An Introduction to Information Theory with Applications to Machine Learning

Cade Reinberger

RIT Computer Science House

September 8, 2020

Outline

α-weighted coins

We begin with a motivating example

- Let $\alpha \in \mathbb{R}$ be a real paramater with $0 \leq \alpha \leq 1$

α-weighted coins

We begin with a motivating example

- Let $\alpha \in \mathbb{R}$ be a real paramater with $0 \leq \alpha \leq 1$
- We define an α-weighted coin, or simply an α-coin, to be a coin which lands heads (H) with probability α and tails (T) with probability $1 - \alpha$

α-weighted coins

We begin with a motivating example

- Let $\alpha \in \mathbb{R}$ be a real paramater with $0 \leq \alpha \leq 1$
- We define an α-weighted coin, or simply an α-coin, to be a coin which lands heads (H) with probability α and tails (T) with probability $1 - \alpha$
- We want to encode the result of this coin toss as information

Binary Trees

- Our goal is to encode the outcome of this coin toss as a sequence of 0s and 1s with optimal efficiency

Binary Trees

- Our goal is to encode the outcome of this coin toss as a sequence of 0s and 1s with optimal efficiency
- We use a binary tree to do this encoding.

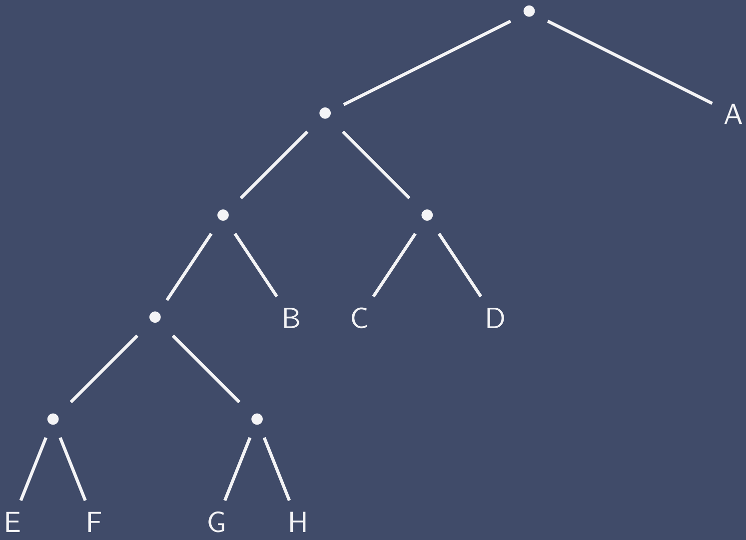
Binary Trees

- Our goal is to encode the outcome of this coin toss as a sequence of 0s and 1s with optimal efficiency
- We use a binary tree to do this encoding.
- a *binary tree* is a recursive graph-like data structure with parent nodes and leaves. Each parent node contains two children, starting with one node known as the *root*, and each leaf contains no children but a value

Binary Trees

```
class binary_tree_node:
    def __init__(self, is_leaf, left, right, val):
        if is_leaf:
            self.is_leaf = True
            self.left_child = None
            self.right_child = None
            self.val = val
        else:
            self.is_leaf = False
            self.left_child = left
            self.right_child = right
            self.val = None
```

Binary Trees



Binary Trees

- Binary Trees are not only natural to express this encoding, but are more or less a fully accurate representation of the problem (that is, encodings are equivalent to Binary Trees).

Binary Trees

- Binary Trees are not only natural to express this encoding, but are more or less a fully accurate representation of the problem (that is, encodings are equivalent to Binary Trees).
- "Normal" binary trees in most of CS (such as BSTs), can contain a value at *all* nodes, not just leaves. Why isn't that true for encoding binary information?

Binary Trees

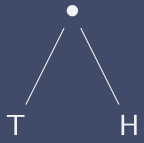
- Binary Trees are not only natural to express this encoding, but are more or less a fully accurate representation of the problem (that is, encodings are equivalent to Binary Trees).
- "Normal" binary trees in most of CS (such as BSTs), can contain a value at *all* nodes, not just leaves. Why isn't that true for encoding binary information?
- Because to read that as an encoding, the encoder has to recognize at the end of the encoding string that it has ended. Thus, implicitly, saying some string has ended adds an extra termination bit of information at the end.

Encoding Coin Flips

- For our goal of encoding the result of an α-coin flip using binary trees, the answer is very boring.

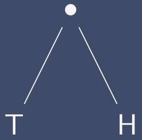
Encoding Coin Flips

- For our goal of encoding the result of an α-coin flip using binary trees, the answer is very boring.
- For $\alpha \neq 0, 1$, there's one obvious best encoding



Encoding Coin Flips

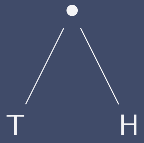
- For our goal of encoding the result of an α-coin flip using binary trees, the answer is very boring.
- For $\alpha \neq 0, 1$, there's one obvious best encoding



- Good night everybody

Encoding Coin Flips

- For our goal of encoding the result of an α-coin flip using binary trees, the answer is very boring.
- For $\alpha \neq 0, 1$, there's one obvious best encoding



- What went wrong?

Encoding Coin Flips

- Well, imagine that $\alpha = 1 - \varepsilon$ with ε small

Encoding Coin Flips

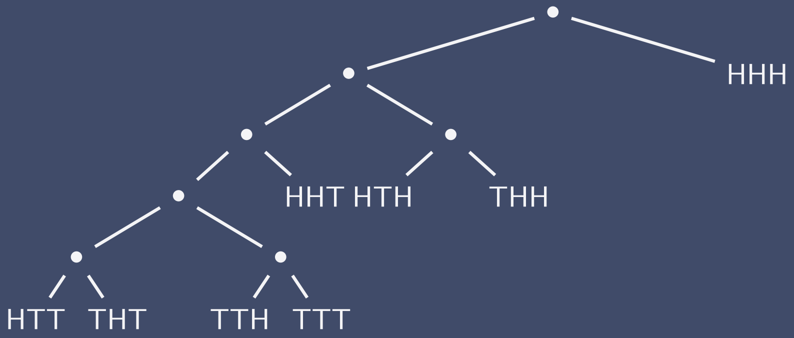
- Well, imagine that $\alpha = 1 - \varepsilon$ with ε small
- Maybe if you toss the coin once, T is not avoidable since ε is not neglible, but say you toss the coin 10 times.
- ε^{10} will be pretty well negligible, so there ought to be at least effectively one less outcome ($TTTTTTTTTT$) than in the case of $\alpha = \frac{1}{2}$, when all strings are equally likely.

Encoding Coin Flips

- Well, imagine that $\alpha = 1 - \varepsilon$ with ε small
- Maybe if you toss the coin once, T is not avoidable since ε is not neglible, but say you toss the coin 10 times.
- ε^{10} will be pretty well negligible, so there ought to be at least effectively one less outcome $(TTTTTTTTTT)$ than in the case of $\alpha = \frac{1}{2}$, when all strings are equally likely.
- Thus, we might expect that we can somehow do better encoding 10 coin tosses than 1 coin toss.

Encoding Coin Flips

- Let's imagine encoding the result of 3 .9-coin flips in a binary tree, such that we minimize the expected length of the message.



Encoding Coin Flips

Result	Encoding	Length	Probability	%
HHH	1	1	(.9)(.9)(.9)	72.9%
HHT	001	3	(.9)(.9)(.1)	8.1%
HTH	010	3	(.9)(.1)(.9)	8.1%
THH	011	3	(.1)(.9)(.9)	8.1%
HTT	00000	5	(.9)(.1)(.1)	0.9%
THT	00001	5	(.1)(.9)(.1)	0.9%
TTH	00010	5	(.1)(.1)(.9)	0.9%
TTT	00011	5	(.1)(.1)(.1)	0.1%

Encoding Coin Flips

Result	Encoding	Length	Probability	%
HHH	1	1	(.9)(.9)(.9)	72.9%
HHT	001	3	(.9)(.9)(.1)	8.1%
HTH	010	3	(.9)(.1)(.9)	8.1%
THH	011	3	(.1)(.9)(.9)	8.1%
HTT	00000	5	(.9)(.1)(.1)	0.9%
THT	00001	5	(.1)(.9)(.1)	0.9%
TTH	00010	5	(.1)(.1)(.9)	0.9%
TTT	00011	5	(.1)(.1)(.1)	0.1%

- So our expected length is $(.729)(1) + (3)(.081)(3) + (3)(.009)(5) + (.001)(5) = \mathbf{1.598}$

Encoding Coin Flips

- So we can encode 3 idependent trials of a .9-coin with less than 3 bits, on average.

Encoding Coin Flips

- So we can encode 3 idependent trials of a .9-coin with less than 3 bits, on average.

$$\frac{1.598}{3} = .532667$$

Encoding Coin Flips

- So we can encode 3 idependent trials of a .9-coin with less than 3 bits, on average.

$$\frac{1.598}{3} = .532667$$

- Thus, there's not really a full bit of information in tossing a coin with $\alpha = .9$

Information Entropy

- As I encode n independent instances of an α-coin, the minimum average length is a non-increasing function of n, so what happens to it in the long term

Information Entropy

- As I encode n independent instances of an α-coin, the minimum average length is a non-increasing function of n, so what happens to it in the long term
- Somewhat miraculously, we have that the best average length approaches an actual limit.

Information Entropy

Theorem (Shannon's Source Coding Theorem, Bernoulli Trial)

Let $\mathbb{T}_n$ denote the set of all binary trees with 2^n leaves, let $C_n \in \{H, T\}^n$ be a random variable, the result of n iid α-coin tosses, and let $\text{len}_\tau(C_n)$ for a binary tree $\tau \in \mathbb{T}_n$ denote the length of encoding C_n within the binary tree τ. Then

$$\lim_{n\to\infty} \min_{\tau\in\mathbb{T}_n} \frac{\mathbb{E}[\text{len}_\tau(C_n)]}{n} = H$$

for some real constant $H \geq 0$

Information Entropy

Proof. The minimum expected length is monotone decreasing and bounded below by 0, the result follows trivially from the monotone convergence theorem

Information Entropy

- This limit H is called the *information entropy* of the coin toss, in units of bits (or Shannons).

Computing Information Entropy

- So why is Shannon famous?

Computing Information Entropy

- So why is Shannon famous?
- We can compute the value of H.

Computing Information Entropy

- So why is Shannon famous?
- We can compute the value of H.
- Shannon's original computation was rather difficult, and was specifically considering information in Markov Chains. In fact, if you're careful about the logarithm as having negative concavity and so-forth, you can simply prove that you can get arbitrarily close to H and never do better than H.

Computing Information Entropy

- So why is Shannon famous?
- We can compute the value of H.
- Shannon's original computation was rather difficult, and was specifically considering information in Markov Chains. In fact, if you're careful about the logarithm as having negative concavity and so-forth, you can simply prove that you can get arbitrarily close to H and never do better than H.
- We're going to consider a slightly less rigorous derivation.

Computing Information Entropy

- Consider some binary tree $\tau \in \mathbb{T}_n$ encoding C_n.

Computing Information Entropy

- Consider some binary tree $\tau \in \mathbb{T}_n$ encoding C_n.
- We claim first that we can determine the expected length of this encoding by simply considering the average lengths of H and T, and weighting appropriately.

Computing Information Entropy

- Consider some binary tree $\tau \in \mathbb{T}_n$ encoding C_n.
- We claim first that we can determine the expected length of this encoding by simply considering the average lengths of H and T, and weighting appropriately.
- We could take this as something of a definition of "average" in a suitable sense, but it's not so tough to see that this is just the arithmetic mean of the length of all instances of H in the encoding in a technical sense.

Computing Information Entropy

- Consider some binary tree $\tau \in \mathbb{T}_n$ encoding C_n.
- We claim first that we can determine the expected length of this encoding by simply considering the average lengths of H and T, and weighting appropriately.
- We could take this as something of a definition of "average" in a suitable sense, but it's not so tough to see that this is just the arithmetic mean of the length of all instances of H in the encoding in a technical sense.

Computing Information Entropy

- So, in the asymptotic, we can ditch the trees themselves and think in terms of *average lengths*.

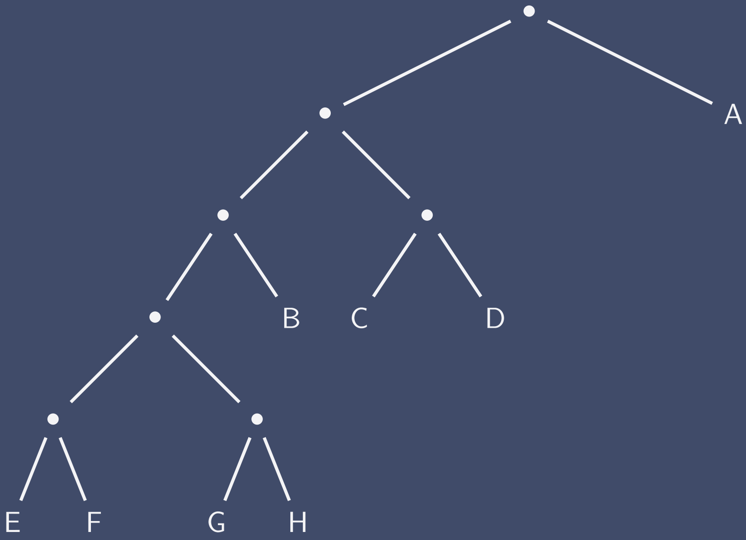
Computing Information Entropy

- So, in the asymptotic, we can ditch the trees themselves and think in terms of *average lengths*.
- Besides the fact this length isn't technically defined, we also haven't shown that we can generate binary trees that create average lengths (pointwise) convergent, to some set of averages.

Computing Information Entropy

- So, in the asymptotic, we can ditch the trees themselves and think in terms of *average lengths*.
- Besides the fact this length isn't technically defined, we also haven't shown that we can generate binary trees that create average lengths (pointwise) convergent, to some set of averages.
- Surely not *any* set of lengths will do. Which ones will?

Computing Information Entropy



Computing Information Entropy

- We want to define some invariant, to understand the lengths of this tree.

Computing Information Entropy

- We want to define some invariant, to understand the lengths of this tree.
- An easy invariant would be a sum of some function of depth.

Computing Information Entropy

- We want to define some invariant, to understand the lengths of this tree.
- An easy invariant would be a sum of some function of depth.
- Such a sum would necessarily be invariant under bifurcation of a node, and thus if it's just a function of depth must half.

Computing Information Entropy

- We want to define some invariant, to understand the lengths of this tree.
- An easy invariant would be a sum of some function of depth.
- Such a sum would necessarily be invariant under bifurcation of a node, and thus if it's just a function of depth must half.
- There an infinite number of such invariant you could define based on the constant of proportionality, but the simplest is to sum simply $2^{-(\text{depth})}$

Computing Information Entropy

- We want to define some invariant, to understand the lengths of this tree.
- An easy invariant would be a sum of some function of depth.
- Such a sum would necessarily be invariant under bifurcation of a node, and thus if it's just a function of depth must half.
- There an infinite number of such invariant you could define based on the constant of proportionality, but the simplest is to sum simply $2^{-(\text{depth})}$ And considering the simplest binary tree, one with just a single node of depth zero, this sum may be one.

Computing Information Entropy

Theorem (Kraft-McMillan Inequality, Full Binary Trees)

Some tuple of number $(\ell_1, \ell_2, \ldots, \ell_n)$ *can form the depth of nodes on some valid binary tree* τ *if and only if*

$$\sum_{k=1}^{n} 2^{-\ell_k} = 1$$

Computing Information Entropy

Proof (Necessity). We use strong induction on the maximum depth. The sum is invariant under bifurcation, so if we "combine" a pair of leaves into a lead of depth 1 lower, that leaves the sum unchanged. Do that for all nodes of maximum depth, and the depth is reduced. The base case is the trivial tree which clearly holds.

Proof (Sufficiency). Simply consider the binary tree that always has children up to some fixed maximum depth, larger than the maximum of $\{\ell_k | k \in \mathbb{Z} \cap [1, n]\}$. Then simply keep going left up to depth ℓ_1, then go left as much as you can, only stopping if you already reach a terminal leaf to go right. This process will yield such a tree. (Exercise: prove it rigorously).

Computing Information Entropy

- This gives us a regime for computing information entropy for an α-coin!

Computing Information Entropy

- This gives us a regime for computing information entropy for an α-coin!
- Consider average depths ℓ_H and ℓ_T that are suitable for binary trees (in the sense of satisfying Kraft's inequality).

Computing Information Entropy

- This gives us a regime for computing information entropy for an α-coin!
- Consider average depths ℓ_H and ℓ_T that are suitable for binary trees (in the sense of satisfying Kraft's inequality).
- Find the minimum average length among all such ℓ_H and ℓ_T

Computing Information Entropy

- Thus we have

$$H = \min_{(\ell_H, \ell_T) \in \{(\ell_H, \ell_T) \,|\, 2^{-\ell_H} + 2^{-\ell_T} = 1\}} (\alpha \ell_H + (1 - \alpha)\ell_T)$$

Computing Information Entropy

- Thus we have

$$H = \min_{(\ell_H,\ell_T)\, \in\, \{(\ell_H,\ell_T) \mid\, 2^{-\ell_H}+2^{-\ell_T}=1\}} (\alpha\ell_H + (1-\alpha)\ell_T)$$

- Solving Kraft's inequality for ℓ_T gives us simply that

$$\ell_T = -\log_2\left(1 - 2^{-\ell_H}\right)$$

Computing Information Entropy

- Thus we have

$$H = \min_{(\ell_H, \ell_T) \in \{(\ell_H, \ell_T) \mid 2^{-\ell_H} + 2^{-\ell_T} = 1\}} (\alpha \ell_H + (1-\alpha)\ell_T)$$

- Solving Kraft's inequality for ℓ_T gives us simply that

$$\ell_T = -\log_2\left(1 - 2^{-\ell_H}\right)$$

- Thus our problem is reduced to the 1-dimensional minimization problem

$$\min_{\ell_H}\left(\alpha \ell_H - (1-\alpha)\log_2\left(1 - 2^{-\ell_H}\right)\right)$$

Computing Information Entropy

- We can do this with some elementary calculus

$$
\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}\ell_H}\left(\alpha\ell_H - (1-\alpha)\log_2\left(1 - 2^{-\ell_H}\right)\right) \\
&= \alpha - (1-\alpha)\frac{\mathrm{d}}{\mathrm{d}\ell_H}\log_2\left(1 - 2^{-\ell_H}\right) \\
&= \alpha - \frac{1-\alpha}{2^{\ell_H} - 1}
\end{aligned}
$$

Computing Information Entropy

- We can do this with some elementary calculus

$$\begin{aligned}&\frac{\mathrm{d}}{\mathrm{d}\ell_H}\left(\alpha\ell_H-(1-\alpha)\log_2\left(1-2^{-\ell_H}\right)\right)\\&=\alpha-(1-\alpha)\frac{\mathrm{d}}{\mathrm{d}\ell_H}\log_2\left(1-2^{-\ell_H}\right)\\&=\alpha-\frac{1-\alpha}{2^{\ell_H}-1}\end{aligned}$$

- Setting this equal to 0 we get

$$\begin{aligned}\ell_H&=-\log_2(\alpha)\\\ell_T&=-\log_2(1-\alpha)\end{aligned}$$

Computing Information Entropy

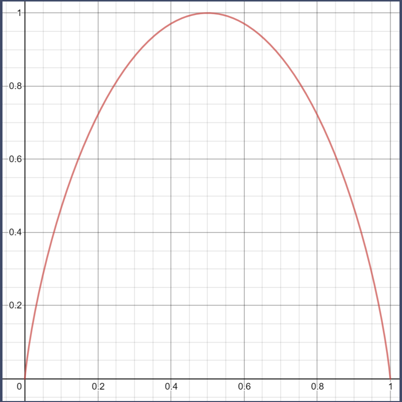
- Thus, the information entropy of an α coin is

$$\boxed{H = -\alpha \log_2(\alpha) - (1-\alpha)\log_2(1-\alpha)}$$

Computing Information Entropy

- Thus, the information entropy of an α coin is

$$\boxed{H = -\alpha \log_2(\alpha) - (1-\alpha)\log_2(1-\alpha)}$$



Computing Information Entropy

- Great. So we've done a bunch of calculation, and solved only this one very particular problem of entropy. :(

Computing Information Entropy

- Great. So we've done a bunch of calculation, and solved only this one very particular problem of entropy. :(
- But in fact, it is easy to adapt this method to any random variable with a finite image (set of possible values it could take).

Computing Information Entropy

- Great. So we've done a bunch of calculation, and solved only this one very particular problem of entropy. :(
- But in fact, it is easy to adapt this method to any random variable with a finite image (set of possible values it could take).
- Simply assign all values the RV could take with an average length , so that all the lengths satisfy Kraft's inequality, and then find the minimum expected length over all such values.

Computing Information Entropy

- Let X be a random variable taking values $x_1, x_2, \ldots, x_n$ with probabilities $p_1, p_2, \ldots, p_n$.

Computing Information Entropy

- Let X be a random variable taking values $x_1, x_2, \ldots, x_n$ with probabilities $p_1, p_2, \ldots, p_n$.
- Then it's entropy is given by the constrained optimization problem in n real dimensions

$$\begin{aligned} \min_{\ell_1, \ell_2, \ldots, \ell_n} \quad & \sum_{k=1}^{n} p_k \ell_k \\ \text{s.t.} \quad & \sum_{k=1}^{n} 2^{-\ell_k} = 1 \end{aligned}$$

where ℓ_k denotes the average length assigned to x_k.

Computing Information Entropy

- Let X be a random variable taking values $x_1, x_2, \ldots, x_n$ with probabilities $p_1, p_2, \ldots, p_n$.
- Then it's entropy is given by the constrained optimization problem in n real dimensions

$$\begin{aligned} \min_{\ell_1, \ell_2, \ldots, \ell_n} \quad & \sum_{k=1}^{n} p_k \ell_k \\ \text{s.t.} \quad & \sum_{k=1}^{n} 2^{-\ell_k} = 1 \end{aligned}$$

  where ℓ_k denotes the average length assigned to x_k.
- We can solve this problem using the method of lagrange multipliers.

Computing Information Entropy

- We define the Lagrangian function

$$\mathcal{L}(\ell_1, \ell_2, \ldots, \ell_n, \lambda) = \sum_{k=1}^{n} p_k \ell_k - \lambda \sum_{k=1}^{m} 2^{-\ell_k}$$

Computing Information Entropy

- We define the Lagrangian function

$$\mathcal{L}(\ell_1, \ell_2, \ldots, \ell_n, \lambda) = \sum_{k=1}^{n} p_k \ell_k - \lambda \sum_{k=1}^{m} 2^{-\ell_k}$$

- Differentiating yields

$$\frac{\partial \mathcal{L}}{\partial \ell_k} = p_k + \lambda \ln(2) 2^{-\ell_k}$$

Computing Information Entropy

- We define the Lagrangian function

$$\mathcal{L}(\ell_1, \ell_2, \ldots, \ell_n, \lambda) = \sum_{k=1}^{n} p_k \ell_k - \lambda \sum_{k=1}^{m} 2^{-\ell_k}$$

- Differentiating yields

$$\frac{\partial \mathcal{L}}{\partial \ell_k} = p_k + \lambda \ln(2) 2^{-\ell_k}$$

- Setting that equal to zero gives

$$\ell_k = -\log_2 \left(\frac{-p_k}{\lambda \ln(2)} \right)$$

Computing Information Entropy

- Next we re-paramaterize our Lagrange multiplier, setting

$$\hat{\lambda} = \frac{-1}{\lambda \ln(2)}$$

Computing Information Entropy

- Next we re-paramaterize our Lagrange multiplier, setting

$$\hat{\lambda} = \frac{-1}{\lambda \ln(2)}$$

- This yields

$$\ell_k = -\log_2\left(\hat{\lambda} p_k\right)$$

Computing Information Entropy

- Next we re-paramaterize our Lagrange multiplier, setting

$$\hat{\lambda} = \frac{-1}{\lambda \ln(2)}$$

- This yields

$$\ell_k = -\log_2\left(\hat{\lambda} p_k\right)$$

- Subbing this into our constraint gives

$$\sum_{k=1}^{n} 2^{-\ell_k} = \sum_{k=1}^{n} 2^{\log_2\left(\hat{\lambda p_k}\right)} = \hat{\lambda} \sum_{k=1}^{n} p_k = \hat{\lambda} = 1$$

Computing Information Entropy

- Next we re-paramaterize our Lagrange multiplier, setting

$$\hat{\lambda} = \frac{-1}{\lambda \ln(2)}$$

- This yields

$$\ell_k = -\log_2 \left(\hat{\lambda} p_k \right)$$

- Subbing this into our constraint gives

$$\sum_{k=1}^{n} 2^{-\ell_k} = \sum_{k=1}^{n} 2^{\log_2 \left(\hat{\lambda p_k} \right)} = \hat{\lambda} \sum_{k=1}^{n} p_k = \hat{\lambda} = 1$$

- It follows that the optimizer is given simply by

$$\ell_k = -\log_2(p_k)$$

Computing Information Entropy

Thus, we have that the information entropy of a random variable taking values $x_1, x_2, \ldots, x_n$ with probabilities $p_1, p_2, \ldots, p_n$ is simply

$$\boxed{H = -\sum_{k=1}^{n} p_k \log_2(p_k)}$$

An Alternative Characterization

- So, this is a beautiful result.

An Alternative Characterization

- So, this is a beautiful result.
- It is however, quite involved to make rigorous.

An Alternative Characterization

- So, this is a beautiful result.
- It is however, quite involved to make rigorous.
- So, now that we have this pinned down a bit, it's nice to have an alternative characterization of information entropy that is
  1. Easier to prove formally
  2. Coincident with our above result

An Alternative Characterization

- The idea is quite intuitive. For simplicity, we again begin with the simple case of the α-coin.

An Alternative Characterization

- The idea is quite intuitive. For simplicity, we again begin with the simple case of the α-coin.
- In particular, consider the result of ℓ iid α-coin tosses. There are 2^n possible strings, but many of those are very unlikely.

An Alternative Characterization

- The idea is quite intuitive. For simplicity, we again begin with the simple case of the α-coin.
- In particular, consider the result of ℓ iid α-coin tosses. There are 2^n possible strings, but many of those are very unlikely.
- Let's consider ℓ large, and think about asymptotically how many of these strings are possible?

An Alternative Characterization

- This distribution is incidentally well-understood. It's sum is a *binomial* distribution.

An Alternative Characterization

- This distribution is incidentally well-understood. It's sum is a *binomial* distribution.
- By the strong law of large numbers (say), we know that we need to consider strings that have roughly $\alpha\ell$ heads, those are the ones with non-negligible likelihood
- We can compute the number of those, with binomial coefficients.

An Alternative Characterization

- We have a string of length ℓ with $0 \leq h \leq \ell$ heads. Equivalently, we choose h heads from the set of ℓ characters.

An Alternative Characterization

- We have a string of length ℓ with $0 \leq h \leq \ell$ heads. Equivalently, we choose h heads from the set of ℓ characters.
- We can generate such a set by undergoing a full permutation of all ℓ elements in the set (or think of their indices if you prefer), and taking the first h as the chosen set.

$$\underbrace{\bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet}_{h} \;\Big|\; \underbrace{\bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet}_{\ell-h}$$

An Alternative Characterization

- We have a string of length ℓ with $0 \leq h \leq \ell$ heads. Equivalently, we choose h heads from the set of ℓ characters.
- We can generate such a set by undergoing a full permutation of all ℓ elements in the set (or think of their indices if you prefer), and taking the first h as the chosen set.

$$\underbrace{\bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet}_{h} \;\Bigg|\; \underbrace{\bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet}_{\ell - h}$$

- The chosen set is invariant under permutations of the first h and the last $\ell - h$, so we have to divide out by the number of those, but any other permutation that doesn't break down into two of those will break the set.

An Alternative Characterization

- We have a string of length ℓ with $0 \leq h \leq \ell$ heads. Equivalently, we choose h heads from the set of ℓ characters.
- We can generate such a set by undergoing a full permutation of all ℓ elements in the set (or think of their indices if you prefer), and taking the first h as the chosen set.

$$\underbrace{\bullet\quad\bullet\quad\bullet\quad\bullet\quad\bullet\quad\bullet}_{h}\ \Big|\ \underbrace{\bullet\quad\bullet\quad\bullet\quad\bullet\quad\bullet\quad\bullet\quad\bullet\quad\bullet\quad\bullet}_{\ell-h}$$

- The chosen set is invariant under permutations of the first h and the last $\ell - h$, so we have to divide out by the number of those, but any other permutation that doesn't break down into two of those will break the set.

$$\binom{\ell}{h} = \frac{\ell!}{h!(\ell - h)!}$$

An Alternative Characterization

- It follows pretty simply that our entropy ought to be asymptotically the number of bits needed to encompass all of the states with a number of heads equal to the mean as a ratio of ℓ.

$$H = \lim_{\ell \to \infty} \frac{\log_2 \left(\binom{\ell}{\alpha \ell} \right)}{\ell}$$

An Alternative Characterization

- It follows pretty simply that our entropy ought to be asymptotically the number of bits needed to encompass all of the states with a number of heads equal to the mean as a ratio of ℓ.

$$H = \lim_{\ell \to \infty} \frac{\log_2 \left(\binom{\ell}{\alpha \ell} \right)}{\ell} = \lim_{\ell \to \infty} \frac{\log_2 \left(\frac{\ell!}{(\alpha \ell)!(\ell - \alpha \ell)!} \right)}{\ell}$$

An Alternative Characterization

- It follows pretty simply that our entropy ought to be asymptotically the number of bits needed to encompass all of the states with a number of heads equal to the mean as a ratio of ℓ.

$$H = \lim_{\ell\to\infty} \frac{\log_2\left(\binom{\ell}{\alpha\ell}\right)}{\ell} = \lim_{\ell\to\infty} \frac{\log_2\left(\frac{\ell!}{(\alpha\ell)!(\ell-\alpha\ell)!}\right)}{\ell}$$

- We can use *Stirling's Approximation*, an asymptotic result about the factorial.

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

An Alternative Characterization

$$H = \lim_{\ell \to \infty} \frac{\log_2 \left(\binom{\ell}{\alpha \ell} \right)}{\ell}$$

An Alternative Characterization

$$H = \lim_{\ell \to \infty} \frac{\log_2 \left(\binom{\ell}{\alpha \ell} \right)}{\ell} = \lim_{\ell \to \infty} \frac{\log_2 \left(\frac{\ell!}{h!(\ell - h)!} \right)}{\ell}$$

An Alternative Characterization

$$H = \lim_{\ell\to\infty} \frac{\log_2\left(\binom{\ell}{\alpha\ell}\right)}{\ell} = \lim_{\ell\to\infty} \frac{\log_2\left(\frac{\ell!}{h!(\ell-h)!}\right)}{\ell}$$

$$= \lim_{\ell\to\infty} \frac{\log_2\left(\frac{\sqrt{2\pi\ell}\ell^{\ell}e^{-\ell}}{\left(\sqrt{2\pi\alpha\ell}(\alpha\ell)^{\alpha\ell}e^{-\alpha\ell}\right)\left(\sqrt{2\pi(1-\alpha)\ell}((1-\alpha)\ell)^{(1-\alpha)\ell}e^{-(1-\alpha)\ell}\right)}\right)}{\ell}$$

An Alternative Characterization

$$H = \lim_{\ell \to \infty} \frac{\log_2 \left(\binom{\ell}{\alpha \ell} \right)}{\ell} = \lim_{\ell \to \infty} \frac{\log_2 \left(\frac{\ell!}{h!(\ell - h)!} \right)}{\ell}$$

$$= \lim_{\ell \to \infty} \frac{\log_2 \left(\frac{\sqrt{2\pi\ell}\ell^{\ell} e^{-\ell}}{\left(\sqrt{2\pi\alpha\ell}(\alpha\ell)^{\alpha\ell} e^{-\alpha\ell}\right) \left(\sqrt{2\pi(1-\alpha)\ell}((1-\alpha)\ell)^{(1-\alpha)\ell} e^{-(1-\alpha)\ell}\right)} \right)}{\ell}$$

$$= \lim_{\ell \to \infty} \frac{\log_2 \left(\frac{1}{\sqrt{2\pi\alpha(1-\alpha)\ell}\alpha^{\alpha\ell}(1-\alpha)^{(1-\alpha)\ell}} \right)}{\ell}$$

An Alternative Characterization

$$H = \lim_{\ell\to\infty} \frac{\log_2\left(\binom{\ell}{\alpha\ell}\right)}{\ell} = \lim_{\ell\to\infty} \frac{\log_2\left(\frac{\ell!}{h!(\ell-h)!}\right)}{\ell}$$

$$= \lim_{\ell\to\infty} \frac{\log_2\left(\frac{\sqrt{2\pi\ell}\ell^{\ell}e^{-\ell}}{\left(\sqrt{2\pi\alpha\ell}(\alpha\ell)^{\alpha\ell}e^{-\alpha\ell}\right)\left(\sqrt{2\pi(1-\alpha)\ell}((1-\alpha)\ell)^{(1-\alpha)\ell}e^{-(1-\alpha)\ell}\right)}\right)}{\ell}$$

$$= \lim_{\ell\to\infty} \frac{\log_2\left(\frac{1}{\sqrt{2\pi\alpha(1-\alpha)\ell}\alpha^{\alpha\ell}(1-\alpha)^{(1-\alpha)\ell}}\right)}{\ell}$$

$$= \lim_{\ell\to\infty} \frac{\ell\left(-\alpha\log_2(\alpha) - (1-\alpha)\log_2(1-\alpha)\right) + O(\log\ell)}{\ell}$$

An Alternative Characterization

$$H = \lim_{\ell\to\infty} \frac{\log_2\left(\binom{\ell}{\alpha\ell}\right)}{\ell} = \lim_{\ell\to\infty} \frac{\log_2\left(\frac{\ell!}{h!(\ell-h)!}\right)}{\ell}$$

$$= \lim_{\ell\to\infty} \frac{\log_2\left(\frac{\sqrt{2\pi\ell}\ell^\ell e^{-\ell}}{\left(\sqrt{2\pi\alpha\ell}(\alpha\ell)^{\alpha\ell}e^{-\alpha\ell}\right)\left(\sqrt{2\pi(1-\alpha)\ell}((1-\alpha)\ell)^{(1-\alpha)\ell}e^{-(1-\alpha)\ell}\right)}\right)}{\ell}$$

$$= \lim_{\ell\to\infty} \frac{\log_2\left(\frac{1}{\sqrt{2\pi\alpha(1-\alpha)\ell}\alpha^{\alpha\ell}(1-\alpha)^{(1-\alpha)\ell}}\right)}{\ell}$$

$$= \lim_{\ell\to\infty} \frac{\ell\left(-\alpha\log_2(\alpha)-(1-\alpha)\log_2(1-\alpha)\right)+O(\log\ell)}{\ell}$$

$$= \boxed{-\alpha\log_2(\alpha)-(1-\alpha)\log_2(1-\alpha)}$$

An Alternative Characterization

- The same approach can work for a random variable taking a finite number of states using a *multinomial distribution*

An Alternative Characterization

- The same approach can work for a random variable taking a finite number of states using a *multinomial distribution*
- $$\binom{n}{k_1, k_2, \ldots, k_m} = \frac{n!}{k_1!k_2!\ldots k_m!} = n!\left(\prod_{j=1}^{m} k_j!\right)^{-1}$$

  where $\sum_{j=1}^{m} k_j = n$

An Alternative Characterization

- The same approach can work for a random variable taking a finite number of states using a *multinomial distribution*
- $$\binom{n}{k_1, k_2, \ldots, k_m} = \frac{n!}{k_1!k_2!\ldots k_m!} = n!\left(\prod_{j=1}^{m} k_j!\right)^{-1}$$

  where $\sum_{j=1}^{m} k_j = n$
- Imagine a random variable taking values $x_1, x_2, \ldots, x_n$ with probabilities $\alpha_1, \alpha_2, \ldots, \alpha_n$. By the same logic

$$H = \lim_{\ell \to \infty} \frac{1}{\ell}\left(\log_2\left(\binom{\ell}{\alpha_1\ell, \alpha_2\ell, \ldots, \alpha_n\ell}\right)\right)$$

An Alternative Characterization

$$H = \lim_{\ell \to \infty} \frac{1}{\ell} \left(\log_2 \left(\binom{\ell}{\alpha_1 \ell, \alpha_2 \ell, \ldots, \alpha_n \ell} \right) \right)$$

An Alternative Characterization

$$H = \lim_{\ell \to \infty} \frac{1}{\ell} \left(\log_2 \left(\binom{\ell}{\alpha_1 \ell, \alpha_2 \ell, \ldots, \alpha_n \ell} \right) \right)$$

$$= \lim_{\ell \to \infty} \frac{1}{\ell} \log_2 \left(\frac{\ell!}{\prod_{k=1}^{n} (\alpha_k \ell)!} \right)$$

An Alternative Characterization

$$H = \lim_{\ell\to\infty} \frac{1}{\ell} \left(\log_2 \left(\binom{\ell}{\alpha_1\ell, \alpha_2\ell, \ldots, \alpha_n\ell} \right) \right)$$

$$= \lim_{\ell\to\infty} \frac{1}{\ell} \log_2 \left(\frac{\ell!}{\prod_{k=1}^{n} (\alpha_k \ell)!} \right)$$

$$= \lim_{\ell\to\infty} \frac{1}{\ell} \log_2 \left(\frac{\sqrt{2\pi\ell}\,\ell^{\ell} e^{-\ell}}{\prod_{k=1}^{n} \sqrt{2\pi\alpha_k\ell}\,(\alpha_k\ell)^{\alpha_k\ell} e^{-\alpha_k\ell}} \right)$$

An Alternative Characterization

$$= \lim_{\ell \to \infty} \frac{\log_2 \left(\prod_{k=1}^{n} (\alpha_k)^{-\alpha_k \ell} \right) + O(n \log \ell)}{\ell}$$

An Alternative Characterization

$$= \lim_{\ell \to \infty} \frac{\log_2 \left(\prod_{k=1}^{n} (\alpha_k)^{-\alpha_k \ell} \right) + O(n \log \ell)}{\ell}$$

$$= \boxed{- \sum_{k=1}^{n} \alpha_k \log_2 \alpha_k}$$

Ideas of Information Theory

- Conditional Entropy

Ideas of Information Theory

- Conditional Entropy
- Exactly like conditional probability

Ideas of Information Theory

- Conditional Entropy
- Exactly like conditional probability
- $$H(X|Y) = \mathbb{E}_Y[H(X|Y=y)] = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log_2 \left(\frac{p(x)}{p(x,y)} \right)$$

Ideas of Information Theory

- Conditional Entropy
- Exactly like conditional probability
- $$H(X|Y) = \mathbb{E}_Y[H(X|Y=y)] = \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x,y) \log_2 \left(\frac{p(x)}{p(x,y)}\right)$$
- Bayes Rule

$$H(X) + H(Y|X) = H(Y) + H(X|Y)$$

Ideas of Information Theory

- Conditional Entropy
- Exactly like conditional probability
- $$H(X|Y) = \mathbb{E}_Y[H(X|Y=y)] = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log_2 \left(\frac{p(x)}{p(x,y)} \right)$$
- Bayes Rule
$$H(X) + H(Y|X) = H(Y) + H(X|Y)$$
- $H(X|Y) \leq H(X)$

Ideas of Information Theory

- Joint Entropy

Ideas of Information Theory

- Joint Entropy
- The total information in X and Y (rvs)

Ideas of Information Theory

- Joint Entropy
- The total information in X and Y (rvs)
- $$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log_2\left(p(x,y)\right)$$

Ideas of Information Theory

- Joint Entropy
- The total information in X and Y (rvs)
- $$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log_2\left(p(x,y)\right)$$
- Chain Rule
$$H(X,Y) = H(X) + H(Y|X)$$

Ideas of Information Theory

- Joint Entropy
- The total information in X and Y (rvs)
- $$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log_2(p(x,y))$$
- Chain Rule
$$H(X,Y) = H(X) + H(Y|X)$$

$$H(X_1, X_2, \ldots, X_n) = \sum_{k=1}^{n} H(X_k|H_1, H_2, \ldots, H_{k-1})$$

Ideas of Information Theory

- Joint Entropy
- The total information in X and Y (rvs)
- $$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log_2(p(x,y))$$
- Chain Rule
$$H(X,Y) = H(X) + H(Y|X)$$
$$H(X_1, X_2, \ldots, X_n) = \sum_{k=1}^{n} H(X_k|H_1, H_2, \ldots, H_{k-1})$$
- $\max\limits_{1\le k\le n} H(X_k) \le H(X_1, X_2, \ldots, X_n) \le \sum_{k=1}^{n} H(X_k)$

Ideas of Information Theory

- Cross Entropy

Ideas of Information Theory

- Cross Entropy
- The expected encoding length in bits of some rv if the encoding is optimized for another. (Therefore a measure of similarity/difference).

Ideas of Information Theory

- Cross Entropy
- The expected encoding length in bits of some rv if the encoding is optimized for another. (Therefore a measure of similarity/difference).
- Related to Kullback-Liebler Diveregnce

Ideas of Information Theory

- Cross Entropy
- The expected encoding length in bits of some rv if the encoding is optimized for another. (Therefore a measure of similarity/difference).
- Related to Kullback-Liebler Diveregnce
- $$H(P,Q) = -\sum_{x \in \mathcal{X}} p(x) \log_2(q(x))$$

Ideas of Information Theory

- Cross Entropy
- The expected encoding length in bits of some rv if the encoding is optimized for another. (Therefore a measure of similarity/difference).
- Related to Kullback-Liebler Diveregnce
- $$H(P, Q) = -\sum_{x \in \mathcal{X}} p(x) \log_2(q(x))$$
- Gibb's inequality

$$H(P, Q) \geq H(P, P)$$

- Yes, the notation is terrible

Ideas of Information Theory

- Mutual Information $I(X;Y) = H(X) - H(X|Y)$. The amount of information Y carries on X.

Ideas of Information Theory

- Mutual Information $I(X;Y) = H(X) - H(X|Y)$. The amount of information Y carries on X.
- Differential Entropy

$$H(x) = -\int_{\mathcal{X}} p(x) \log_2(p(x))\, \mathrm{d}x$$

A Proof of the Comparison-Based Sorting Lower Bound

- Imagine a permutation σ chose uniformly at random from S_n.
- A comparison-based sort makes Q queries of random variables C checking $\sigma(i_1) < \sigma(i_2)$, and $H(C_1) = 1$
- The entropy of σ is easily seen to be $\log_2(n!)$
- The joint entropy $H(\sigma, C_1, C_2, \ldots, C_Q) \geq H(\sigma) = \log_2(n!)$
- By the chain rule

$$
\begin{aligned}
H(\sigma, C_1, C_2, \ldots, C_Q) =& H(\sigma | C_1, C_2, \ldots, C_Q) \\
& + H(C_Q | C_1, C_2, \ldots, C_{Q-1}) \\
& + \cdots + H(C_1)
\end{aligned}
$$

- Hence

$$H(\sigma, C_1, C_2, \ldots, C_Q) \geq \log_2(n!) - Q$$

Entropy Decision Trees

- Very old method in Machine Learning.

Entropy Decision Trees

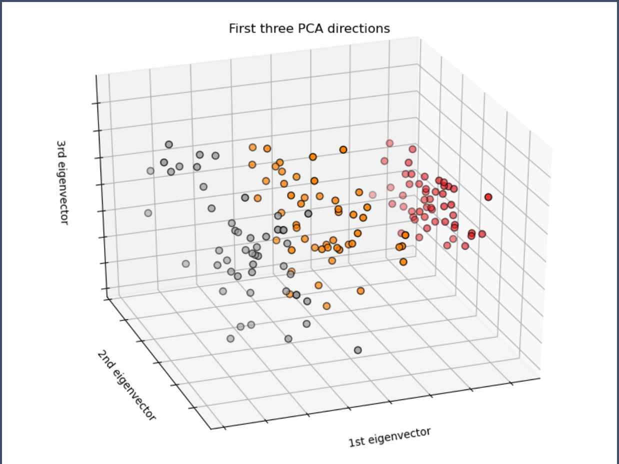
- Very old method in Machine Learning.
- The idea: split data-set in half recursively in some meaningful way.

Entropy Decision Trees

- Very old method in Machine Learning.
- The idea: split data-set in half recursively in some meaningful way.
- How to determine this splitting? So that the information entropy of the children is minimized!

Entropy Decision Trees

- Iris dataset. Flower data.

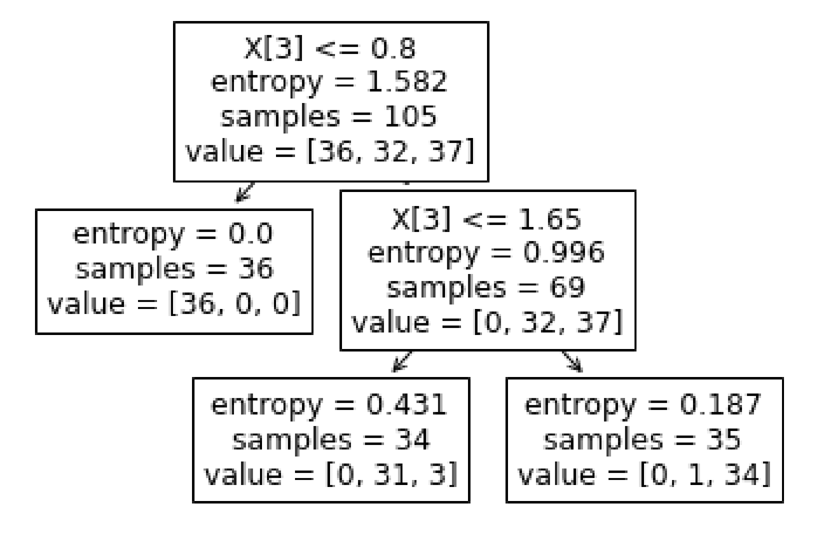


Entropy Decision Trees

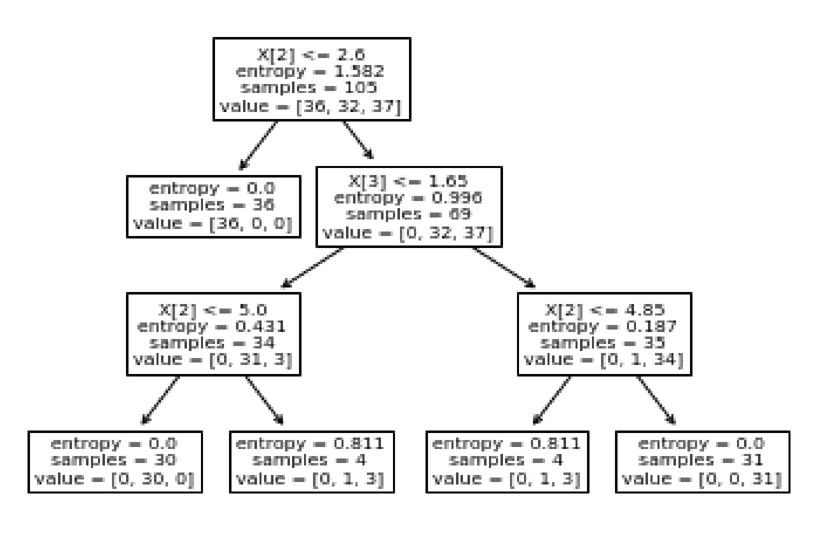
```python
from sklearn.datasets import load_iris
from sklearn import tree
from sklearn import metrics
from sklearn.model_selection import train_test_split

#Get the data
X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3,
                                                    random_state=1)
#Train the model
clf = tree.DecisionTreeClassifier(criterion="entropy", max_depth=2)
clf = clf.fit(X_train, y_train)
#Test and ouput
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
tree.plot_tree(clf)
```

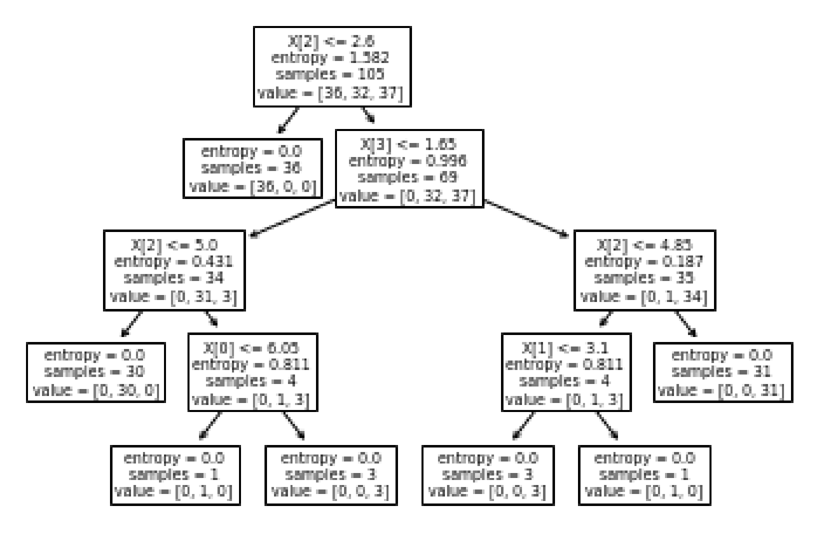
Entropy Decision Trees



Entropy Decision Trees



Entropy Decision Trees



Entropy Decision Trees

```
Console 1/A

In [18]: runfile('C:/Users/willi/OneDrive/Documents/college/freshman/
fall/CSH/SEMINAR/ITITWATML/decision_tree.py', wdir='C:/Users/willi/
OneDrive/Documents/college/freshman/fall/CSH/SEMINAR/ITITWATML')
Accuracy: 0.9555555555555556
```

Entropy as a Cost Function

- Suppose we are attempting binary classification, with a logistic model
$$p(\mathbf{x}) = \frac{1}{1 + \exp\left(\mathbf{w}^\top \mathbf{x}\right)}$$

Entropy as a Cost Function

- Suppose we are attempting binary classification, with a logistic model
$$p(\mathbf{x}) = \frac{1}{1 + \exp\left(\mathbf{w}^\top \mathbf{x}\right)}$$
- We have a bunch of observed values, and we want to tune **w**, but what should we minimize/maximize?

Entropy as a Cost Function

- Suppose we are attempting binary classification, with a logistic model

$$p(\mathbf{x}) = \frac{1}{1 + \exp\left(\mathbf{w}^\top \mathbf{x}\right)}$$

- We have a bunch of observed values, and we want to tune **w**, but what should we minimize/maximize?
- Could try the cross entropy between the model distribution and the observed distribution

Entropy as a Cost Function

- Suppose we are attempting binary classification, with a logistic model

$$p(\mathbf{x}) = \frac{1}{1 + \exp\left(\mathbf{w}^\top \mathbf{x}\right)}$$

- We have a bunch of observed values, and we want to tune **w**, but what should we minimize/maximize?
- Could try the cross entropy between the model distribution and the observed distribution

$$-\sum_{k=1}^{N} y_k \log(p(x_k)) + (1 - y_k) \log(1 - p(x_k))$$

- This is a result called logistic regression.

Entropy Decision Trees

decision_tree.py*

```python
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn import metrics
from sklearn.model_selection import train_test_split

#Get the data
X, y = load_iris(return_X_y=True)
#make it logistc
for i in range(len(y)):
    y[i] = 1 if y[i] == 2 else 0
#split it up
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3,
                                                    random_state=17)
#Train the model
mod = LogisticRegression()
mod.fit(X_train, y_train)
#Test and ouput
y_pred = mod.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))

```

Entropy Decision Trees

```
In [29]: runfile('C:/Users/willi/OneDrive/Documents/college/freshman/
fall/CSH/SEMINAR/ITITWATML/decision_tree.py', wdir='C:/Users/willi/
OneDrive/Documents/college/freshman/fall/CSH/SEMINAR/ITITWATML')
Accuracy: 0.9777777777777777
              precision    recall  f1-score   support

           0       0.97      1.00      0.98        31
           1       1.00      0.93      0.96        14

    accuracy                           0.98        45
   macro avg       0.98      0.96      0.97        45
weighted avg       0.98      0.98      0.98        45
```

Cross-Entropy as a Cost Function

- This is typically the most important application to machine learning.

Cross-Entropy as a Cost Function

- This is typically the most important application to machine learning.
- Neural Networks

Cross-Entropy as a Cost Function

- This is typically the most important application to machine learning.
- Neural Networks
- Other models

Cross-Entropy as a Cost Function

- This is typically the most important application to machine learning.
- Neural Networks
- Other models
- Priors for Bayesian Classification

Kernel Principle Component Analysis

- Normal PCA, you find the orthogonal projections that maximize the variance of the RV after projection.

Kernel Principle Component Analysis

- Normal PCA, you find the orthogonal projections that maximize the variance of the RV after projection.
- Can be computed by Eigendecomposition of the (usually MLE) sample covariance matrix.

Kernel Principle Component Analysis

- Normal PCA, you find the orthogonal projections that maximize the variance of the RV after projection.
- Can be computed by Eigendecomposition of the (usually MLE) sample covariance matrix.
- In a very complicated sense, if you imagine maximum generalized entropy projections in a special way, one gets out that one could actually does to PCA after mapping to some *feature space*. The inner product in feature space is a measure of closeness, a kernel.

Kernel Principle Component Analysis

- Normal PCA, you find the orthogonal projections that maximize the variance of the RV after projection.
- Can be computed by Eigendecomposition of the (usually MLE) sample covariance matrix.
- In a very complicated sense, if you imagine maximum generalized entropy projections in a special way, one gets out that one could actually does to PCA after mapping to some *feature space*. The inner product in feature space is a measure of closeness, a kernel.
- http://citeseerx.ist.psu.edu/ viewdoc/download?doi=10.1.1.381.288rep=rep1type=pdf