CMPSCI 240: "Reasoning Under Uncertainty"

Lecture 10

Prof. Hanna Wallach
wallach@cs.umass.edu

February 23, 2012

Reminders

- Check the course website: `http://www.cs.umass.edu/~wallach/courses/s12/cmpsci240/`
- Third homework is due TOMORROW
- IMPORTANT: check you can log into the EdLab in preparation for the fourth homework

Recap

Information Theory

- Probability and information content are inversely related

Last Time: Information Content

- If events $A_1, \ldots, A_n$ have probabilities $P(A_1), \ldots, P(A_n)$ and partition Ω, the information content $I(A_i)$ of event A_i is

$$I(A_i) = \log_2 \frac{1}{P(A_i)}$$

Last Time: Information Content

- If events $A_1, \ldots, A_n$ have probabilities $P(A_1), \ldots, P(A_n)$ and partition Ω, the information content $I(A_i)$ of event A_i is

$$I(A_i) = \log_2 \frac{1}{P(A_i)}$$

- Intuition: number of (theoretical) equiprobable yes/no questions required to uniquely identify that $x \in A_i$

Last Time: Information Content

- If events $A_1, \ldots, A_n$ have probabilities $P(A_1), \ldots, P(A_n)$ and partition Ω, the information content $I(A_i)$ of event A_i is

$$I(A_i) = \log_2 \frac{1}{P(A_i)}$$

- Intuition: number of (theoretical) equiprobable yes/no questions required to uniquely identify that $x \in A_i$
- Additive: $I(A \cap B) = I(A) + I(B \mid A) = I(B) + I(A \mid B)$

Last Time: Entropy

- Entropy: average information content of a set of n disjoint, mutually exclusive events $A_1, \ldots, A_n$ that partition Ω

$$H(A_1, \ldots, A_n) = \sum_{i=1}^{n} P(A_i) \log_2 \frac{1}{P(A_i)}$$

Last Time: Entropy

- Entropy: average information content of a set of n disjoint, mutually exclusive events $A_1, \ldots, A_n$ that partition Ω

$$H(A_1, \ldots, A_n) = \sum_{i=1}^{n} P(A_i) \log_2 \frac{1}{P(A_i)}$$

- Measure of uncertainty of the entire set of events: maximized when events are equiprobable, e.g., $P(A_1) = P(A_2) = 1/2$

Last Time: Information Rate

- Suppose $A_1, \ldots, A_n$ are encoded using $L(A_1), \ldots, L(A_n)$ bits, the information rate is the average number of bits per event

$$R(A_1, \ldots, A_n) = \sum_{i=1}^{n} P(A_i)\, L(A_i)$$

Last Time: Information Rate

- Suppose $A_1, \ldots, A_n$ are encoded using $L(A_1), \ldots, L(A_n)$ bits, the information rate is the average number of bits per event

$$R(A_1, \ldots, A_n) = \sum_{i=1}^{n} P(A_i)\, L(A_i)$$

- Entropy $H(A_1, \ldots, A_n)$ is the best achievable (lowest possible) information rate if events must be uniquely encoded

Last Time: Representing Events

- Fixed length codes: use same number of bits to encode each event, e.g., $A_1 = 11$, $A_2 = 10$, $A_3 = 01$, $A_4 = 00$

Last Time: Representing Events

- Fixed length codes: use same number of bits to encode each event, e.g., $A_1 = 11$, $A_2 = 10$, $A_3 = 01$, $A_4 = 00$
- Optimal for events with equal probabilities

Last Time: Representing Events

- Fixed length codes: use same number of bits to encode each event, e.g., $A_1 = 11$, $A_2 = 10$, $A_3 = 01$, $A_4 = 00$
- Optimal for events with equal probabilities
- Variable length codes: use different number of bits to encode each event, e.g., $A_1 = 1$, $A_2 = 01$, $A_3 = 001$, $A_4 = 000$

Last Time: Representing Events

- Fixed length codes: use same number of bits to encode each event, e.g., $A_1 = 11$, $A_2 = 10$, $A_3 = 01$, $A_4 = 00$
- Optimal for events with equal probabilities
- Variable length codes: use different number of bits to encode each event, e.g., $A_1 = 1$, $A_2 = 01$, $A_3 = 001$, $A_4 = 000$
- Optimal for events with unequal probabilities

Compression

Compression

- Message: sequence of events, e.g., $A_1 A_1 A_3 A_1 A_2 A_4$

Compression

- Message: sequence of events, e.g., $A_1 A_1 A_3 A_1 A_2 A_4$
- Goal: represent messages using as few bits as possible

Compression

- Message: sequence of events, e.g., $A_1 A_1 A_3 A_1 A_2 A_4$
- Goal: represent messages using as few bits as possible
- Compressed messages must be uniquely decodeable

Compression

- Message: sequence of events, e.g., $A_1 A_1 A_3 A_1 A_2 A_4$
- Goal: represent messages using as few bits as possible
- Compressed messages must be uniquely decodeable
- Simplest binary code: fixed length, k bits to encode 2^k events

Compression

- Message: sequence of events, e.g., $A_1 A_1 A_3 A_1 A_2 A_4$
- Goal: represent messages using as few bits as possible
- Compressed messages must be uniquely decodeable
- Simplest binary code: fixed length, k bits to encode 2^k events
- Variable length code: short bit strings for probable events

Compression

- Message: sequence of events, e.g., $A_1 A_1 A_3 A_1 A_2 A_4$
- Goal: represent messages using as few bits as possible
- Compressed messages must be uniquely decodeable
- Simplest binary code: fixed length, k bits to encode 2^k events
- Variable length code: short bit strings for probable events
- Compression limit: determined by entropy

Decompression and Prefix Codes

- Decompressing messages is hard for variable-length codes

Decompression and Prefix Codes

- Decompressing messages is hard for variable-length codes
- e.g., $A_1 = 0$, $A_2 = 00$, $A_3 = 000$, what's 0000?

Decompression and Prefix Codes

- Decompressing messages is hard for variable-length codes
- e.g., $A_1 = 0$, $A_2 = 00$, $A_3 = 000$, what's 0000?
- e.g., $A_1 = 0$, $A_2 = 01$, $A_3 = 011$, what's 00?

Decompression and Prefix Codes

- Decompressing messages is hard for variable-length codes
- e.g., $A_1 = 0$, $A_2 = 00$, $A_3 = 000$, what's 0000?
- e.g., $A_1 = 0$, $A_2 = 01$, $A_3 = 011$, what's 00?

- Prefix code: no "code word" is a prefix of any other

Decompression and Prefix Codes

- Decompressing messages is hard for variable-length codes
- e.g., $A_1 = 0$, $A_2 = 00$, $A_3 = 000$, what's 0000?
- e.g., $A_1 = 0$, $A_2 = 01$, $A_3 = 011$, what's 00?
- Prefix code: no "code word" is a prefix of any other
- e.g., $A_1 = 0$, $A_2 = 10$, $A_3 = 110$, $A_4 = 111$

Prefix Codes and Binary Trees

- Consider a binary tree with events $A_1, \ldots, A_n$ as leaves

Prefix Codes and Binary Trees

- Consider a binary tree with events $A_1, \ldots, A_n$ as leaves
- Encode each event A_i as the unique bit string that identifies A_i (i.e., represents the path from the root to A_i)

Prefix Codes and Binary Trees

- Consider a binary tree with events $A_1, \ldots, A_n$ as leaves
- Encode each event A_i as the unique bit string that identifies A_i (i.e., represents the path from the root to A_i)
- Any code constructed this way will be a prefix code

Prefix Codes and Binary Trees

- Consider a binary tree with events $A_1, \ldots, A_n$ as leaves
- Encode each event A_i as the unique bit string that identifies A_i (i.e., represents the path from the root to A_i)

- Any code constructed this way will be a prefix code
- But not necessarily optimal (information rate $\geq$ entropy)

Optimal Prefix Codes

	A_1	A_2	A_3	A_4	A_5	A_6	A_7
$P(A_i)$	0.01	0.24	0.05	0.20	0.47	0.01	0.02

- Goal: prefix code with information rate = entropy = 1.93

Optimal Prefix Codes

	A_1	A_2	A_3	A_4	A_5	A_6	A_7
$P(A_i)$	0.01	0.24	0.05	0.20	0.47	0.01	0.02

- Goal: prefix code with information rate = entropy = 1.93
- We've (kind of) seen this already...

Optimal Prefix Codes

	A_1	A_2	A_3	A_4	A_5	A_6	A_7
$P(A_i)$	0.01	0.24	0.05	0.20	0.47	0.01	0.02

- Goal: prefix code with information rate $=$ entropy $= 1.93$
- We've (kind of) seen this already...
- A balanced binary tree $\implies$ shorter code words

Building Prefix Codes

	A_1	A_2	A_3	A_4	A_5	A_6	A_7
$P(A_i)$	0.01	0.24	0.05	0.20	0.47	0.01	0.02

- Top-down construction: build the tree from the root down

Building Prefix Codes

	A_1	A_2	A_3	A_4	A_5	A_6	A_7
$P(A_i)$	0.01	0.24	0.05	0.20	0.47	0.01	0.02

- Top-down construction: build the tree from the root down
- Does not nessarily result in an optimal prefix code:

$$H(A_1, \ldots, A_n) \leq R(A_1, \ldots, A_n) \leq H(A_1, \ldots, A_n) + 2$$

Huffman Coding

	A_1	A_2	A_3	A_4	A_5	A_6	A_7
$P(A_i)$	0.01	0.24	0.05	0.20	0.47	0.01	0.02

- Bottom-up construction: build the tree from the leaves up

Huffman Coding

	A_1	A_2	A_3	A_4	A_5	A_6	A_7
$P(A_i)$	0.01	0.24	0.05	0.20	0.47	0.01	0.02

- Bottom-up construction: build the tree from the leaves up
- Upper bound on information rate is better:

$$H(A_1, \ldots, A_n) \leq R(A_1, \ldots, A_n) < H(A_1, \ldots, A_n) + 1$$

Huffman Coding

	A_1	A_2	A_3	A_4	A_5	A_6	A_7
$P(A_i)$	0.01	0.24	0.05	0.20	0.47	0.01	0.02

- Bottom-up construction: build the tree from the leaves up
- Upper bound on information rate is better:

$$H(A_1, \ldots, A_n) \leq R(A_1, \ldots, A_n) < H(A_1, \ldots, A_n) + 1$$

- Can prove this is optimal for a prefix code

[Got to here in class...]

Communicating Perfectly

Transmitting Information

- Goal: transmit some message, encoded in binary

Transmitting Information

- Goal: transmit some message, encoded in binary
- Want to make sure the correct message is received even if there are transmission errors (e.g., static, disk failure, ...)

Transmitting Information

- Goal: transmit some message, encoded in binary
- Want to make sure the correct message is received even if there are transmission errors (e.g., static, disk failure, ...)
- Probability of a single bit being flipped is p

Transmitting Information

- Goal: transmit some message, encoded in binary
- Want to make sure the correct message is received even if there are transmission errors (e.g., static, disk failure, ...)
- Probability of a single bit being flipped is p
- Error probability: overall probability of there being an undetected error when using some encoding scheme

Examples of Error Probability

- e.g., 8 events represented as 000, 001, 010, . . . 111, probability of a single bit flip is $1\,/\,10$, what is the error probability?

Encoding with Redundancy

- Can use additional bits when encoding events to ensure that they are "protected" against errors in transmission

Encoding with Redundancy

- Can use additional bits when encoding events to ensure that they are "protected" against errors in transmission
- Error detecting codes vs. error correcting codes

Encoding with Redundancy

- Can use additional bits when encoding events to ensure that they are "protected" against errors in transmission
- Error detecting codes vs. error correcting codes
- Fundamental trade-off: want encoding schemes that minimize both the error probability and the information rate

Error-Detecting Codes: Parity Check Codes

- Append a parity bit to each code word such that every code word always contains an even number of ones

Error-Detecting Codes: Parity Check Codes

- Append a parity bit to each code word such that every code word always contains an even number of ones
- e.g., 0000, 0011, 0101, . . . , 1100, 1111

Error-Detecting Codes: Parity Check Codes

- Append a parity bit to each code word such that every code word always contains an even number of ones
- e.g., 0000, 0011, 0101, . . . , 1100, 1111
- Can detect error if an odd number of bits get flipped

Error-Detecting Codes: Parity Check Codes

- Append a parity bit to each code word such that every code word always contains an even number of ones
- e.g., 0000, 0011, 0101, . . . , 1100, 1111
- Can detect error if an odd number of bits get flipped
- Cannot detect error if an even number of bits get flipped

Examples of Parity Check Codes

- e.g., 8 events represented as 0000, 0011, . . . , 1111, probability of a single bit flip is $1 \,/\, 10$, what is the error probability?

Hamming Distance

- Hamming distance: number of positions (bits) in which two binary strings of equal length differ from each other

Hamming Distance

- Hamming distance: number of positions (bits) in which two binary strings of equal length differ from each other
- Adding a parity bit means that any two code words have a Hamming distance of at least 2 from each other

Hamming Distance

- Hamming distance: number of positions (bits) in which two binary strings of equal length differ from each other
- Adding a parity bit means that any two code words have a Hamming distance of at least 2 from each other
- e.g., 000 and 001 vs. 0000 and 0011

Hamming Distance

- Hamming distance: number of positions (bits) in which two binary strings of equal length differ from each other
- Adding a parity bit means that any two code words have a Hamming distance of at least 2 from each other
- e.g., 000 and 001 vs. 0000 and 0011
- Can only detect odd number of bit flips, can't correct errors

Error-Correcting Codes: 7/4 Hamming Codes

- e.g., 16 events represented as 0000, 0001, . . . , 1111

Error-Correcting Codes: 7/4 Hamming Codes

- e.g., 16 events represented as 0000, 0001, ..., 1111
- Add 3 bits $t_5 t_6 t_7$ to each code word $s_1 s_2 s_3 s_4$ such that

$$t_5 = s_1 + s_2 + s_3 \pmod 2$$
$$t_6 = s_2 + s_3 + s_4 \pmod 2$$
$$t_7 = s_3 + s_4 + s_1 \pmod 2$$

Error-Correcting Codes: 7/4 Hamming Codes

- e.g., 16 events represented as 0000, 0001, ..., 1111
- Add 3 bits $t_5 t_6 t_7$ to each code word $s_1 s_2 s_3 s_4$ such that

$$t_5 = s_1 + s_2 + s_3 \pmod 2$$
$$t_6 = s_2 + s_3 + s_4 \pmod 2$$
$$t_7 = s_3 + s_4 + s_1 \pmod 2$$

- e.g., what do 0000, 0001, ..., 0101, ..., 1111 become?

Error-Correcting Codes: 7/4 Hamming Codes

- Any two code words have a Hamming distance of ≥ 3

Error-Correcting Codes: 7/4 Hamming Codes

- Any two code words have a Hamming distance of ≥ 3
- 1 bit flip can be detected and corrected

Error-Correcting Codes: 7/4 Hamming Codes

- Any two code words have a Hamming distance of ≥ 3
- 1 bit flip can be detected and corrected
- ≥ 2 bit flips will be corrected to the wrong code word

Error-Correcting Codes: 7/4 Hamming Codes

- Any two code words have a Hamming distance of ≥ 3
- 1 bit flip can be detected and corrected
- ≥ 2 bit flips will be corrected to the wrong code word
- 2 bit flips can be detected using a global parity bit $\implies 8/4$

Correcting Single-Bit Errors

- Write the received code word in 3 overlapping circles

Correcting Single-Bit Errors

- Write the received code word in 3 overlapping circles
- Goal: every circle should have parity 0 (i.e., even # 1s)

Correcting Single-Bit Errors

- Write the received code word in 3 overlapping circles
- Goal: every circle should have parity 0 (i.e., even # 1s)
- Check each circle to see if its parity is 0 or 1

Correcting Single-Bit Errors

- Write the received code word in 3 overlapping circles
- Goal: every circle should have parity 0 (i.e., even # 1s)
- Check each circle to see if its parity is 0 or 1
- Is there a single unique bit (s or t) that lies inside all the parity 1 circles but outside all the parity 0 circles?

Correcting Single-Bit Errors

- Write the received code word in 3 overlapping circles
- Goal: every circle should have parity 0 (i.e., even # 1s)
- Check each circle to see if its parity is 0 or 1
- Is there a single unique bit (s or t) that lies inside all the parity 1 circles but outside all the parity 0 circles?
- If so, flipping this bit accounts for the parity violation

Examples of Decoding 7/4 Hamming Codes

- e.g., suppose 1000101 was transmitted but a) $1000\underline{0}01$, b) $1\underline{1}00101$, c) $10\underline{1}0101$, d) $10\underline{1}010\underline{0}$ were received?

Examples of Error Probability

- e.g., 16 events represented as 0000, 0001, . . . 1111, probability of a single bit flip is $1 / 10$, what is the error probability?

Examples of Error Probability

- e.g., 16 events now represented as 0000000, 0001011, . . . , 1111111, now what is the error probability?

For Next Time

- Check the course website: `http://www.cs.umass.edu/~wallach/courses/s12/cmpsci240/`
- Third homework is due TOMORROW
- IMPORTANT: check you can log into the EdLab in preparation for the fourth homework