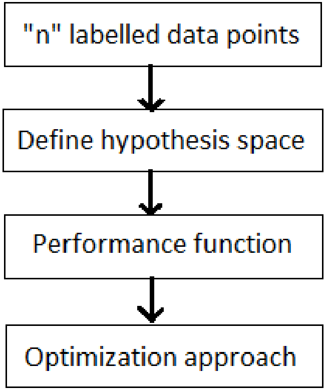
Testing and Training data



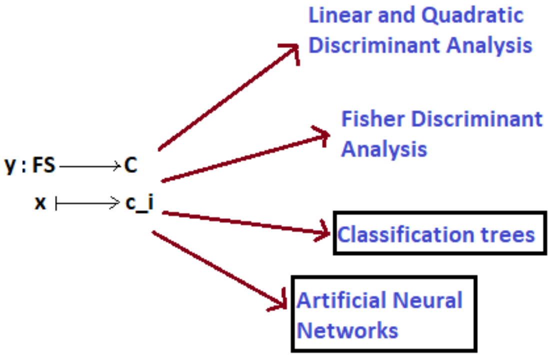
- We are given a set of "$n + t$" labelled data points i.e. data points for which the class labels are already known.
- We divide the complete data sets. One part is used to construct the model while the other part is used to test the correctness of the constructed model.

How is *y* constructed?

- The function y is estimated based on a set of data observations whose class labels are already known.
- Hypothesis space (H) is the set of all functions in which we search for the "best" functional form of y. Eg. set of continuous functions.
- A performance function is defined to quantify how well a function $h \in H$ when a new set of observations are given as inputs.
- Once the performance function if defined, an optimization technique to optimize the performance function has to be chosen.



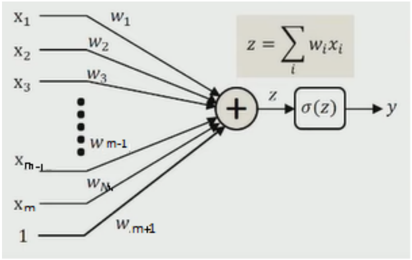
Different classification techniques


y : FS ⟶ C
x ⟼ c_i
Linear and Quadratic Discriminant Analysis
Fisher Discriminant Analysis
Classification trees
Artificial Neural Networks

Artificial neural networks

- Artificial neural networks(ANNs) were first introduced to mathematically model how neurons work in the human brain.
- Although ANNs were initially designed to model how the human brain works, these models were treated more and more abstractly over time.
- Modern day ANNs are not limited to understanding the human brain alone.
- Artificial neural networks, as the name suggests, is a **network of individual neurons**.
- Following slides explain what these individual neurons are and how they are connected to one another.
- neuron - network - network funtion -how to estimate network classifier?

Basic unit of ANN : Neuron



- Assume "*n*" observations each of dimension *m*.
- A neuron takes in a m-dimensional observation $(x_1, \cdots, x_m)^T$ as an input.
- A weight w_i is associated to each input x_i. Both weights and inputs are real valued.
- An additional weight w_{m+1} corresponding to the input $x_{m+1} = 1$ is added as **bias**.
- purpose of bias will be explained soon.
- σ is called the **activation function**.

Activation function

- Different choices of activation function give different properties to the neuron and the neural network.
- σ is usually a function from $\mathbb{R}$ to a bounded interval like $[0, 1]$ or $[-1, 1]$.
- Two of the many possible activation functions are:
  - Threshold activation

$$\sigma(x) = \mathcal{I}_{x \geq 0}$$

  - Sigmoid activation

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- ANNs which use the threshold activation are discussed first. After that, ANNs which use sigmoid activation are discussed.

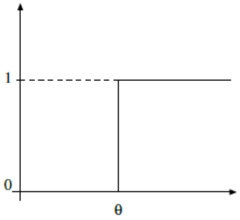


Figure: Threshold activation

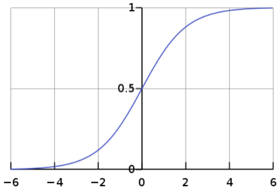
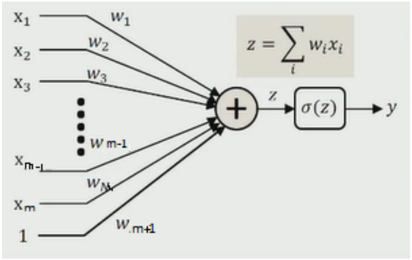


Figure: Sigmoid activation

Basic unit of a perceptron

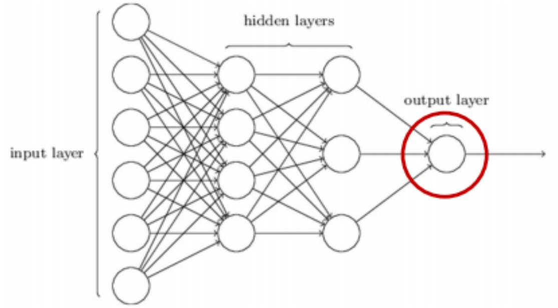


$y = 1$ iff $\sum_{i=1}^{m} w_i x_i + w_{m+1} \geq 0$
$y = 1$ iff $\sum_{i=1}^{m} w_i x_i \geq -w_{m+1}$
w_{m+1} is the bias term as it is the minimum sum we expect the weighted sum to have.

- A neuron (as defined earlier) with threshold activation function is the basic building unit of an ANN called **perceptron**.
- The inputs and weights are real but the output is binary i.e. $y = 0$ or 1. A single peceptron unit can only do binary classification.
- The neuron fires or outputs 1 iff the weighted sum of inputs is more than or equal to 0.
- If the bias term was not present, the neuron would fire iff the weighted sum of inputs exceeded $-w_i$. Bias is the minimum amount we want the weighted input to have to let the neuron fire.
- We add bias as one of the inputs for convenience during calculations.

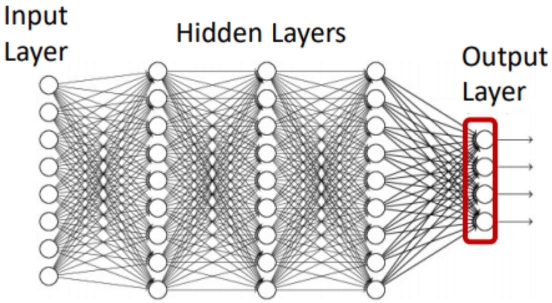
Architecture of a perceptron

- We only consider a special type of ANNs - feed-forward multi-layer perceptrons.
- This architecture is generally used for binary classification.
- In this type of ANN, the neurons are organized in layers. The outputs from the neurons of one layer act as inputs for the neurons in the next layer.
- The input layer is the layer that receives the presented input data. The output layer is the layer that supplies the modeled outputs. All other layers are denoted by hidden layers.



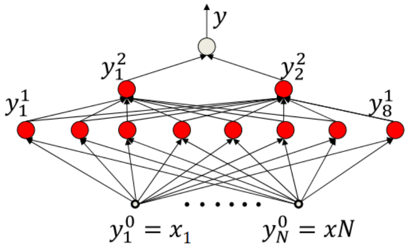
Architecture of a perceptron

- Feed-forward multi-layer perceptrons can have more than two outputs as well.
- Used for multi-class classification.
- Eg. $y = (1, 0, 0)$ stands for cat, $y = (0, 1, 0)$ stands for dog while $y = (0, 0, 1)$ stands for goat.



Every perceptron corresponds to a function

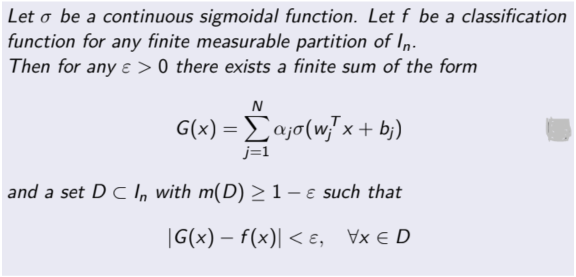
- Going along a series of nodes is equivalent to function composition.
- The outputs of one layer act as inputs for the next layer.
- Observe that the function a perceptron represents will not be differentiable if the activation function being use is the threshold function.
- Therefore, using a sigmoid activation is useful because it makes the over network correspond to a differentiable function - enables us to use different optimization techniques.



$$y_j^k = \sigma\left(\sum_i w_{i,j}^{k-1} y_i^{k-1}\right)$$

Universal approximation theorem

According to this theorem any classification function can be approximated by a neural network with arbitrary precision. However, this is only an existential proof.



Let σ be a continuous sigmoidal function. Let f be a classification function for any finite measurable partition of I_n.
Then for any $\varepsilon > 0$ there exists a finite sum of the form

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(w_j^T x + b_j)$$

and a set $D \subset I_n$ with $m(D) \geq 1 - \varepsilon$ such that

$$|G(x) - f(x)| < \varepsilon, \quad \forall x \in D$$

Performance measure of a perceptron

- $d(\tilde{x})$ = true classifier
- $f(\tilde{x}; \tilde{w})$ = perceptron function.
- Assume that the network architecture which can approximate d is already given to us.

Then our aim is to find appropriate weights and biases for such that the expected error of information lost in approximating d with the function f is minimized. In other words, we have to minimize the following,

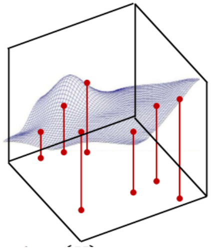
$$\min_{\tilde{w}} \mathbb{E}[\mathcal{D}(d(\tilde{X})||f(\tilde{X}))]$$

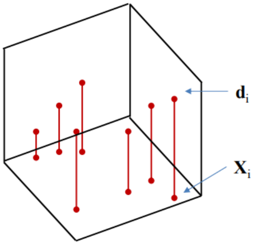
$\mathcal{D}(.||.)$ stands for KL-divergence:

$$\mathcal{D}(p(\tilde{x})||q(\tilde{x})) = \sum_{x \in \mathcal{X}} p(\tilde{x}) \ln\left(\frac{p(\tilde{x})}{q(\tilde{x})}\right)$$

Performance measure of a perceptron

- The function $d(\tilde{x})$ is almost always unknown to us. This is dealt with by sampling input and output pairs for the function $d(\tilde{x})$.
- Using sigmoid activation makes the network function differentiable $\implies$ gradient descent algorithm can be used.
- Sigmoid enables us to quantify change in the output for small changes in weights because the derivative of sigmoid is never zero. This is not possible for ANN with threshold activation.





Empirical risk

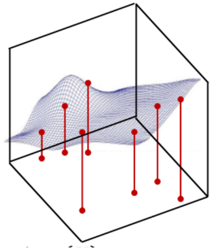
The function we actually want to minimize with respect to $\tilde{w}$:

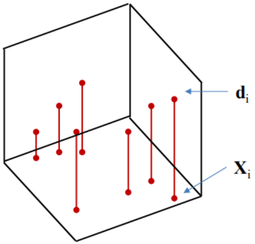
$$h(\tilde{w}) = \mathbb{E}[\mathcal{D}(d(\tilde{X})||f(\tilde{X}))] \tag{1}$$

However, since d is unknown, we minimize the following estimate of h:

$$\widehat{h}(\tilde{w}) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{D}(d_i||f(\tilde{x}_i)) \tag{2}$$

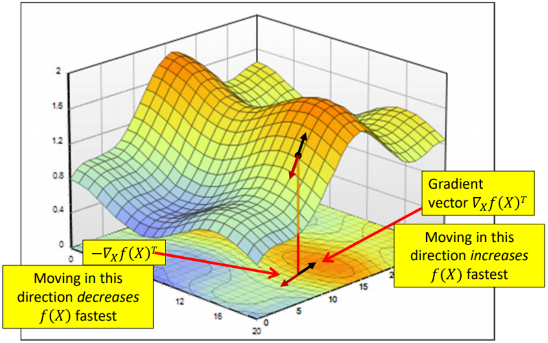
where d_i is the true class corresponding to input $\tilde{x}_i$.





Gradient descent

The empirical risk is minimized with respect to $\tilde{w}$ using the gradient descent algorithm to find the appropriate $\tilde{w}$.



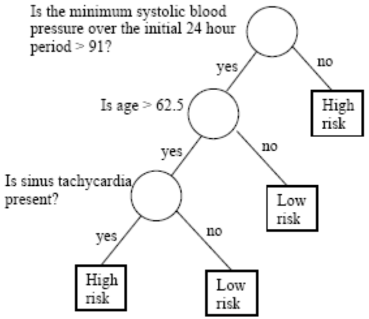
Problems with the method

- Gradient descent doesn't always take us to the global minima. Depends on where we start.
- It is common for the emperical risk function to have saddle points. Using gradient descent algorithm can imply that we might eventually get stuck at saddle points.
- How do we interpret the outputs?
- Adding a new training data points doesn't change the perceptron network we derive using gradient descent.

What is a classification tree?

Let F denote the feature space while $C = \{c_1, c_2, \cdots, c_k\}$ denotes the set of all possible classes. Then a classifier T is called said to be a **classification tree** for F and C if the following conditions hold:

- T is a finite tree i.e. no. of nodes is finite.
- T has one unique root node.
- T is a connected acyclic graph.
- Every non-terminal node is split in a certain way with respect to only one variable V.
- Every leaf node is assigned to a class label.


Is the minimum systolic blood pressure over the initial 24 hour period > 91?
yes
no
Is age > 62.5
High risk
yes
no
Is sinus tachycardia present?
Low risk
yes
no
High risk
Low risk

What is a classification tree?

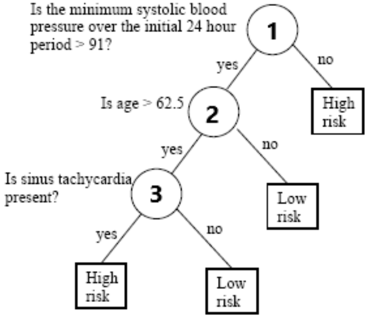
The example to the right has three variables on which y is defined. Let these three variables be denoted, in short, as:

- B - minimum systolic blood pressure
- A - age of the patient
- S - 1 if sinus is present, otherwise 0

Let all possible values in input space be:

$$Sp(B \times A \times S) = [0, 200] \times [1, 100] \times \{0, 1\}$$

where Sp denotes support of variables.


Is the minimum systolic blood pressure over the initial 24 hour period > 91?
1
yes
no
High risk
Is age > 62.5
2
yes
no
Low risk
Is sinus tachycardia present?
3
yes
no
High risk
Low risk

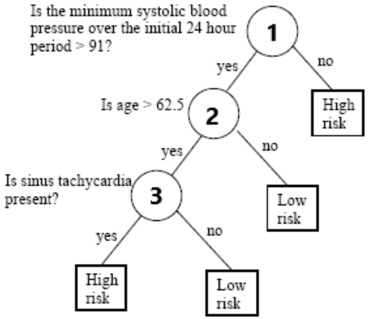
What is a classification tree?

Every node of a classification tree is associated with a subset of training observations.

- D = set of n training obs.

  $$D = \{(\tilde{x}_1, c_1), (\tilde{x}_2, c_2) \cdots, (\tilde{x}_n, c_n)\}$$

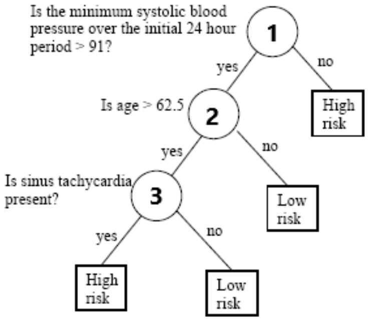
- $D(\tau) \subset D$ = subset of D that is associated to node τ.
- $D(1) = D$
- $D(2) = \{(\tilde{x}, c) | x_B > 91\}$
- $D(3) = \{(\tilde{x}, c) | x_B > 91 \& A > 62.5\}$
- Both terminal and non-terminal nodes are associated with a subset of training observations.


Is the minimum systolic blood pressure over the initial 24 hour period > 91?
1
yes
no
High risk
Is age > 62.5
2
yes
no
Low risk
Is sinus tachycardia present?
3
yes
no
High risk
Low risk

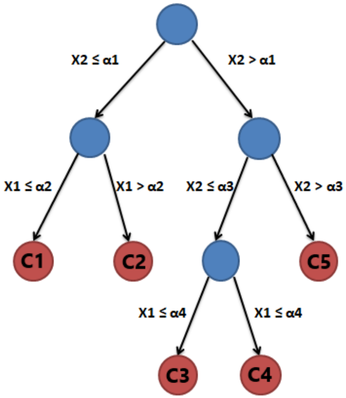
Classification tree

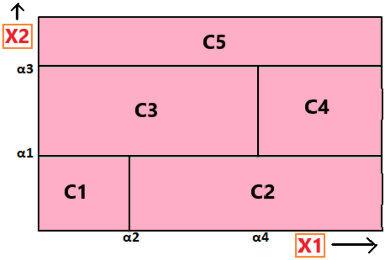
- Each terminal node is also associated with a subset of training observations. $Sp(B \times A \times S) = [0, 200] \times [1, 100] \times \{0, 1\}$
- $LR_1 = (91, 200] \times [1, 62.5] \times \{0, 1\}$
- $LR_2 = (91, 200] \times (62.5, 100] \times \{0\}$
- $HR_1 = [0, 91] \times [1, 100] \times \{0, 1\}$
- $HR_2 == (91, 200] \times (62.5, 100] \times \{1\}$
- input space = $LR_1 \cup LR_2 \cup HR_1 \cup HR_2$

Terminals nodes partition the input space.


Is the minimum systolic blood pressure over the initial 24 hour period > 91?
1
yes
no
High risk
Is age > 62.5
2
yes
no
Low risk
Is sinus tachycardia present?
3
yes
no
High risk
Low risk

Terminals nodes partition the input space.





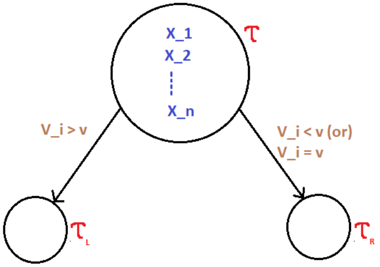
- Hypothesis space consists of all *m* dimensional functions whose input space is divided by hyperplanes of the type $x_i = c$
- Every such function corresponds to a classification tree and vice-versa.

Questions we will address

- How do we determine which variable to use to split a given node i.e. what is the best split at a given node?
- When do we stop splitting?
- How do we find the “best” tree classifier that can be built using *D*.

Choosing the best split

Each node can be split with repect to any one of the *m* variables. How do we quantify the performance of a given split at a given node? - **using Impurity functions**.

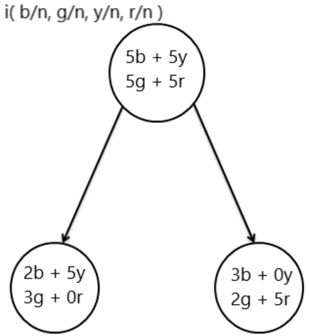


- Impurity functions can help quantify how well the resulting daughter nodes classify in comparison to that of the parent node.

Two possible impurity functions:

- Impurity function that can be used to quantify how homogeneous or pure the daughter nodes are in comparison to parent nodes.
- Impurity functions that can be used to quantify gain in information caused by partitioning training set based on a given variable.

Choosing the best split



- Impurity functions i are real valued functions on variables $p_1, p_2, \cdots, p_k$ where p_i denotes the proportion of observations (associated with the given node) that belong to class c_i.
- For example, $i_p = i_p(0.25, 0.25, 0.25, 0.25)$ while $i_L(2/10, 3/10, 5/10, 0)$

Reduction in impurity (Δi):

$$i_p(0.25, 0.25, 0.25, 0.25) - \frac{10}{20} i_L\left(\frac{2}{10}, \frac{3}{10}, \frac{5}{10}, 0\right) - \frac{10}{20} i_R\left(\frac{3}{10}, \frac{2}{10}, 0, \frac{5}{10}\right)$$

Impurity functions

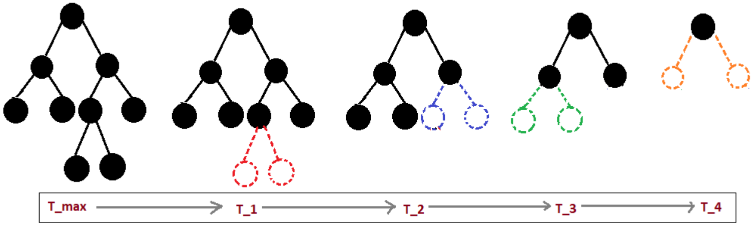
Possible impurity functions:

- $i(p_1, \cdots, p_k) = 1 - \max_{i=1}^{k} p_i$
  Δi quantifies how homogeneous daughter nodes are in comparision to parent node.
- $i(p_1, \cdots, p_k) = -\sum_{i=1}^{k} p_i \log(p_i)$
  Δi is equivalent to calculating information gain. Therefore, it quantifies gain in information or reduction in uncertainity when a node is split based on a given variable.

Best split at a given node is the split which corresponds to maximum reduction in impurity

How do find the best tree classifier?

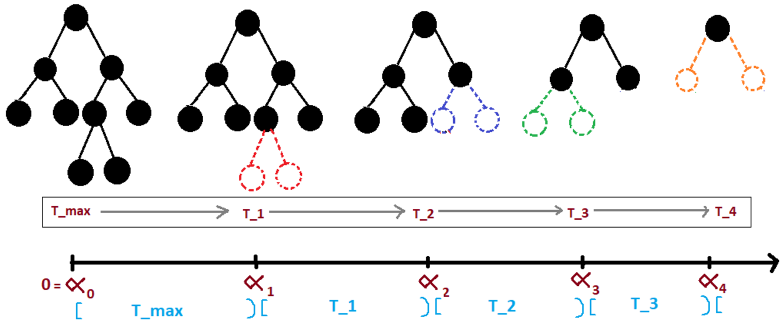
- Grow the tree until the nodes cannot be split further.
- Such a tree, whose nodes can't be split further, is said to be **saturated**. It is denoted by T_{max}.
- In order to find the best tree, we **prune** T_{max} upwards i.e. we sequentially chop off "unecessary" daughter nodes.
- Intuitively, the best tree is neither split too much nor split too less.



How to prune?

$$R_\alpha(T) = R(T) + \alpha \times \#\text{terminal nodes}$$

- $R(T)$ is the number of observations miss-classified by the tree.
- As α value increases, the tree minimizing R_α has fewer splittings.



Pros and Cons of Classification trees

Pros:

- Non-parametric method i.e. no distribution assumptions
- No scaling of input required
- Easy to interpret and implement.

Cons:

- Since the method doesn't make any assumptions about the data, the tree tends to overfit the structure present in the data.
- Small changes in input data can greatly change the output.

References

- Alan J. Izenman. *Modern Multivariate Statistical Techniques*. Springer-Verlag New York, 2008.
- Jiawei Han. *Kullback-Leibler Divergence* (Additional subsection of the book “Data Mining: Concepts and Techniques”, Morgan Kaufmann Publishers, 2012). CS412: An Introduction to Data Warehousing and Data Mining (Fall 2008 lecture notes), University of Illinois Urbana-Champaign: `http://hanj.cs.illinois.edu/cs412/bk3/KL-divergence.pdf`
- Linda Shapiro. *Information Gain* (Slides for chapter 4 of the book “Computer Vision”, Prentice-Hall, 2001). EE P 596: Machine Vision (Autumn 2019 lecture notes), University of Washington, Seattle: `https://homes.cs.washington.edu/~shapiro/EE596/notes/InfoGain.pdf`
- Bhiksha Raj and Rita Singh. *11-785 Introduction to Deep Learning* (Fall 2020 lecture videos and notes). Carnegie Mellon University, Pittsburgh: `https://deeplearning.cs.cmu.edu/F20/index.html`
- Mark J. Grover and Miguel Maldonado. *Pros and Cons of Decision Trees* (Lecture video from “Supervised Machine Learning: Classification” course). IBM Coursera MOOC: `https://www.coursera.org/lecture/supervised-learning-classification/pros-and-cons-of-decision-trees-lTqot`