EEE 551 Information Theory (Spring 2022)

A Bit of Machine Learning

What is machine learning?

Training a computer to accomplish some statistical task, based on samples from a (complex) distribution, without access to the distribution itself

Examples of statistical tasks that ML is good for:

- Classification (i.e. detection)
- Estimation
- Sampling (i.e. generation of fake data)

In this lecture, we'll cover two ML techniques with connections to information theory:

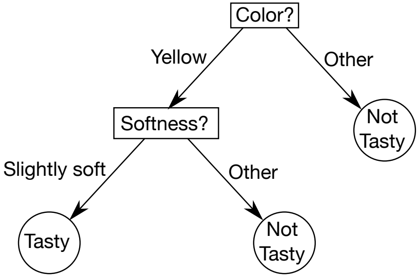
- **Decisions trees** — a classification technique
- **Generative adversarial networks** — a sampling technique

Decision Trees

- **Supervised classification:** We are given $(X_i, Y_i) \sim P_{XY}$ for $i = 1, \dots, m$, where $Y_i \in \{0, 1\}$, but P_{XY} is unknown
- **Goal**: Learn a function $d : \mathcal{X} \to \{0, 1\}$, such that, given a new sample $X \sim P_X$, $\hat{Y} = g(X)$ is a good estimate for Y
- A **decision tree** is a type of classifier that makes successive decisions along a tree
- **Example:** By observing a papaya, we want to guess whether it's tasty or not

  $X = (X_{\text{color}}, X_{\text{softness}})$, where
  - $X_{\text{color}} \in \{\text{green, yellow, orange}, \dots\}$
  - $X_{\text{softness}} \in \{\text{very hard, hard, slightly soft, mushy}\}$

  $Y \in \{\text{tasty, not tasty}\}$



- How to learn a decision tree from samples (X_i, Y_i)?

Attributes

- Assume that $X = (A_1, A_2, \ldots, A_d)$, where $A_j \in \{0, 1\}$ is a binary attribute
- We build the tree from the root to the leaves
- At each step, we choose the **most informative** attribute, and form two branches based on that attribute
- How to decide which attribute is most informative? Mutual information!
- That is, we choose attribute with index

$$\arg\max_j I(A_j; Y)$$

- However, we do not have the true distribution $p_{A_j,Y}$, we only have samples (a_{ji}, y_i)
- We can estimate the mutual information based on the samples, using the empirical distribution (i.e., the type)

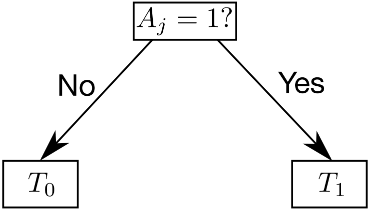
The ID3 Algorithm

Initialize: $S = \{1, \ldots, m\}$, $Q = \{1, \ldots, d\}$

We recursively call the following function:

ID3(S, Q), where $S \subset \{1, \ldots, m\}$, $Q \subset \{1, \ldots, d\}$

1. If all examples in $Y_i = 0$ for all $i \in S$, return a leaf 0
2. If all examples in $Y_i = 1$ for all $i \in S$, return a leaf 1
3. Let $j = \arg\max_{j \in Q} I(A_j; Y)$ where the mutual information is calculated from the empirical distribution of the samples $((a_{ji}, y_i) : i \in S)$
4. Let $T_0 =$ **ID3**$(\{i \in S : a_{ji} = 0\},\ Q \setminus \{j\})$
5. Let $T_1 =$ **ID3**$(\{i \in S : a_{ji} = 1\},\ Q \setminus \{j\})$
6. Return the tree:



Learning a Generative Model

- We are given $X_1, \ldots, X_m \sim P_X$, but P_X is unknown
- We want to learn a **generative model**: given $Z \sim P_Z$, where P_Z is a distribution that is easy to sample from (e.g. uniform, Gaussian), find a function g such that

  $$W = g(Z)$$

  and the distribution of W should be "close" to P_X
- **Example**: X_i are photos, and we want to generate synthetic photos
- How to measure the closeness of two distributions?

Jensen-Shannon Divergence

- The relative entropy $D(P_0 \| P_1)$ measures the distance between two distributions P_0, P_1, but it has the disadvantage that it is not symmetric
- One way to form a symmetric distance is the **Jensen-Shannon Divergence**:

$$J(P_0, P_1) = \frac{1}{2} D(P_0 \| \tfrac{1}{2}(P_0 + P_1)) + \frac{1}{2} D(P_1 \| \tfrac{1}{2}(P_0 + P_1))$$

- $J(P_0, P_1) = J(P_1, P_0)$
- $J(P_0, P_1) \geq 0$, with equality if and only if $P_0 = P_1$
- $J(P_0, P_1) = I(V; X)$, where $V \sim \text{Bern}(1/2)$ and

$$p(x|v) = \begin{cases} P_0(x), & v = 0, \\ P_1(x), & v = 1 \end{cases}$$

- $J(P_0, P_1) \leq 1$

Variational Representation of Jensen-Shannon Divergence

$$J(P_0, P_1) = \max_{d:\mathcal{X}\to[0,1]} \frac{1}{2}\mathbb{E}_{P_0}\left[\log(1-d(X))\right] + \frac{1}{2}\mathbb{E}_{P_1}\left[\log d(X)\right] + 1$$

Proof:

- The right-hand side is

$$\max_{d:\mathcal{X}\to[0,1]} \frac{1}{2}\sum_x P_0(x)\log(1-d(x)) + \frac{1}{2}\sum_x P_1(x)\log d(x) + 1$$

- To maximize over d, differentiate:

$$0 = \frac{\partial}{\partial d(x)}(\text{above}) = -\frac{P_0(x)}{2(1-d(x))} + \frac{P_1(x)}{2d(x)}$$

- This is solved by setting

$$d(x) = \frac{P_1(x)}{P_0(x)+P_1(x)}$$

- This gives

$$\begin{aligned}
&\frac{1}{2}\sum_x P_0(x)\log\frac{P_0(x)}{P_0(x)+P_1(x)} + \frac{1}{2}\sum_x P_1(x)\log\frac{P_1(x)}{P_0(x)+P_1(x)} + 1\\
&= \frac{1}{2}\sum_x P_0(x)\log\frac{P_0(x)}{(P_0(x)+P_1(x))/2} + \frac{1}{2}\sum_x P_1(x)\log\frac{P_1(x)}{(P_0(x)+P_1(x))/2}\\
&= \frac{1}{2}D(P_0\|\tfrac{1}{2}(P_0+P_1)) + \frac{1}{2}D(P_1\|\tfrac{1}{2}(P_0+P_1)) = J(P_0, P_1)
\end{aligned}$$

A min-max game

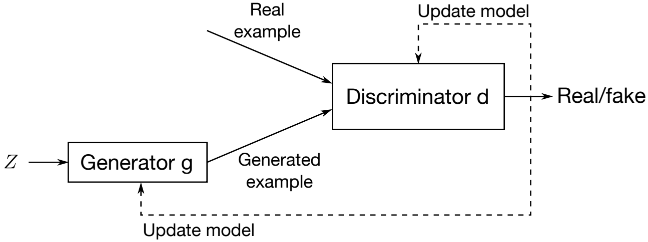
- Recall we want a function g where $W = g(Z)$ has the same distribution as X
- Consider the following:

$$
\begin{aligned}
&\min_g \max_{d:\mathcal{X}\to[0,1]} \mathbb{E}[\log(1-d(X))] + \mathbb{E}[\log d(g(Z))] \\
&= \min_g \max_{d:\mathcal{X}\to[0,1]} \mathbb{E}[\log(1-d(X))] + \mathbb{E}[\log d(W)] \\
&= \min_g \; 2\left(J(P_X, P_W) - 1\right)
\end{aligned}
$$

- This will choose the function g that minimizes $J(P_X, P_Y)$
- In practice, we only have **samples** of X and Z, so we approximate by

$$
\min_g \max_{d:\mathcal{X}\to[0,1]} \frac{1}{m}\sum_{i=1}^{m} \log(1-d(x_i)) + \frac{1}{m}\sum_{i=1}^{m} \log d(g(z_i))
$$

Generative Adversarial Networks (GANs)



The functions g and d are neural networks, and we alternatively minimize/maximize g and d using gradient descent

$$\min_{g} \max_{d:\mathcal{X}\to[0,1]} \frac{1}{m}\sum_{i=1}^{m} \log(1 - d(x_i)) + \frac{1}{m}\sum_{i=1}^{m} \log d(g(z_i))$$

Source: Karras et al, ICLR 2018