Predictive Lossless Coding

\[ s_n + u_n \xrightarrow{\text{entropy encoder}} \hat{s}_n \]

\[ u_n \xrightarrow{\text{bitstream}} \]

\[ \hat{s}_n \xrightarrow{\text{entropy decoder}} s_n \]

\[ s_n - \hat{s}_n \xrightarrow{\text{prediction}} \hat{s}_n \xrightarrow{\text{prediction}} \]
Arithmetic Coding

- Practical realization of Shannon-Fano-Elias Coding (using standard integer arithmetic)
- No codeword table, on-the-fly encoding and decoding

Arithmetic Coding vs Huffman Coding

- For given block size $N$: Huffman coding is optimal, arithmetic coding is suboptimal
- Arithmetic coding is realizable for large $N$, while Huffman coding is not

Coding Efficiency of Arithmetic Coding

- Given probabilities and large $N$: Coding efficiency is very close to theoretical optimum
- In practice: Coding efficiency depends on using suitable probabilities
  - Adaptive pmf estimation during encoding and decoding
  - Using conditional pmfs (switch adaptive pmfs during encoding and decoding)
Arithmetic Coding with Adaptive Pmfs

Basic Idea
- Update pmf after encoding of each symbol
- Update pmf after decoding of each symbol

Straightforward realization
- Count occurrences $N_k$ of alphabet letters $a_k$
- $V$-bit probabilities $p_V(a_k)$ are given by
  $$p_V(a_k) = \lfloor 2^V \cdot N_k \sum_k N_k \rfloor$$
  (rounding down!)

Initialization:
- $\forall k, N_k = 1$

Controlling Adaptation Speed
One possibility:
- Rescale counts after sum exceeds some limit

```cpp
class Pmf // example for adaptive pmf implementation
{
public :
Pmf ( int Vbits , int numLetters , int maxSumCounts ) : V ( Vbits ), maxSum ( maxSumCounts ), sumCounts ( numLetters ), counts ( numLetters , 1 ) // all counts = 1 {}

int operator [] ( int index ) const {
  return ( counts [ index ] << V ) / sumCounts ;
}

void update ( int index ) {
  counts [ index ]++;
  if( ++ sumCounts >= maxSum ) {
    sumCounts = 0;
    for ( auto & cnt : counts ) // rounding up !
      sumCounts += ( cnt = ( cnt + 1 ) >> 1 );
  }
}

private :
const int V; // number of bits for pmf
const int maxSum ; // maximum sum of counts
int sumCounts ; // sum of all counts
vector <int > counts ; // counts for letters
};
```
Arithmetic Coding with Adaptive Pmfs

Basic Idea
- Update pmf after encoding of each symbol
- Update pmf after decoding of each symbol

Straightforward realization
- Count occurrences $N_k$ of alphabet letters $a_k$
- $V$-bit probabilities $p_V(a_k)$ are given by

$$p_V(a_k) = \left\lfloor 2^V \cdot \frac{N_k}{\sum_k N_k} \right\rfloor \quad \text{(rounding down!)}$$

- Initialization: $\forall k, \ N_k = 1$
Arithmetic Coding with Adaptive Pmfs

Basic Idea
- Update pmf after encoding of each symbol
- Update pmf after decoding of each symbol

Straightforward realization
- Count occurrences $N_k$ of alphabet letters $a_k$
- $V$-bit probabilities $p_V(a_k)$ are given by

$$p_V(a_k) = \left\lfloor 2^V \cdot \frac{N_k}{\sum_k N_k} \right\rfloor \quad \text{(rounding down !)}$$

- Initialization: $\forall k, \quad N_k = 1$

Controlling Adaptation Speed
- One possibility:
  Rescale counts after sum exceeds some limit
Arithmetic Coding with Adaptive Pmfs

Basic Idea
- Update pmf after encoding of each symbol
- Update pmf after decoding of each symbol

Straightforward realization
- Count occurrences $N_k$ of alphabet letters $a_k$
- $V$-bit probabilities $p_V(a_k)$ are given by

$$p_V(a_k) = \left\lfloor 2^V \cdot \frac{N_k}{\sum_k N_k} \right\rfloor \text{ (rounding down!)}$$

- Initialization: $\forall k, N_k = 1$

Controlling Adaptation Speed
- One possibility:
  
  Rescale counts after sum exceeds some limit

```cpp
class Pmf // example for adaptive pmf implementation
{
public:
    Pmf( int Vbits, int numLetters, int maxSumCounts )
    : V ( Vbits ), maxSum ( maxSumCounts ), // adaption speed
      sumCounts( numLetters ),
      counts ( numLetters, 1 ) // all counts = 1
    {}

    int operator[]( int index ) const {
        return ( counts[ index ] << V ) / sumCounts;
    }

    void update( int index ) {
        counts[ index ]++;
        if( ++sumCounts >= maxSum ) {
            sumCounts = 0;
            for( auto& cnt : counts ) // rounding up!
                sumCounts += ( cnt = ( cnt + 1 ) >> 1 );
        }
    }

private:
    const int V; // number of bits for pmf
    const int maxSum; // maximum sum of counts
    int sumCounts; // sum of all counts
    vector<int> counts; // counts for letters
};
```
Arithmetic Coding with Conditional Pmfs

Basic Idea

- Switch pmf after encoding of each symbol
- Switch pmf after decoding of each symbol
- Can be combined with adaptive pmfs
Arithmetic Coding with Conditional Pmfs

Basic Idea
- Switch pmf after encoding of each symbol
- Switch pmf after decoding of each symbol
- Can be combined with adaptive pmfs

Conditions for 1D Signals (e.g., audio)
- Directly preceding sample
- Two (or more) directly preceding samples
- Function of one or more preceding samples
Arithmetic Coding with Conditional Pmfs

Basic Idea

- Switch pmf after encoding of each symbol
- Switch pmf after decoding of each symbol
- Can be combined with adaptive pmfs

Conditions for 1D Signals (e.g., audio)

- Directly preceding sample
- Two (or more) directly preceding samples
- Function of one or more preceding samples

Conditions for 2D Signals (e.g., images)

- One already coded neighboring sample
- Two or more already coded neighboring samples
- Function of already coded neighboring samples
Arithmetic Coding with Conditional Pmfs

Basic Idea
- Switch pmf after encoding of each symbol
- Switch pmf after decoding of each symbol
- Can be combined with adaptive pmfs

Conditions for 1D Signals (e.g., audio)
- Directly preceding sample
- Two (or more) directly preceding samples
- Function of one or more preceding samples

Conditions for 2D Signals (e.g., images)
- One already coded neighboring sample
- Two or more already coded neighboring samples
- Function of already coded neighboring samples

```
void encodeMessage(const vector<int>& message, ...)
{
    vector<Pmf> pmfs(numLetters, {...});
    ArithEncoder aenc(...);
    int lastSymbol = 0;
    for (const auto& currSymbol : message) {
        Pmf& currPmf = pmfs[lastSymbol];
        aenc.encode(currSymbol, currPmf);
        currPmf.update(currSymbol);
        lastSymbol = currSymbol;
    }
    aenc.terminate();
}
```

```
vector<int> decodeMessage(int numSymbols, ...)
{
    vector<int> message;
    vector<Pmf> pmfs(numLetters, {...});
    ArithDecoder adec(...);
    int symbol = 0;
    while (message.size() < numSymbols) {
        Pmf& currPmf = pmfs[symbol];
        symbol = adec.decode(currPmf);
        currPmf.update(symbol);
        message.push_back(symbol);
    }
    return message;
}
```
Arithmetic Coding with Conditional Pmfs

Basic Idea
- Switch pmf after encoding of each symbol
- Switch pmf after decoding of each symbol
- Can be combined with adaptive pmfs

Conditions for 1D Signals (e.g., audio)
- Directly preceding sample
- Two (or more) directly preceding samples
- Function of one or more preceding samples

Conditions for 2D Signals (e.g., images)
- One already coded neighboring sample
- Two or more already coded neighboring samples
- Function of already coded neighboring samples

```c++
void encodeMessage( const vector<int>& message, ... )
{
    vector<Pmf> pmfs( numLetters, {...} );
    ArithEncoder aenc( ... );
    int lastSymbol = 0;
    for( const auto& currSymbol : message ) {
        Pmf& currPmf = pmfs[ lastSymbol ];
        aenc.encode( currSymbol, currPmf );
        currPmf.update( currSymbol );
        lastSymbol = currSymbol;
    }
    aenc.terminate();
}

vector<int> decodeMessage( int numSymbols, ... )
{
    vector<int> message;
    vector<Pmf> pmfs( numLetters, {...} );
    ArithDecoder adec( ... );
    int symbol = 0;
    while( message.size() < numSymbols ) {
        Pmf& currPmf = pmfs[ symbol ];
        symbol = adec.decode( currPmf );
        currPmf.update( symbol );
        message.push_back( symbol );
    }
    return message;
}
```
Example: Conditional Arithmetic Coding for Document Scans

Original: 1 052 713 bytes

gzip: 183 705 bytes (17.45 %)

bzip2: 169 049 bytes (16.05 %)

lzip: 140 307 bytes (13.33 %)

A Method for the Construction of Minimum-Redundancy Codes*

DAVID A. HUFFMAN†, ASSOCIATE, IEEE

**Summary**—An optimum method of coding an ensemble of messages consisting of a finite number of members is developed. A minimum-redundancy code is one constructed in such a way that the average number of coding digits per message is minimized.

**Introduction**

One important method of transmitting messages is to transmit in their place sequences of symbols. If there are more messages which might be sent than there are kinds of symbols available, then some of the messages must use more than one symbol. If it is assumed that each symbol requires the same time for transmission, then the time for transmission (length) of a message is directly proportional to the number of symbols associated with it. In this paper, the symbol or sequence of symbols associated with a given message will be called the *message code.* The entire number of messages which might be transmitted will be called the *message ensemble.* The mutual agreement between the transmitter and the receiver about the meaning of the code for each message of the ensemble will be called the *ensemble code.*

Probably the most familiar ensemble code was stated in the phrase "one if by land and two if by sea." In this will be defined here as an ensemble code which, for a message ensemble consisting of a finite number of members, 

$n$, and for a given number of coding digits, $D$, yields the lowest possible average message length. In order to avoid the use of the lengthy term "minimum-redundancy," this term will be replaced here by "optimum." It will be understood then that, in this paper, "optimum code" means "minimum-redundancy code."

The following basic restrictions will be imposed on an ensemble code:

(a) No two messages will consist of identical arrangements of coding digits.

(b) The message codes will be constructed in such a way that no additional indication is necessary to specify where a message code begins and ends once the starting point of a sequence of messages is known.

Restriction (b) necessitates that no message code be coded in such a way that its code appears, digit for digit, as the first part of any message code of greater length. Thus, 01, 102, 111, and 202 are valid message codes for an ensemble of four members. For instance, a sequence of these messages 111012202011111102 can be broken up...
Arithmetic Coding in Practice / Conditional PMFs

Example: Conditional Arithmetic Coding for Document Scans

1. No condition: 1 binary pmf
   → File size = 418,369 bytes (39.74%)

Original: 1,052,713 bytes

gzip: 183,705 bytes (17.45%)
bzip2: 169,049 bytes (16.05%)
lzop: 140,307 bytes (13.33%)

Heiko Schwarz (Freie Universität Berlin) — Data Compression: Predictive Lossless Coding
Example: Conditional Arithmetic Coding for Document Scans

1. No condition: 1 binary pmf
   ➔ File size = 418,369 bytes (39.74%)

2. Left neighbour: 2 binary pmfs
   ➔ File size = 192,841 bytes (18.32%)

Original: 1,052,713 bytes

gzip: 183,705 bytes (17.45%)
bzip2: 169,049 bytes (16.05%)
lzip: 140,307 bytes (13.33%)

Summary—An optimum method of coding an ensemble of messages consisting of a finite number of members is developed. A minimum-redundancy code in one constructed in such a way that the average number of coding digits per message is minimized.

Introduction

One important method of transmitting messages is to transmit in their place sequences of symbols. If there are more messages which might be sent than there are kinds of symbols available, then some of the messages must use more than one symbol. If it is assumed that each symbol requires the same time for transmission, then the time for transmission (length) of a message is directly proportional to the number of symbols associated with it. In this paper, the symbol or sequence of symbols associated with a given message will be called the “message code.” The entire number of messages which might be transmitted will be called the “message ensemble.” The mutual agreement between the transmitter and the receiver about the meaning of the code for each message of the ensemble will be called the “ensemble code.”

Probably the most familiar ensemble code was stated in the rhyme “one if by land and two if by sea.” In this way, the code can be defined here as an ensemble code which, for a message ensemble consisting of a finite number of members, N, and for a given number of coding digits, D, yields the lowest possible average message length. In order to avoid the use of the lengthy term “minimum-redundancy,” this term will be replaced here by “optimum.” It will be understood then that, in this paper, the “optimum code” means “minimum-redundancy code.”

The following basic restrictions will be imposed on an ensemble code:
(a) No two messages will consist of identical arrangements of coding digits.
(b) The message codes will be constructed in such a way that no additional indication is necessary to specify where a message code begins and ends once the starting point of a sequence of messages is known.

Restriction (b) necessitates that no message code be coded in such a way that its code appears, digit for digit, as the first part of any message code of greater length. Thus, 01, 102, 111, and 201 are valid message codes for an ensemble of four members. For instance, a sequence of these messages 111102020101111102 can be broken up...
Example: Conditional Arithmetic Coding for Document Scans

1. No condition: 1 binary pmf
   ➞ File size = 418369 bytes (39.74%)

2. Left neighbour: 2 binary pmfs
   ➞ File size = 192841 bytes (18.32%)

3. Left and above: 4 binary pmfs
   ➞ File size = 120198 bytes (11.42%)

Original: 1 052 713 bytes

gzip: 183 705 bytes (17.45%)
bzip2: 169 049 bytes (16.05%)
lzip: 140 307 bytes (13.33%)

A Method for the Construction of Minimum-Redundancy Codes*

DAVID A. HUFFMAN†, ASSOCIATE, IEEE

Summary—An optimum method of coding an ensemble of messages consisting of a finite number of members is developed. A minimum-redundancy code is one constructed in such a way that the average number of coding digits per message is minimized.

PROCEEDINGS OF THE I.R.E.

September

Introduction

One important method of transmitting messages is to transmit in their place sequences of symbols. If there are more messages which might be sent than there are kinds of symbols available, then some of the messages must use more than one symbol. If it is assumed that each symbol requires the same time for transmission, then the time for transmission (length) of a message is directly proportional to the number of symbols associated with it. In this paper, the symbol or sequence of symbols associated with a given message will be called the "message code." The entire number of messages which might be transmitted will be called the "message ensemble." The mutual agreement between the transmitter and the receiver about the meaning of the code for each message of the ensemble will be called the "ensemble code."

Probably the most familiar ensemble code was stated in the phrase "one if by land and two if by sea." In this

will be defined here as an ensemble code which, for a message ensemble consisting of a finite number of members, N, and for a given number of coding digits, D, yields the lowest possible average message length. In order to avoid the use of the lengthy term "minimum-redundancy," this term will be replaced here by "optimum." It will be understood then that, in this paper, "optimum code means "minimum-redundancy code."

The following basic restrictions will be imposed on an ensemble code:

(a) No two messages will consist of identical arrangements of coding digits.

(b) The message codes will be constructed in such a way that no additional indication is necessary to specify where a message code begins and ends once the starting point of a sequence of messages is known.

Restriction (b) necessitates that no message be coded in such a way that its code appears, digit for digit, as the first part of any message code of greater length. Thus, 01, 102, 111, and 202 are valid message codes for an ensemble of four members. For instance, a sequence of these messages 11110220201111102 can be broken up

Heiko Schwarz (Freie Universität Berlin) — Data Compression: Predictive Lossless Coding
Example: Conditional Arithmetic Coding for Document Scans

1. No condition: 1 binary pmf
   - File size = 418,369 bytes (39.74%)

2. Left neighbour: 2 binary pmfs
   - File size = 192,841 bytes (18.32%)

3. Left and above: 4 binary pmfs
   - File size = 120,198 bytes (11.42%)

4. Four neighbours: 16 binary pmfs
   - File size = 101,819 bytes (9.67%)

Original: 1,052,713 bytes

gzip: 183,705 bytes (17.45%)
bzip2: 169,049 bytes (16.05%)
lzip: 140,307 bytes (13.33%)

---

A Method for the Construction of Minimum-Redundancy Codes

DAVID A. HUFFMAN, ASSOCIATE, IEEE

Summary—An optimum method of coding an ensemble of messages consisting of a finite number of members is developed. A minimum-redundancy code is one constructed in such a way that the average number of coding digits per message is minimized.

Introduction

ONE IMPORTANT METHOD of transmitting messages is to transmit in their place sequences of symbols. If there are more messages which might be sent than there are kinds of symbols available, then some of the messages must use more than one symbol. If it is assumed that each symbol requires the same time for transmission, then the time for transmission (length) of a message is directly proportional to the number of symbols associated with it. In this paper, the symbol or sequence of symbols associated with a given message will be called the "message code." The entire number of messages which might be transmitted will be called the "message ensemble." The mutual agreement between the transmitter and the receiver about the meaning of the code for each message of the ensemble will be called the "ensemble code."

Probably the most familiar ensemble code was stated in the phrase "one if by land and two if by sea." In this

---

Heiko Schwarz (Freie Universität Berlin) — Data Compression: Predictive Lossless Coding
Example: Conditional Arithmetic Coding for Document Scans

1. No condition: 1 binary pmf
   ➞ File size = 418,369 bytes (39.74%)

2. Left neighbour: 2 binary pmfs
   ➞ File size = 192,841 bytes (18.32%)

3. Left and above: 4 binary pmfs
   ➞ File size = 120,198 bytes (11.42%)

4. Four neighbours: 16 binary pmfs
   ➞ File size = 101,819 bytes (9.67%)

5. Eleven neighbours: 2048 binary pmfs
   ➞ File size = 92,527 bytes (8.79%)

Original: 1,052,713 bytes

gzip: 183,705 bytes (17.45%)
bzip2: 169,049 bytes (16.05%)
lz: 140,307 bytes (13.33%)

A Method for the Construction of Minimum-Redundancy Codes*

DAVID A. HUFFMAN†, ASSOCIATE, IEEE

Summary—An optimum method of coding an ensemble of messages consisting of a finite number of members is developed. A minimum-redundancy code in each constructed in such a way that the average number of coding digits per message is minimized.

Introduction

One important method of transmitting messages is to transmit in their place sequences of symbols. If there are more messages which might be sent than there are kinds of symbols available, then some of the messages must use more than one symbol. If it is assumed that each symbol requires the same time for transmission, then the time for transmission (length) of a message is directly proportional to the number of symbols associated with it. In this paper, the symbol or sequence of symbols associated with a given message will be called the "message code." The entire number of messages which might be transmitted will be called the "message ensemble." The mutual agreement between the transmitter and the receiver about the meaning of the code for each message of the ensemble will be called the "ensemble code."

Probably the most familiar ensemble code was stated in the phrase "one if by land and two if by sea." In this will be defined here as an ensemble code which, for a message ensemble consisting of a finite number of members, \( N \), and for a given number of coding digits, \( D \), yields the lowest possible average message length. In order to avoid the use of the lengthy term "minimum-redundancy", this term will be replaced here by "optimum." It will be understood then that, in this paper, "optimum code" means "minimum-redundancy code."

The following basic restrictions will be imposed on an ensemble code:

(a) No two messages will consist of identical arrangements of coding digits.

(b) The message codes will be constructed in such a way that no additional indication is necessary to specify where a message code begins and ends once the starting point of a sequence of messages is known.

Restriction (b) necessitates that no message be coded in such a way that its code appears, digit for digit, as the first part of any message code of greater length. Thus, 01, 102, 114, and 202 are valid message codes for an ensemble of four members. For instance, a sequence of these messages 111002202010111102 can be broken up
Example: Conditional Arithmetic Coding for 8-bit Images

- **No condition**: 256 probability masses ($2^8$)
  - File size = 240112 bytes (91.59%)

- **Left sample**: 65536 probability masses ($2^8 \cdot 2^8$)
  - File size = 179179 bytes (68.35%)

- **Left sample and above sample**: 16777216 probability masses ($2^8 \cdot 2^8 \cdot 2^8$)
  - File size = 221849 bytes (84.62%)

Too many probability masses
Pmfs do not adapt to image statistics

Original: 262159 bytes (512 × 512)
- gzip: 222999 bytes (85.06%)
- bzip2: 173877 bytes (66.33%)
- lzip: 180000 bytes (68.66%)
Example: Conditional Arithmetic Coding for 8-bit Images

1. **No condition:**
   - 256 probability masses ($2^8$)
   - File size = 240,112 bytes (91.59%)

### File Sizes

<table>
<thead>
<tr>
<th>Method</th>
<th>Original Size</th>
<th>Compressed Size</th>
<th>Compression Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>262,159 bytes</td>
<td>(512 x 512)</td>
<td></td>
</tr>
<tr>
<td>gzip</td>
<td>222,999 bytes</td>
<td>(85.06%)</td>
<td></td>
</tr>
<tr>
<td>bzip2</td>
<td>173,877 bytes</td>
<td>(66.33%)</td>
<td></td>
</tr>
<tr>
<td>lzip</td>
<td>180,000 bytes</td>
<td>(68.66%)</td>
<td></td>
</tr>
</tbody>
</table>

Heiko Schwarz (Freie Universität Berlin) — Data Compression: Predictive Lossless Coding
Example: Conditional Arithmetic Coding for 8-bit Images

1. **No condition:**
   - 256 probability masses ($2^8$)
   - File size = 240,112 bytes (91.59 %)

2. **Left sample:**
   - 65,536 probability masses ($2^8 \cdot 2^8$)
   - File size = 179,179 bytes (68.35 %)

<table>
<thead>
<tr>
<th>Method</th>
<th>File Size</th>
<th>Compression (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>262,159 bytes</td>
<td>(512 × 512)</td>
</tr>
<tr>
<td>gzip</td>
<td>222,999 bytes</td>
<td>(85.06 %)</td>
</tr>
<tr>
<td>bzip2</td>
<td>173,877 bytes</td>
<td>(66.33 %)</td>
</tr>
<tr>
<td>lzip</td>
<td>180,000 bytes</td>
<td>(68.66 %)</td>
</tr>
</tbody>
</table>
Example: Conditional Arithmetic Coding for 8-bit Images

1. No condition:
   - 256 probability masses ($2^8$)
   - File size = 240 112 bytes (91.59%)

2. Left sample:
   - 65 536 probability masses ($2^8 \cdot 2^8$)
   - File size = 179 179 bytes (68.35%)

3. Left sample and above sample:
   - 16 777 216 probability masses ($2^8 \cdot 2^8 \cdot 2^8$)
   - File size = 221 849 bytes (84.62%)

Original: 262 159 bytes (512 × 512)
gzip: 222 999 bytes (85.06%)
bzip2: 173 877 bytes (66.33%)
lzip: 180 000 bytes (68.66%)
Example: Conditional Arithmetic Coding for 8-bit Images

1. No condition:
   - 256 probability masses \(2^8\)
   - File size = 240,112 bytes (91.59%)

2. Left sample:
   - 65,536 probability masses \(2^8 \cdot 2^8\)
   - File size = 179,179 bytes (68.35%)

3. Left sample and above sample:
   - 16,777,216 probability masses \(2^8 \cdot 2^8 \cdot 2^8\)
   - File size = 221,849 bytes (84.62%)

   ➔ Too many probability masses
   ➔ Pmfs do not adapt to image statistics

Original: 262,159 bytes \((512 \times 512)\)
gzip: 222,999 bytes (85.06%)
bzip2: 173,877 bytes (66.33%)
lzip: 180,000 bytes (68.66%)
Example: Conditional Arithmetic Coding for 16-bit Audio

Original: 26,559,960 bytes (5:01 minutes)
gzip: 24,926,843 bytes (93.85 %)
bzip2: 22,445,509 bytes (84.51 %)
lzip: 23,777,258 bytes (89.52 %)
Example: Conditional Arithmetic Coding for 16-bit Audio

1 No condition:

- 65,36 probability masses ($2^{16}$)
- File size = 23,700,606 bytes (89.23%)

Original: 26,559,960 bytes (5:01 minutes)

gzip: 24,926,843 bytes (93.85%)
bzip2: 22,445,509 bytes (84.51%)
lzzip: 23,777,258 bytes (89.52%)
Example: Conditional Arithmetic Coding for 16-bit Audio

1. No condition:
   - 65536 probability masses ($2^{16}$)
   - File size = 23,700,606 bytes (89.23%)

2. Previous sample:
   - 4,294,967,296 probability masses ($2^{16} \cdot 2^{16}$)

<table>
<thead>
<tr>
<th>Method</th>
<th>Original Size</th>
<th>Compressed Size</th>
<th>Compression (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>gzip</td>
<td>26,559,960 bytes</td>
<td>24,926,843 bytes</td>
<td>93.85%</td>
</tr>
<tr>
<td>bzip2</td>
<td>22,445,509 bytes</td>
<td>22,445,509 bytes</td>
<td>84.51%</td>
</tr>
<tr>
<td>lzip</td>
<td>23,777,258 bytes</td>
<td>23,777,258 bytes</td>
<td>89.52%</td>
</tr>
</tbody>
</table>
## Example: Conditional Arithmetic Coding for 16-bit Audio

1. **No condition:**
   - 65,536 probability masses \(2^{16}\)
   - File size = 23,700,606 bytes (89.23%)

2. **Previous sample:**
   - 4,294,967,296 probability masses \(2^{16} \cdot 2^{16}\)
   - Not possible on normal computers
   - Requires 16 GByte of memory
     - (when we use 32 bit per probability)
   - Would not adapt well to statistics

<table>
<thead>
<tr>
<th>Method</th>
<th>Original Size</th>
<th>Compressed Size</th>
<th>Compression Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>gzip</td>
<td>26,559,960</td>
<td>24,926,843</td>
<td>93.85%</td>
</tr>
<tr>
<td>bzip2</td>
<td>22,445,509</td>
<td>22,445,509</td>
<td>84.51%</td>
</tr>
<tr>
<td>lzip</td>
<td>23,777,258</td>
<td>23,777,258</td>
<td>89.52%</td>
</tr>
</tbody>
</table>
Example: Conditional Arithmetic Coding for 16-bit Audio

1. **No condition:**
   - 65,536 probability masses ($2^{16}$)
   - File size = 23,700,606 bytes (89.23%)

2. **Previous sample:**
   - 4,294,967,296 probability masses ($2^{16} \cdot 2^{16}$)
   - Not possible on normal computers
   - Requires 16 GByte of memory 
     (when we use 32 bit per probability)
   - Would not adapt well to statistics

→ Cannot exploit dependencies between symbols using this type of coding !!!
Analysis: Typical Properties of Real Signals

8-bit image data

\[ H = 7.45 \text{ (8-bit data)} \]

Conditional PMFs

Significantly narrower than marginal PMF (room for compression)

\[
\begin{align*}
\mathbf{p}(x|a) &= P(X_n = x | X_{n-1} = a) \\
\mathbf{p}(x|50) &= P(X_n = x | X_{n-1} = 50) \\
\mathbf{p}(x|128) &= P(X_n = x | X_{n-1} = 128) \\
\mathbf{p}(x|200) &= P(X_n = x | X_{n-1} = 200)
\end{align*}
\]
**Analysis: Typical Properties of Real Signals**

8-bit image data

**Marginal Pmf**

- Not much room for compression
- Entropy $H = 7.45$ (8-bit data)
Analysis: Typical Properties of Real Signals

8-bit image data

Marginal Pmf
- Not much room for compression
- Entropy $H = 7.45$ (8-bit data)

Conditional Pmfs
- Significantly narrower than marginal pmf (room for compression)
- Shape is very similar for all conditions, but shifted by $x_{n-1}$
Analysis: Typical Properties of Real Signals

8-bit image data

**Marginal Pmf**
- Not much room for compression
- Entropy $H = 7.45$ (8-bit data)

**Conditional Pmfs**
- Significantly narrower than marginal pmf (room for compression)
- Shape is very similar for all conditions, but shifted by $x_{n-1}$

→ **Idea: Code difference to previous sample !!!**
Simple Predictive Coding using Preceding Sample

**Encoding**
- Generate difference sample
  \[ u_n = s_n - s_{n-1} \]
- Encode difference sample \( u_n \)

**Diagram**
- Input: \( s_n \) and \( s_{n-1} \)
- Output: \( u_n \) and encoder output
- Encoder: \( u_n \) and \( s_{n-1} \)
- Decoder: \( u_n \) and \( s_n \)
- Output: \( s_n \)

**Basic Effect**
- Prediction removes large part of inter-symbol dependencies before actual entropy coding
- Reduces required complexity for the entropy coding (e.g., marginal instead of conditional coding)
Simple Predictive Coding using Preceding Sample

**Encoding**
- Generate difference sample
  \[ u_n = s_n - s_{n-1} \]
- Encode difference sample \( u_n \)

**Decoding**
- Decode difference sample \( u_n \)
- Reconstruct original sample
  \[ s_n = s_{n-1} + u_n \]
Simple Predictive Coding using Preceding Sample

Encoding
- Generate difference sample
  \[ u_n = s_n - s_{n-1} \]
- Encode difference sample \( u_n \)

Decoding
- Decode difference sample \( u_n \)
- Reconstruct original sample
  \[ s_n = s_{n-1} + u_n \]

Basic Effect
- Prediction removes large part of inter-symbol dependencies before actual entropy coding
- Reduces required complexity for the entropy coding (e.g., marginal instead of conditional coding)
How Well Does That Type of Prediction Work?

\[ H = 7.45 \]

\[ H = 5.05 \]

\[ H = 14.25 \]

\[ H = 12.39 \]

saves nearly 2 bits per sample!

White Gaussian Noise

\[ H = 6.95 \]

\[ H = 7.45 \]

prediction increases entropy!
How Well Does That Type of Prediction Work?

H = 7.45

H = 5.05

H = 14.25

H = 12.39

saves nearly 2 bits per sample!
How Well Does That Type of Prediction Work?

\[ H = 7.45 \]

\[ u_n = s_n - s_{n-1} \]

\[ H = 5.05 \]

\[ H = 14.25 \]

Saves nearly 2 bits per sample!
How Well Does That Type of Prediction Work?

\[ H = 7.45 \]

\[ u_n = s_n - s_{n-1} \]

\[ H = 5.05 \]

\[ H = 14.25 \]

\[ u_n = s_n - s_{n-1} \]

\[ H = 12.39 \]

saves nearly 2 bits per sample!
How Well Does That Type of Prediction Work?

\[ H = 7.45 \]

\[ u_n = s_n - s_{n-1} \]

\[ H = 5.05 \]

\[ H = 14.25 \]

\[ u_n = s_n - s_{n-1} \]

\[ H = 12.39 \]

saves nearly 2 bits per sample!

White Gaussian Noise

\[ H = 6.95 \]
How Well Does That Type of Prediction Work?

- **Predictive Coding / Motivation**

  - For the given type of prediction, the entropies are calculated as follows:
    - \( H = 7.45 \) for \( p_s(x) \)
    - \( H = 5.05 \) for \( p_u(x) \)

  - For another type of prediction:
    - \( H = 14.25 \) for \( p_s(x) \)
    - \( H = 12.39 \) for \( p_u(x) \)

- **White Gaussian Noise**

  - \( H = 6.95 \) for \( p_s(x) \)
  - \( H = 7.45 \) for \( p_u(x) \)

- The prediction decreases the entropy by nearly 2 bits per sample.

- The entropy increases when using the prediction.
Predictive Lossless Coding

- Predict current sample $s_n$ using a function of preceding samples
  \[ \hat{s}_n = f(s_{n-1}, s_{n-2}, \cdots) \]

- Entropy coding of prediction error samples
  \[ u_n = s_n - \hat{s}_n \]

- Decoder uses exactly the same prediction and reconstructs the original samples according to
  \[ s_n = \hat{s}_n + u_n \]
Choice of Observation Set and Predictor

\[ S_n \rightarrow \text{prediction} \rightarrow \hat{S}_n \rightarrow U_n \]

**Choice of Observation Set** \( B_n \)

- Use small number of preceding samples for prediction
- Choose the samples \( B_n \) with highest dependencies to current sample \( s_n \)

For 1D signals (e.g., audio): \( N \) directly preceding samples

\[ B_n = \{ s_{n-1}, s_{n-2}, \ldots, s_{n-N} \} \]

For 2D signals (e.g., images): Samples in causal direct neighborhood
Choice of Observation Set $\mathcal{B}_n$

- Use small number of preceding samples for prediction
- Choose the samples $\mathcal{B}_n$ with highest dependencies to current sample $s_n$
  - 1D signals (e.g., audio): $N$ directly preceding samples $\rightarrow \mathcal{B}_n = \{s_{n-1}, s_{n-2}, \cdots, s_{n-N}\}$

1D signals: 

![1D Signal Example](image-url)
Choice of Observation Set $B_n$

- Use small number of preceding samples for prediction
- Choose the samples $B_n$ with highest dependencies to current sample $s_n$
  - 1D signals (e.g., audio): $N$ directly preceding samples  $\rightarrow$  $B_n = \{s_{n-1}, s_{n-2}, \cdots, s_{n-N}\}$
  - 2D signals (e.g., images): Samples in causal direct neighborhood

1D signals: 

2D signals:
Choice of Observation Set and Predictor

**Choice of Observation Set** $B_n$

- Use small number of preceding samples for prediction
- Choose the samples $B_n$ with highest dependencies to current sample $s_n$
  - 1D signals (e.g., audio): N directly preceding samples $\rightarrow B_n = \{s_{n-1}, s_{n-2}, \cdots, s_{n-N}\}$
  - 2D signals (e.g., images): Samples in causal direct neighborhood

**Choice of Predictor for given Observation Set**

- Question: What function $\hat{s}_n = f(B_n)$ should we use for prediction?
Choice of Observation Set $\mathcal{B}_n$

- Use small number of preceding samples for prediction
- Choose the samples $\mathcal{B}_n$ with highest dependencies to current sample $s_n$
  - 1D signals (e.g., audio): $N$ directly preceding samples $\rightarrow \mathcal{B}_n = \{s_{n-1}, s_{n-2}, \cdots, s_{n-N}\}$
  - 2D signals (e.g., images): Samples in causal direct neighborhood

Choice of Predictor for given Observation Set

- Question: What function $\hat{s}_n = f(\mathcal{B}_n)$ should we use for prediction?
- Want to minimize entropy $H(U_n)$ of prediction error samples $\{u_n\}$ (difficult to access directly)
Choice of Observation Set $B_n$

- Use small number of preceding samples for prediction
  - Choose the samples $B_n$ with highest dependencies to current sample $s_n$
    - 1D signals (e.g., audio): $N$ directly preceding samples $\rightarrow B_n = \{s_{n-1}, s_{n-2}, \cdots, s_{n-N}\}$
    - 2D signals (e.g., images): Samples in causal direct neighborhood

Choice of Predictor for given Observation Set

- Question: What function $\hat{s}_n = f(B_n)$ should we use for prediction?
- Want to minimize entropy $H(U_n)$ of prediction error samples $\{u_n\}$ (difficult to access directly)
  - Can minimize variance $\sigma^2_{U}$ and ignore shape of pmf $p(u)$
Optimization Criterion for Deriving Predictor

Typical Optimization Criterion

- Minimize Energy of Prediction Error Signal

\[ \varepsilon_U^2 = \mathbb{E}\{ U_n^2 \} = \mathbb{E}\left\{ \left( S_n - \hat{S}_n \right)^2 \right\} = \mathbb{E}\left\{ \left( S_n - f(B_n) \right)^2 \right\} \]
Optimization Criterion for Deriving Predictor

Typical Optimization Criterion

- Minimize Energy of Prediction Error Signal

\[ \varepsilon_{U}^2 = E\{U_n^2\} = E\left\{ (S_n - \hat{S}_n)^2 \right\} = E\left\{ (S_n - f(B_n))^2 \right\} \]

- Reformulate prediction error energy

\[
\varepsilon_{U}^2 = E\{U_n^2\} = E\left\{ (U_n - \mu_U + \mu_u)^2 \right\} \\
= E\left\{ (U_n - \mu_U)^2 \right\} + \mu_u^2 + 2 \mu_U E\{U_n - \mu_U\} \\
= \sigma_u^2 + \mu_u^2
\]
Optimization Criterion for Deriving Predictor

Typical Optimization Criterion

- Minimize Energy of Prediction Error Signal

\[ \varepsilon^2_U = \mathbb{E}\{ U_n^2 \} = \mathbb{E}\left\{ (S_n - \hat{S}_n)^2 \right\} = \mathbb{E}\left\{ (S_n - f(B_n))^2 \right\} \]

- Reformulate prediction error energy

\[
\varepsilon^2_U = \mathbb{E}\{ U_n^2 \} = \mathbb{E}\left\{ (U_n - \mu_U + \mu_u)^2 \right\} \\
= \mathbb{E}\left\{ (U_n - \mu_U)^2 \right\} + \mu_U^2 + 2 \mu_U \mathbb{E}\{ U_n - \mu_U \} \\
= \sigma_U^2 + \mu_U^2
\]

⇒ Minimization of squared prediction error \( \varepsilon^2_U \) implies minimization of variance \( \sigma_U^2 \) and mean \( \mu_U^2 \)
Optimization Criterion for Deriving Predictor

Typical Optimization Criterion

- Minimize Energy of Prediction Error Signal

\[
\varepsilon^2_U = E\left\{ U_n^2 \right\} = E\left\{ (S_n - \hat{S}_n)^2 \right\} = E\left\{ (S_n - f(\mathcal{B}_n))^2 \right\}
\]

- Reformulate prediction error energy

\[
\varepsilon^2_U = E\left\{ U_n^2 \right\} = E\left\{ (U_n - \mu_U + \mu_u)^2 \right\} = E\left\{ (U_n - \mu_U)^2 \right\} + \mu_u^2 + 2\mu_U E\left\{ U_n - \mu_U \right\}
\]

\[
= \sigma_U^2 + \mu_u^2
\]

⇒ Minimization of squared prediction error \( \varepsilon^2_U \) implies minimization of variance \( \sigma_U^2 \) and mean \( \mu_u^2 \)

⇒ Minimize the width of the pmf \( p_U(u) \), but ignore its actual shape
Optimization Criterion for Deriving Predictor

Typical Optimization Criterion

- Minimize Energy of Prediction Error Signal

\[ \varepsilon_U^2 = \mathbb{E}\{ U_n^2 \} = \mathbb{E}\left\{ \left( S_n - \hat{S}_n \right)^2 \right\} = \mathbb{E}\left\{ \left( S_n - f(B_n) \right)^2 \right\} \]

- Reformulate prediction error energy

\[ \varepsilon_U^2 = \mathbb{E}\{ U_n^2 \} = \mathbb{E}\left\{ (U_n - \mu_U + \mu_U)^2 \right\} = \mathbb{E}\left\{ (U_n - \mu_U)^2 \right\} + 2\mu_U \mathbb{E}\{ U_n - \mu_U \} = \sigma_U^2 + \mu_U^2 \]

- Minimization of squared prediction error \( \varepsilon_U^2 \) implies minimization of variance \( \sigma_U^2 \) and mean \( \mu_U^2 \)

- Minimize the width of the pmf \( p_{U}(u) \), but ignore its actual shape

- Suitable alternative to minimization of the entropy \( H(U) \)
Question: What value \( a \) minimizes the prediction error energy ?

\[
\varepsilon^2_U = \mathbb{E}\{(S_n - a)^2\}
\]
**Optimal Predictor for Given Observation Set**

- Question: What value $a$ minimizes the prediction error energy?

\[ \varepsilon_U^2 = E\{(S_n - a)^2\} = E\{(S_n - E\{S_n\} + E\{S_n\} - a)^2\} \]
Optimal Predictor for Given Observation Set

Question: What value $a$ minimizes the prediction error energy?

\[
\epsilon_u^2 = E\{(S_n - a)^2\} = E\{(S_n - E\{S_n\} + E\{S_n\} - a)^2\} \\
= E\{(S_n - E\{S_n\})^2\} + E\{(E\{S_n\} - a)^2\} + 2E\{(S_n - E\{S_n\})(E\{S_n\} - a)\}
\]
Optimal Predictor for Given Observation Set

Question: What value $a$ minimizes the prediction error energy?

$$\varepsilon^2_{\hat{U}} = E\{ (S_n - a)^2 \} = E\{ (S_n - E\{ S_n \} + E\{ S_n \} - a)^2 \}$$

$$= E\{ (S_n - E\{ S_n \})^2 \} + E\{ (E\{ S_n \} - a)^2 \} + 2 E\left\{ (S_n - E\{ S_n \})(E\{ S_n \} - a) \right\}$$

$$= \sigma_S^2 + (E\{ S_n \} - a)^2 \geq 0$$
Question: What value $a$ minimizes the prediction error energy?

$$
\varepsilon^2_U = \mathbb{E}\{(S_n - a)^2\} = \mathbb{E}\{(S_n - \mathbb{E}\{S_n\} + \mathbb{E}\{S_n\} - a)^2\}
$$

$$
= \mathbb{E}\{(S_n - \mathbb{E}\{S_n\})^2\} + \mathbb{E}\{(\mathbb{E}\{S_n\} - a)^2\} + 2\mathbb{E}\{(S_n - \mathbb{E}\{S_n\})(\mathbb{E}\{S_n\} - a)\}
$$

$$
= \sigma^2_S + (\mathbb{E}\{S_n\} - a)^2 \geq 0
$$

Prediction error energy is minimized by mean, i.e., by setting $a = \mathbb{E}\{S_n\}$
Optimal Predictor for Given Observation Set

- Question: What value $a$ minimizes the prediction error energy?

\[
\varepsilon^2_U = \mathbb{E}\{(S_n - a)^2\} = \mathbb{E}\{(S_n - \mathbb{E}\{S_n\} + \mathbb{E}\{S_n\} - a)^2\}
\]

\[
= \mathbb{E}\{(S_n - \mathbb{E}\{S_n\})^2\} + \mathbb{E}\{(\mathbb{E}\{S_n\} - a)^2\} + 2\mathbb{E}\{(S_n - \mathbb{E}\{S_n\})(\mathbb{E}\{S_n\} - a)\}
\]

\[
= \sigma_S^2 + (\mathbb{E}\{S_n\} - a)^2 \geq 0
\]

→ Prediction error energy is minimized by mean, i.e., by setting $a = \mathbb{E}\{S_n\}$

Minimization of mean-squared prediction error for given observation set $\mathcal{B}_n$

- Similar optimization problem: Minimize $\mathbb{E}\{(S_n - f(\mathcal{B}_n))^2 \mid \mathcal{B}_n\}$
Question: What value $a$ minimizes the prediction error energy?

\[ \varepsilon^2_U = \mathbb{E}\{ (S_n - a)^2 \} = \mathbb{E}\{ (S_n - \mathbb{E}\{ S_n \}) + \mathbb{E}\{ S_n \} - a)^2 \} \]

\[ = \mathbb{E}\{ (S_n - \mathbb{E}\{ S_n \})^2 \} + \mathbb{E}\{ (\mathbb{E}\{ S_n \} - a)^2 \} + 2 \mathbb{E}\{ (S_n - \mathbb{E}\{ S_n \}) (\mathbb{E}\{ S_n \} - a) \} \]

\[ = \sigma_S^2 + (\mathbb{E}\{ S_n \} - a)^2 \geq 0 \]

Prediction error energy is minimized by mean, i.e., by setting $a = \mathbb{E}\{ S_n \}$

Minimization of mean-squared prediction error for given observation set $B_n$

Similar optimization problem: Minimize $\mathbb{E}\{ (S_n - f(B_n))^2 \mid B_n \}$

Solution: Optimal predictor is given by the conditional mean

\[ \hat{s}_n = f_{\text{opt}}(B_n) = \mathbb{E}\{ S_n \mid B_n \} \]
Optimal Predictor for Given Observation Set

Question: What value $a$ minimizes the prediction error energy?

$$
e_{U}^2 = E \{ (S_n - a)^2 \} = E \{ (S_n - E \{ S_n \} + E \{ S_n \} - a)^2 \}
$$

$$= E \{ (S_n - E \{ S_n \})^2 \} + E \{ (E \{ S_n \} - a)^2 \} + 2E \{ (S_n - E \{ S_n \})(E \{ S_n \} - a) \}
$$

$$= \sigma_S^2 + (E \{ S_n \} - a)^2 \geq 0$$

⇒ Prediction error energy is minimized by mean, i.e., by setting $a = E \{ S_n \}$

Minimization of mean-squared prediction error for given observation set $B_n$

Similar optimization problem: Minimize $E \{ (S_n - f(B_n))^2 \mid B_n \}$

⇒ Solution: Optimal predictor is given by the conditional mean

$$\hat{s}_n = f_{opt}(B_n) = E \{ S_n \mid B_n \}$$

⇒ General case requires storage of large tables (often impractical and complex)
Optimal Predictor for Autoregressive Sources

Autoregressive Sources of Order $m$

- Good probabilistic model for many real signals: **AR($m$) process**

$$S_n = \mu_s + \sum_{k=1}^{m} a_k \cdot (S_{n-k} - \mu_s) + Z_n$$  \hspace{1cm} (Z_n is zero-mean iid process)
Optimal Predictor for Autoregressive Sources

**Autoregressive Sources of Order $m$**

- Good probabilistic model for many real signals: **AR($m$) process**

\[
S_n = \mu_s + \sum_{k=1}^{m} a_k \cdot (S_{n-k} - \mu_s) + Z_n \quad (Z_n \text{ is zero-mean iid process})
\]

- Optimal predictor if we know the entire past, i.e., $\mathcal{B}_n = \{s_{n-1}, s_{n-2}, \cdots\}$

\[
E\{ S_n \mid \mathcal{B}_n \}
\]
Optimal Predictor for Autoregressive Sources

Autoregressive Sources of Order $m$

- Good probabilistic model for many real signals: $\text{AR}(m)$ process

\[ S_n = \mu_s + \sum_{k=1}^{m} a_k \cdot (S_{n-k} - \mu_s) + Z_n \quad (Z_n \text{ is zero-mean iid process}) \]

⇒ Optimal predictor if we know the entire past, i.e., $\mathcal{B}_n = \{s_{n-1}, s_{n-2}, \cdots\}$

\[ \mathbb{E}\{ S_n \mid \mathcal{B}_n \} = \mathbb{E}\left\{ \mu_s + \sum_{k=1}^{m} a_k \cdot (S_{n-k} - \mu_s) + Z_n \mid s_{n-1}, s_{n-2}, \cdots \right\} \]
Optimal Predictor for Autoregressive Sources

**Autoregressive Sources of Order $m$**

- Good probabilistic model for many real signals: **AR($m$) process**

\[
S_n = \mu_s + \sum_{k=1}^{m} a_k \cdot (S_{n-k} - \mu_s) + Z_n \quad \text{($Z_n$ is zero-mean iid process)}
\]

→ Optimal predictor if we know the entire past, i.e., $B_n = \{s_{n-1}, s_{n-2}, \cdots\}$

\[
E\{ S_n \mid B_n \} = E\left\{ \mu_s + \sum_{k=1}^{m} a_k \cdot (S_{n-k} - \mu_s) + Z_n \mid s_{n-1}, s_{n-2}, \cdots \right\}
\]

\[
= \mu_s \left(1 - \sum_{k=1}^{m} a_k\right) + \sum_{k=1}^{m} a_k \cdot E\{ S_{n-k} \mid s_{n-1}, s_{n-2}, \cdots \} + E\{ Z_n \mid s_{n-1}, s_{n-2}, \cdots \}
\]
Optimal Predictor for Autoregressive Sources

Autoregressive Sources of Order $m$

- Good probabilistic model for many real signals: AR($m$) process

$$S_n = \mu_s + \sum_{k=1}^{m} a_k \cdot (S_{n-k} - \mu_s) + Z_n \quad (Z_n \text{ is zero-mean iid process})$$

Optimal predictor if we know the entire past, i.e., $\mathcal{B}_n = \{s_{n-1}, s_{n-2}, \cdots\}$

$$E\{ S_n \mid \mathcal{B}_n \} = E\left\{ \mu_s + \sum_{k=1}^{m} a_k \cdot (S_{n-k} - \mu_s) + Z_n \middle| s_{n-1}, s_{n-2}, \cdots \right\}$$

$$= \mu_s \left( 1 - \sum_{k=1}^{m} a_k \right) + \sum_{k=1}^{m} a_k \cdot E\{ S_{n-k} \mid s_{n-1}, s_{n-2}, \cdots \} + E\{ Z_n \mid s_{n-1}, s_{n-2}, \cdots \}$$

$$= a_0 + \sum_{k=1}^{m} a_k \cdot s_{n-k} \quad \text{with} \quad a_0 = \mu_s \left( 1 - \sum_{k=1}^{m} a_k \right)$$
Optimal Predictor for Autoregressive Sources

Autoregressive Sources of Order $m$

- Good probabilistic model for many real signals: \textbf{AR}(m) process

\[ S_n = \mu_s + \sum_{k=1}^{m} a_k \cdot (S_{n-k} - \mu_s) + Z_n \] (\(Z_n\) is zero-mean iid process)

\[ \Rightarrow \text{Optimal predictor if we know the entire past, i.e., } B_n = \{s_{n-1}, s_{n-2}, \cdots\} \]

\[
E\{S_n | B_n\} = E\left\{\mu_s + \sum_{k=1}^{m} a_k \cdot (S_{n-k} - \mu_s) + Z_n | s_{n-1}, s_{n-2}, \cdots\right\}
\]

\[= \mu_s \left(1 - \sum_{k=1}^{m} a_k\right) + \sum_{k=1}^{m} a_k \cdot E\{S_{n-k} | s_{n-1}, s_{n-2}, \cdots\} + E\{Z_n | s_{n-1}, s_{n-2}, \cdots\}\]

\[= a_0 + \sum_{k=1}^{m} a_k \cdot s_{n-k} \quad \text{with} \quad a_0 = \mu_s \left(1 - \sum_{k=1}^{m} a_k\right)\]

\[\Rightarrow \text{Optimal predictor is an affine function of the past } m \text{ samples}\]
Variance and Mean for Affine Prediction

Affine Predictor:  
\[ \hat{s}_n = a_0 + \sum_{k=1}^{K} a_k \cdot b_k \]
for any observation set  
\[ B_n = \{ b_1, b_2, \cdots, b_K \} \]
Variance and Mean for Affine Prediction

Affine Predictor: 
\[ \hat{s}_n = a_0 + \sum_{k=1}^{K} a_k \cdot b_k \]

for any observation set \( B_n = \{b_1, b_2, \cdots , b_K\} \)

- Mean \( \mu_U \) of prediction error
  \[ \mu^2_U = E\{ U_n \} \]
Variance and Mean for Affine Prediction

Affine Predictor: \( \hat{s}_n = a_0 + \sum_{k=1}^{K} a_k \cdot b_k \)

for any observation set \( B_n = \{ b_1, b_2, \cdots, b_K \} \)

- **Mean** \( \mu_U \) of prediction error

\[
\mu_U^2 = E\{ U_n \} = E\left\{ S_n - a_0 - \sum_{k=1}^{K} a_k B_k \right\}
\]
Affine Predictor: \[ \hat{s}_n = a_0 + \sum_{k=1}^{K} a_k \cdot b_k \] for any observation set \( B_n = \{ b_1, b_2, \cdots, b_K \} \)

Mean \( \mu_U \) of prediction error
\[
\mu_U^2 = E\{ U_n \} = E\left\{ S_n - a_0 - \sum_{k=1}^{K} a_k B_k \right\} \\
= \mu_s \left( 1 - \sum_{k=1}^{K} a_k \right) - a_0
\]
### Variance and Mean for Affine Prediction

**Affine Predictor:**

\[
\hat{s}_n = a_0 + \sum_{k=1}^{K} a_k \cdot b_k
\]

for any observation set \( B_n = \{b_1, b_2, \ldots, b_K\} \)

**Mean \( \mu_U \) of prediction error**

\[
\mu_U^2 = \mathbb{E}\{U_n\} = \mathbb{E}\{S_n - a_0 - \sum_{k=1}^{K} a_k \cdot B_k\}
\]

\[
= \mu_S \left(1 - \sum_{k=1}^{K} a_k\right) - a_0
\]

Can be forced to \( \mu_U = 0 \) by setting \( a_0 = \mu_S \left(1 - \sum_{k=1}^{K} a_k\right) \)
Variance and Mean for Affine Prediction

Affine Predictor:  \[ \hat{s}_n = a_0 + \sum_{k=1}^{K} a_k \cdot b_k \]
for any observation set  \( B_n = \{ b_1, b_2, \cdots, b_K \} \)

Mean \( \mu_U \) of prediction error
\[ \mu_U^2 = \mathbb{E}\{ U_n \} = \mathbb{E}\left\{ S_n - a_0 - \sum_{k=1}^{K} a_k B_k \right\} \]
\[ = \mu_S \left( 1 - \sum_{k=1}^{K} a_k \right) - a_0 \]

\( \Rightarrow \) Can be forced to \( \mu_U = 0 \) by setting \( a_0 = \mu_S \left( 1 - \sum_{k=1}^{K} a_k \right) \)

Variance \( \sigma_U^2 \) of predictor error does not depend on constant offset \( a_0 \)
\[ \sigma_U^2 = \mathbb{E}\{ (U_n - \mathbb{E}\{ U_n \})^2 \} \]
Variance and Mean for Affine Prediction

Affine Predictor: \[ \hat{s}_n = a_0 + \sum_{k=1}^{K} a_k \cdot b_k \]
for any observation set \( B_n = \{ b_1, b_2, \cdots, b_K \} \)

- **Mean** \( \mu_U \) of prediction error
  \[
  \mu_U^2 = E\{ U_n \} = E\left\{ S_n - a_0 - \sum_{k=1}^{K} a_k B_k \right\} \\
  = \mu_S \left( 1 - \sum_{k=1}^{K} a_k \right) - a_0
  \]
  ➡️ Can be forced to \( \mu_U = 0 \) by setting \( a_0 = \mu_S \left( 1 - \sum_{k=1}^{K} a_k \right) \)

- **Variance** \( \sigma_U^2 \) of predictor error does not depend on constant offset \( a_0 \)
  \[
  \sigma_U^2 = E\{ (U_n - E\{ U_n \})^2 \} = E\left\{ \left( S_n - a_0 - \sum_{k=1}^{K} a_k B_k - E\{ S_n - a_0 - \sum_{k=1}^{K} a_k B_k \} \right)^2 \right\}
  \]
**Variance and Mean for Affine Prediction**

**Affine Predictor:**

\[
\hat{s}_n = a_0 + \sum_{k=1}^{K} a_k \cdot b_k
\]

for any observation set \( B_n = \{ b_1, b_2, \ldots, b_K \} \)

- **Mean \( \mu_U \) of prediction error**

\[
\mu_U^2 = E\{ U_n \} = E\left\{ S_n - a_0 - \sum_{k=1}^{K} a_k B_k \right\} \\
= \mu_S \left( 1 - \sum_{k=1}^{K} a_k \right) - a_0
\]

\( \Rightarrow \) Can be forced to \( \mu_U = 0 \) by setting \( a_0 = \mu_S \left( 1 - \sum_{k=1}^{K} a_k \right) \)

- **Variance \( \sigma_U^2 \) of predictor error** does not depend on constant offset \( a_0 \)

\[
\sigma_U^2 = E\left\{ (U_n - E\{ U_n \})^2 \right\} = E\left\{ \left( S_n - a_0 - \sum_{k=1}^{K} a_k B_k - E\{ S_n - a_0 - \sum_{k=1}^{K} a_k B_k \} \right)^2 \right\} \\
= E\left\{ \left( S_n - \sum_{k=1}^{K} a_k B_k \right) - \mu_S \left( 1 - \sum_{k=1}^{K} a_k \right) \}^2 \right\} \neq f(a_0)
Affine and Linear Prediction

Affine Predictor: \[ \hat{s}_n = a_0 + \sum_{k=1}^{K} a_k \cdot b_k \]

for any observation set \( B_n = \{b_1, b_2, \cdots, b_K\} \)
Affine and Linear Prediction

Affine Predictor: \[ \hat{s}_n = a_0 + \sum_{k=1}^{K} a_k \cdot b_k \] for any observation set \( B_n = \{b_1, b_2, \cdots, b_K\} \)

For a minimization of the prediction error variance \( \sigma_U^2 \), a linear predictor is sufficient

\[ \hat{s}_n = \sum_{k=1}^{K} a_k \cdot b_k \]
Affine and Linear Prediction

Affine Predictor: 
\[ \hat{s}_n = a_0 + \sum_{k=1}^{K} a_k \cdot b_k \]
for any observation set \( B_n = \{b_1, b_2, \ldots, b_K\} \)

→ For a minimization of the prediction error variance \( \sigma_U^2 \), a linear predictor is sufficient

\[ \hat{s}_n = \sum_{k=1}^{K} a_k \cdot b_k \]

→ The mean \( \mu_U \) of the prediction error can be forced to zero by additionally setting

\[ a_0 = \mu_S \left( 1 - \sum_{k=1}^{K} a_k \right) \]
Affine and Linear Prediction

Affine Predictor:
\[ \hat{s}_n = a_0 + \sum_{k=1}^{K} a_k \cdot b_k \]
for any observation set \( B_n = \{b_1, b_2, \cdots, b_K\} \)

For a minimization of the prediction error variance \( \sigma_U^2 \), a linear predictor is sufficient

\[ \hat{s}_n = \sum_{k=1}^{K} a_k \cdot b_k \]

The mean \( \mu_U \) of the prediction error can be forced to zero by additionally setting

\[ a_0 = \mu_S \left( 1 - \sum_{k=1}^{K} a_k \right) \]

How can we derive optimal prediction parameters \( \{a_1, a_2, \cdots, a_K\} \)
for a given observation set and a given source?
Consider: Linear Prediction and Minimization of Variance

**Vector Notation** (for simplifying derivations)

- Observation set: \( B_n = (B_1, B_2, \cdots, B_K)^T \)
Consider: Linear Prediction and Minimization of Variance

**Vector Notation** (for simplifying derivations)

- Observation set: \( B_n = (B_1, B_2, \cdots, B_K)^T \)
- Prediction parameters: \( a = (a_1, a_2, \cdots, a_K)^T \)
Consider: Linear Prediction and Minimization of Variance

**Vector Notation** (for simplifying derivations)

- Observation set: \( \mathbf{B}_n = (B_1, B_2, \cdots, B_K)^T \)
- Prediction parameters: \( \mathbf{a} = (a_1, a_2, \cdots, a_K)^T \)
- Linear predictor: \( \hat{S}_n = \mathbf{a}^T \cdot \mathbf{B}_n \)
Consider: Linear Prediction and Minimization of Variance

**Vector Notation** (for simplifying derivations)

- Observation set: \( B_n = (B_1, B_2, \cdots, B_K)^T \)
- Prediction parameters: \( a = (a_1, a_2, \cdots, a_K)^T \)

\( \hat{S}_n = a^T \cdot B_n \)

\( U_n = S_n - \hat{S}_n = S_n - a^T B_n \)
Consider: Linear Prediction and Minimization of Variance

Vector Notation (for simplifying derivations)

- Observation set: \( \mathbf{B}_n = (B_1, B_2, \ldots, B_K)^T \)
- Prediction parameters: \( \mathbf{a} = (a_1, a_2, \ldots, a_K)^T \)
- Linear predictor: \( \hat{S}_n = \mathbf{a}^T \cdot \mathbf{B}_n \)
- Predictor error: \( U_n = S_n - \hat{S}_n = S_n - \mathbf{a}^T \mathbf{B}_n \)

Optimization Problem

- Minimization of prediction error variance

\[
\sigma^2_U(\mathbf{a}) = \mathbb{E}\left\{ \left( U_n - \mathbb{E}\{ U_n \} \right)^2 \right\}
\]
Consider: Linear Prediction and Minimization of Variance

**Vector Notation** (for simplifying derivations)

- Observation set: \( B_n = (B_1, B_2, \cdots, B_K)^T \)
- Prediction parameters: \( a = (a_1, a_2, \cdots, a_K)^T \)

- Linear predictor: \( \hat{S}_n = a^T \cdot B_n \)
- Predictor error: \( U_n = S_n - \hat{S}_n = S_n - a^T B_n \)

**Optimization Problem**

- Minimization of prediction error variance

\[
\sigma^2_U(a) = E \left\{ \left( U_n - E\{ U_n \} \right)^2 \right\} = E \left\{ \left( S_n - a^T B_n - E\{ S_n - a^T B_n \} \right)^2 \right\}
\]
Consider: Linear Prediction and Minimization of Variance

**Vector Notation** (for simplifying derivations)

- Observation set: \( \mathbf{B}_n = (B_1, B_2, \cdots, B_K)^T \)
- Prediction parameters: \( \mathbf{a} = (a_1, a_2, \cdots, a_K)^T \)
- Linear predictor: \( \hat{S}_n = \mathbf{a}^T \cdot \mathbf{B}_n \)
- Predictor error: \( U_n = S_n - \hat{S}_n = S_n - \mathbf{a}^T \mathbf{B}_n \)

**Optimization Problem**

- Minimization of prediction error variance

\[
\sigma_U^2(\mathbf{a}) = \mathbb{E} \left\{ \left( U_n - \mathbb{E} \{ U_n \} \right)^2 \right\} = \mathbb{E} \left\{ \left( S_n - \mathbf{a}^T \mathbf{B}_n - \mathbb{E} \{ S_n - \mathbf{a}^T \mathbf{B}_n \} \right)^2 \right\} \\
= \mathbb{E} \left\{ \left( \left( S_n - \mathbb{E} \{ S_n \} \right) - \mathbf{a}^T \left( \mathbf{B}_n - \mathbb{E} \{ \mathbf{B}_n \} \right) \right)^2 \right\}
\]
Reformulate Prediction Error Variance

- Prediction error variance

\[ \sigma^2_U(a) = E \left\{ \left( S_n - E\{ S_n \} \right) - a^T \left( B_n - E\{ B_n \} \right) \right\}^2 \]
Reformulate Prediction Error Variance

- Prediction error variance

\[
\sigma^2_U(a) = \mathbb{E}\left\{ \left( \left( S_n - \mathbb{E}\{ S_n \} \right) - a^T \left( B_n - \mathbb{E}\{ B_n \} \right) \right)^2 \right\}
\]

\[
= \mathbb{E}\left\{ \left( S_n - \mathbb{E}\{ S_n \} \right)^2 \right\} - 2 a^T \cdot \mathbb{E}\left\{ \left( S_n - \mathbb{E}\{ S_n \} \right) \left( B_n - \mathbb{E}\{ B_n \} \right) \right\}
\]

\[
+ a^T \cdot \mathbb{E}\left\{ \left( B_n - \mathbb{E}\{ B_n \} \right) \left( B_n - \mathbb{E}\{ B_n \} \right)^T \right\} \cdot a
\]
Reformulate Prediction Error Variance

- Prediction error variance

\[
\sigma^2_U(a) = E \left\{ \left( \left( S_n - E\{S_n\} \right) - a^T \left( B_n - E\{B_n\} \right) \right)^2 \right\} \\
= E \left\{ \left( S_n - E\{S_n\} \right)^2 \right\} - 2 a^T \cdot E \left\{ \left( S_n - E\{S_n\} \right) \left( B_n - E\{B_n\} \right) \right\} \\
+ a^T \cdot E \left\{ \left( B_n - E\{B_n\} \right) \left( B_n - E\{B_n\} \right)^T \right\} \cdot a \\
= \sigma^2_S - 2 a^T c + a^T C_B a
\]

with \( C_B \) and \( c \) being given by

\[
C_B = E \left\{ \left( B_n - E\{B_n\} \right) \left( B_n - E\{B_n\} \right)^T \right\} \\
c = E \left\{ \left( S_n - E\{S_n\} \right) \left( B_n - E\{B_n\} \right) \right\} \]
Auto-Covariance Matrix of the Observation Set

\[ C_B = E\left\{ \left( B_n - E\{ B_n \} \right) \left( B_n - E\{ B_n \} \right)^T \right\} = \sigma_S^2 \cdot \begin{bmatrix} 1 & \varrho_{1,2} & \varrho_{1,3} & \cdots & \varrho_{1,K} \\ \varrho_{2,1} & 1 & \varrho_{2,3} & \cdots & \varrho_{2,K} \\ \varrho_{3,1} & \varrho_{3,2} & 1 & \cdots & \varrho_{3,K} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \varrho_{K,1} & \varrho_{K,2} & \varrho_{K,3} & \cdots & 1 \end{bmatrix} \]

- Each entry represents the correlation coefficient between two samples of the observation set

\[ \varrho_{a,b} = \varrho_{b,a} = \frac{\text{cov}(B_a, B_b)}{\sigma_S^2} = \frac{E\left\{ \left( B_a - \mu_S \right) \left( B_b - \mu_S \right) \right\}}{E\left\{ \left( S_n - \mu_S \right)^2 \right\}} \]
Auto-Covariance Matrix of the Observation Set

\[
C_B = \mathbb{E}\left\{ \left( B_n - \mathbb{E}\{ B_n \} \right) \left( B_n - \mathbb{E}\{ B_n \} \right)^T \right\} = \sigma_S^2 \cdot \begin{bmatrix} 1 & \varrho_{1,2} & \varrho_{1,3} & \cdots & \varrho_{1,K} \\ \varrho_{2,1} & 1 & \varrho_{2,3} & \cdots & \varrho_{2,K} \\ \varrho_{3,1} & \varrho_{3,2} & 1 & \cdots & \varrho_{3,K} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \varrho_{K,1} & \varrho_{K,2} & \varrho_{K,3} & \cdots & 1 \end{bmatrix}
\]

- Each entry represents the correlation coefficient between two samples of the observation set

\[
\varrho_{a,b} = \varrho_{b,a} = \frac{\text{cov}(B_a, B_b)}{\sigma_S^2} = \frac{\mathbb{E}\left\{ \left( B_a - \mu_S \right) \left( B_b - \mu_S \right) \right\}}{\mathbb{E}\left\{ \left( S_n - \mu_S \right)^2 \right\}}
\]

⇒ Property of the source (and choice of observation set)
Auto-Covariance Matrix of the Observation Set

\[ C_B = \mathbb{E}\left\{ (B_n - \mathbb{E}\{B_n\})(B_n - \mathbb{E}\{B_n\})^T\right\} = \sigma^2_S \cdot \begin{bmatrix}
1 & \varrho_{1,2} & \varrho_{1,3} & \cdots & \varrho_{1,K} \\
\varrho_{2,1} & 1 & \varrho_{2,3} & \cdots & \varrho_{2,K} \\
\varrho_{3,1} & \varrho_{3,2} & 1 & \cdots & \varrho_{3,K} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\varrho_{K,1} & \varrho_{K,2} & \varrho_{K,3} & \cdots & 1
\end{bmatrix} \]

- Each entry represents the correlation coefficient between two samples of the observation set

\[ \varrho_{a,b} = \varrho_{b,a} = \frac{\text{cov}(B_a, B_b)}{\sigma^2_S} = \frac{\mathbb{E}\left\{ (B_a - \mu_S)(B_b - \mu_S) \right\}}{\mathbb{E}\left\{ (S_n - \mu_S)^2 \right\}} \]

- Property of the source (and choice of observation set)
- Can be measured for given signal (or set of signals)
Cross-Covariance Vector of the Observation Set and Current Sample

\[ c = \mathbb{E}\left\{ \left( S_n - \mathbb{E}\{ S_n \} \right) \left( B_n - \mathbb{E}\{ B_n \} \right) \right\} = \sigma_S^2 \cdot \begin{bmatrix} \varrho_{c,1} \\
\varrho_{c,2} \\
\varrho_{c,3} \\
\vdots \\
\varrho_{c,K} \end{bmatrix} \]

- Each entry represents the correlation coefficient between the sample to be predicted and a sample of the observation set

\[ \varrho_{c,k} = \frac{\text{cov}(S_n, B_k)}{\sigma_S^2} = \frac{\mathbb{E}\left\{ \left( S_n - \mu_S \right) \left( B_k - \mu_S \right) \right\}}{\mathbb{E}\left\{ \left( S_n - \mu_S \right)^2 \right\}} \]
Cross-Covariance Vector of the Observation Set and Current Sample

\[ \mathbf{c} = \mathbb{E}\left\{ \left( S_n - \mathbb{E}\{ S_n \} \right) \left( B_n - \mathbb{E}\{ B_n \} \right) \right\} = \sigma_S^2 \cdot \begin{bmatrix} \varrho_{c,1} \\ \varrho_{c,2} \\ \varrho_{c,3} \\ \vdots \\ \varrho_{c,K} \end{bmatrix} \]

Each entry represents the correlation coefficient between the sample to be predicted and a sample of the observation set

\[ \varrho_{c,k} = \frac{\text{cov}(S_n, B_k)}{\sigma_S^2} = \frac{\mathbb{E}\left\{ \left( S_n - \mu_S \right) \left( B_k - \mu_S \right) \right\}}{\mathbb{E}\left\{ \left( S_n - \mu_S \right)^2 \right\}} \]

- Property of the source (and choice of observation set)
Cross-Covariance Vector of the Observation Set and Current Sample

\[ \mathbf{c} = \mathbb{E}\left\{ \left( S_n - \mathbb{E}\{ S_n \} \right) \left( B_n - \mathbb{E}\{ B_n \} \right) \right\} = \sigma_S^2 \cdot \begin{bmatrix} \varrho_{c,1} \\ \varrho_{c,2} \\ \vdots \\ \varrho_{c,K} \end{bmatrix} \]

- Each entry represents the correlation coefficient between the sample to be predicted and a sample of the observation set

\[ \varrho_{c,k} = \frac{\text{cov}(S_n, B_k)}{\sigma_S^2} = \frac{\mathbb{E}\left\{ \left( S_n - \mu_S \right) \left( B_k - \mu_S \right) \right\}}{\mathbb{E}\left\{ \left( S_n - \mu_S \right)^2 \right\}} \]

- Property of the source (and choice of observation set)
- Can be measured for given signal (or set of signals)
Optimal Linear and Affine Prediction

- Goal: Minimization of prediction error variance \( \sigma^2_U \)

\[
\sigma^2_U(a) = \sigma^2_S - 2a^Tc + a^TC_Ba
\]
Optimal Linear and Affine Prediction

- **Goal:** Minimization of prediction error variance $\sigma_U^2$

  $$\sigma_U^2(a) = \sigma_S^2 - 2 a^T c + a^T C_B a$$

- **Set derivative with respect to $a$ equal to zero**

  $$\frac{\partial}{\partial a} \sigma_U^2(a) = -2 c + 2 C_B \cdot a = 0$$

Yule-Walker equations (linear equation system)

Optimal predictor $a$ solves $C_B \cdot a = c$
Optimal Linear and Affine Prediction

- **Goal:** Minimization of prediction error variance $\sigma_U^2$
  \[
  \sigma_U^2(a) = \sigma_S^2 - 2a^T c + a^T C_B a
  \]
- **Set derivative with respect to $a$ equal to zero**
  \[
  \frac{\partial}{\partial a} \sigma_U^2(a) = -2c + 2C_B \cdot a = 0
  \]

⇒ **Yule-Walker equations** (linear equation system)

optimal predictor $a$ solves $C_B \cdot a = c$
Optimal Linear and Affine Prediction

- Goal: Minimization of prediction error variance $\sigma_U^2$

$$\sigma_U^2(a) = \sigma_S^2 - 2a^Tc + a^TC_Ba$$

- Set derivative with respect to $a$ equal to zero

$$\frac{\partial}{\partial a} \sigma_U^2(a) = -2c + 2C_Ba = 0$$

→ Yule-Walker equations (linear equation system)

optimal predictor $a$ solves

$$C_B \cdot a = c$$

→ Optimal affine prediction: Additionally, set

$$a_0 = \mu_S \left(1 - \sum_{k=1}^{K} a_k\right)$$
Summary: Derivation of Optimal Affine Predictor for Given Source

1. Choose suitable observation set $\mathcal{B}_n = \{ b_1, b_2, \cdots, b_K \}$
Summary: Derivation of Optimal Affine Predictor for Given Source

1. Choose suitable observation set $\mathcal{B}_n = \{b_1, b_2, \ldots, b_K\}$
2. Determine mean $\mu_S$ of sources and required correlation coefficients $\varrho_{k,\ell}$
Summary: Derivation of Optimal Affine Predictor for Given Source

1. Choose suitable observation set \( \mathcal{B}_n = \{b_1, b_2, \cdots, b_K\} \)

2. Determine mean \( \mu_S \) of sources and required correlation coefficients \( \varrho_{k,\ell} \)

3. Solve linear equation system for determining prediction parameters \( a_1, a_2, \cdots, a_K \)

\[
\begin{bmatrix}
1 & \varrho_{1,2} & \varrho_{1,3} & \cdots & \varrho_{1,K} \\
\varrho_{2,1} & 1 & \varrho_{2,3} & \cdots & \varrho_{2,K} \\
\varrho_{3,1} & \varrho_{3,2} & 1 & \cdots & \varrho_{3,K} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\varrho_{K,1} & \varrho_{K,2} & \varrho_{K,3} & \cdots & 1
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
a_3 \\
\vdots \\
a_K
\end{bmatrix}
=
\begin{bmatrix}
\varrho_{c,1} \\
\varrho_{c,2} \\
\varrho_{c,3} \\
\vdots \\
\varrho_{c,K}
\end{bmatrix}
\]
Summary: Derivation of Optimal Affine Predictor for Given Source

1. Choose suitable observation set $\mathcal{B}_n = \{b_1, b_2, \cdots, b_K\}$

2. Determine mean $\mu_S$ of sources and required correlation coefficients $\varrho_{k,\ell}$

3. Solve linear equation system for determining prediction parameters $a_1, a_2, \cdots, a_K$

$$
\begin{pmatrix}
1 & \varrho_{1,2} & \varrho_{1,3} & \cdots & \varrho_{1,K} \\
\varrho_{2,1} & 1 & \varrho_{2,3} & \cdots & \varrho_{2,K} \\
\varrho_{3,1} & \varrho_{3,2} & 1 & \cdots & \varrho_{3,K} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\varrho_{K,1} & \varrho_{K,2} & \varrho_{K,3} & \cdots & 1
\end{pmatrix}
\begin{pmatrix}
a_1 \\
a_2 \\
a_3 \\
\vdots \\
a_K
\end{pmatrix}
= 
\begin{pmatrix}
\varrho_{c,1} \\
\varrho_{c,2} \\
\varrho_{c,3} \\
\vdots \\
\varrho_{c,K}
\end{pmatrix}
$$

4. Determine constant offset

$$a_0 = \mu_S \left(1 - \sum_{k=1}^{K} a_k\right)$$
Example: Prediction of Audio Signals using Three Preceding Samples

1. Choice of observation set for sample $s_n$: $B_n = \{s_{n-1}, s_{n-2}, s_{n-3}\}$
Example: Prediction of Audio Signals using Three Preceding Samples

1. Choice of observation set for sample $s_n$: $\mathcal{B}_n = \{s_{n-1}, s_{n-2}, s_{n-3}\}$

2. Need to determine mean $\mu_S$ and three correlation coefficients $\rho_k$ (for $k = 1, 2, 3$)

\[
\mu_S = E\{ S_n \} = \frac{1}{N} \sum_{n=1}^{N} s_n
\]

\[
\rho_k = \frac{E\{ (S_n - \mu_S)(S_{n-k} - \mu_S) \}}{E\{ (S_n - \mu_S)^2 \}}
\]

Example

\[
\mu_S = -37.6917
\]

\[
\rho_1 = 0.9581
\]

\[
\rho_2 = 0.8619
\]

\[
\rho_3 = 0.7564
\]
Example: Prediction of Audio Signals using Three Preceding Samples

1. Choice of observation set for sample $s_n$: $B_n = \{s_{n-1}, s_{n-2}, s_{n-3}\}$

2. Need to determine mean $\mu_S$ and three correlation coefficients $\rho_k$ (for $k = 1, 2, 3$)

$$\mu_S = E\{S_n\} = \frac{1}{N} \sum_{n=1}^{N} s_n$$

$$\rho_k = \frac{E\{(S_n - \mu_S)(S_{n-k} - \mu_S)\}}{E\{(S_n - \mu_S)^2\}}$$

Example $\mu_S = -37.6917$.

$\rho_1 = 0.9581$.

$\rho_2 = 0.8619$.

$\rho_3 = 0.7564$. 
Example: Prediction of Audio Signals using Three Preceding Samples

1. Choice of observation set for sample \( s_n \): \( B_n = \{ s_{n-1}, s_{n-2}, s_{n-3} \} \)

2. Need to determine mean \( \mu_S \) and three correlation coefficients \( \varrho_k \) (for \( k = 1, 2, 3 \))

\[
\mu_S = \mathbb{E}\{ S_n \} = \frac{1}{N} \sum_{n=1}^{N} s_n
\]

\[
\varrho_k = \frac{\mathbb{E}\{ (S_n - \mu_S)(s_{n-k} - \mu_S) \}}{\mathbb{E}\{ (S_n - \mu_S)^2 \}} = \frac{\sum_{n=k}^{N} (s_n - \mu_S)(s_{n-k} - \mu_S)}{\sum_{n=k}^{N} (s_n - \mu_S)^2}
\]

Example:

\( \mu_S = -37.6917 \)

\( \varrho_1 = 0.9581 \)

\( \varrho_2 = 0.8619 \)

\( \varrho_3 = 0.7564 \)
Example: Prediction of Audio Signals using Three Preceding Samples

1. Choice of observation set for sample $s_n$: $\mathcal{B}_n = \{s_{n-1}, s_{n-2}, s_{n-3}\}$

2. Need to determine mean $\mu_S$ and three correlation coefficients $\varrho_k$ (for $k = 1, 2, 3$)

\[
\mu_S = E\{S_n\} = \frac{1}{N} \sum_{n=1}^{N} s_n
\]

\[
\varrho_k = \frac{E\{(S_n - \mu_S)(S_{n-k} - \mu_S)\}}{E\{(S_n - \mu_S)^2\}} = \frac{\sum_{n=k}^{N}(s_n - \mu_S)(s_{n-k} - \mu_S)}{\sum_{n=k}^{N}(s_n - \mu_S)^2}
\]

Example

$\mu_S = -37.6917$

$\varrho_1 = 0.9581$

$\varrho_2 = 0.8619$

$\varrho_3 = 0.7564$
Example: Prediction of Audio Signals using Three Preceding Samples

3 Solve linear equation system

\[
\begin{bmatrix}
1 & \varrho_1 & \varrho_2 \\
\varrho_1 & 1 & \varrho_1 \\
\varrho_2 & \varrho_1 & 1
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
a_3
\end{bmatrix}
= 
\begin{bmatrix}
\varrho_1 \\
\varrho_2 \\
\varrho_3
\end{bmatrix}
\implies
a_1 = 1.9409 \\ a_2 = -1.4580 \\ a_3 = 0.4804
\]

Note: Predictor must be rounded to an integer (want to apply lossless coding)
Example: Prediction of Audio Signals using Three Preceding Samples

3 Solve linear equation system

\[
\begin{bmatrix}
1 & \varrho_1 & \varrho_2 \\
\varrho_1 & 1 & \varrho_1 \\
\varrho_2 & \varrho_1 & 1
\end{bmatrix}
\cdot
\begin{bmatrix}
a_1 \\
a_2 \\
a_3
\end{bmatrix}
=
\begin{bmatrix}
\varrho_1 \\
\varrho_2 \\
\varrho_3
\end{bmatrix}
\Rightarrow
\begin{align*}
a_1 &= 1.9409 \\
a_2 &= -1.4580 \\
a_3 &= 0.4804
\end{align*}
\]

4 Determine constant offset

\[
a_0 = \mu S \left( 1 - \sum_{k=1}^{K} a_k \right)
\Rightarrow
a_0 = -1.3833
\]
Example: Prediction of Audio Signals using Three Preceding Samples

3 Solve linear equation system

\[
\begin{bmatrix}
1 & \varrho_1 & \varrho_2 \\
\varrho_1 & 1 & \varrho_1 \\
\varrho_2 & \varrho_1 & 1
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
a_3
\end{bmatrix}
= 
\begin{bmatrix}
\varrho_1 \\
\varrho_2 \\
\varrho_3
\end{bmatrix}
\Rightarrow 
\begin{align*}
a_1 &= 1.9409 \\
a_2 &= -1.4580 \\
a_3 &= 0.4804
\end{align*}

4 Determine constant offset

\[
a_0 = \mu s \left( 1 - \sum_{k=1}^{K} a_k \right)
\Rightarrow 
\begin{align*}
a_0 &= -1.3833
\end{align*}

→ Predictor is given by

\[
\hat{s}_n = \text{round} \left( a_0 + a_1 \cdot s_{n-1} + a_2 \cdot s_{n-2} + a_3 \cdot s_{n-3} \right)
\]

\[
= \text{round} \left( -1.3833 + 1.9409 \cdot s_{n-1} -1.4580 \cdot s_{n-2} + 0.4804 \cdot s_{n-3} \right)
\]
Example: Prediction of Audio Signals using Three Preceding Samples

3 Solve linear equation system

\[
\begin{bmatrix}
1 & \varrho_1 & \varrho_2 \\
\varrho_1 & 1 & \varrho_1 \\
\varrho_2 & \varrho_1 & 1
\end{bmatrix}
\cdot
\begin{bmatrix}
a_1 \\
a_2 \\
a_3
\end{bmatrix}
=
\begin{bmatrix}
\varrho_1 \\
\varrho_2 \\
\varrho_3
\end{bmatrix}
\implies
\begin{align*}
a_1 &= 1.9409 \\
a_2 &= -1.4580 \\
a_3 &= 0.4804
\end{align*}
\]

4 Determine constant offset

\[a_0 = \mu_\mathcal{S} \left( 1 - \sum_{k=1}^{K} a_k \right) \implies a_0 = -1.3833\]

\(\Rightarrow\) Predictor is given by

\[
\hat{s}_n = \text{round}\left( a_0 + a_1 \cdot s_{n-1} + a_2 \cdot s_{n-2} + a_3 \cdot s_{n-3} \right)
\]

\[
= \text{round}\left( -1.3833 + 1.9409 \cdot s_{n-1} - 1.4580 \cdot s_{n-2} + 0.4804 \cdot s_{n-3} \right)
\]

Note: Predictor must be rounded to an integer (want to apply lossless coding)
Example: Prediction of Audio Signals using Three Preceding Samples

\[ \sigma = 4880.3 \]
\[ H = 14.25 \]
Example: Prediction of Audio Signals using Three Preceding Samples

- **Original Signal**: $\sigma = 4880.3$, $H = 14.25$
- **Simple Predictor** ($\hat{s}_n = s_{n-1}$): $\sigma = 1412.0$, $H = 12.39$

---

Heiko Schwarz (Freie Universität Berlin) — Data Compression: Predictive Lossless Coding
Example: Prediction of Audio Signals using Three Preceding Samples

original

\[ p_S(x) \]

\[ \sigma = 4880.3 \]
\[ H = 14.25 \]

simple predictor
\( \hat{s}_n = s_{n-1} \)

\[ p_U(x) \]

\[ \sigma = 1412.0 \]
\[ H = 12.39 \]

optimal affine predictor (3 samples)

\[ p_U(x) \]

\[ \sigma = 887.6 \]
\[ H = 11.72 \]
Potential Improvements for Audio Coding

1. Use more samples as observation set
   - May improve the affine predictor
   - Requires transmission of more prediction parameters
   - Audio compression format FLAC uses up to 32 preceding samples
### Potential Improvements for Audio Coding

1. **Use more samples as observation set**
   - May improve the affine predictor
   - Requires transmission of more prediction parameters
   - Audio compression format **FLAC** uses up to 32 preceding samples

2. **Adapt predictor during coding**
   - Audio signals have instationary properties
   - A single predictor may not be suitable for all parts of an audio stream
   - Split data into chunks and determine best predictor for each chunk
Example: Prediction of Image Signals using Three Neighboring Samples

1. Choice of prediction structure

\[ \hat{X} = a_0 + a_L \cdot L + a_A \cdot A + a_C \cdot C \]
Example: Prediction of Image Signals using Three Neighboring Samples

1. **Choice of prediction structure**

\[ \hat{X} = a_0 + a_L \cdot L + a_A \cdot A + a_C \cdot C \]

2. **Determine mean and required correlation factors**

- \[ \varrho_{\text{hor}} = \frac{\sum_{x,y}(s[x, y] - \mu_S)(s[x - 1, y] - \mu_S)}{\sum_{x,y}(s[x, y] - \mu_S)^2} \]
- \[ \varrho_{\text{ver}} = \frac{\sum_{x,y}(s[x, y] - \mu_S)(s[x, y - 1] - \mu_S)}{\sum_{x,y}(s[x, y] - \mu_S)^2} \]
- \[ \varrho_{\text{al}} = \frac{\sum_{x,y}(s[x, y] - \mu_S)(s[x - 1, y - 1] - \mu_S)}{\sum_{x,y}(s[x, y] - \mu_S)^2} \]
- \[ \varrho_{\text{ar}} = \frac{\sum_{x,y}(s[x, y] - \mu_S)(s[x + 1, y - 1] - \mu_S)}{\sum_{x,y}(s[x, y] - \mu_S)^2} \]
Example: Prediction of Image Signals using Three Neighboring Samples

1. Choice of prediction structure

\[ \hat{X} = a_0 + a_L \cdot L + a_A \cdot A + a_C \cdot C \]

2. Determine mean and required correlation factors

\[
\begin{align*}
\varrho_{\text{hor}} &= \frac{\sum_{x,y} (s[x, y] - \mu_S)(s[x - 1, y] - \mu_S)}{\sum_{x,y} (s[x, y] - \mu_S)^2} \\
\varrho_{\text{ver}} &= \frac{\sum_{x,y} (s[x, y] - \mu_S)(s[x, y - 1] - \mu_S)}{\sum_{x,y} (s[x, y] - \mu_S)^2} \\
\varrho_{\text{al}} &= \frac{\sum_{x,y} (s[x, y] - \mu_S)(s[x - 1, y - 1] - \mu_S)}{\sum_{x,y} (s[x, y] - \mu_S)^2} \\
\varrho_{\text{ar}} &= \frac{\sum_{x,y} (s[x, y] - \mu_S)(s[x + 1, y - 1] - \mu_S)}{\sum_{x,y} (s[x, y] - \mu_S)^2}
\end{align*}
\]

\[ \mu_S = 124.05 \]
\[ \varrho_{\text{hor}} = 0.9722 \]
\[ \varrho_{\text{ver}} = 0.9850 \]
\[ \varrho_{\text{al}} = 0.9598 \]
\[ \varrho_{\text{ar}} = 0.9689 \]
3 Solve linear equation system

\[
\begin{bmatrix}
1 & \varrho_{ar} & \varrho_{ver} \\
\varrho_{ar} & 1 & \varrho_{hor} \\
\varrho_{ver} & \varrho_{hor} & 1
\end{bmatrix} \cdot \begin{bmatrix}
a_L \\
a_A \\
a_C
\end{bmatrix} = \begin{bmatrix}
\varrho_{hor} \\
\varrho_{ver} \\
\varrho_{al}
\end{bmatrix} \implies
\]

Predictor is given by

\[
\hat{X} = \text{round}\left( a_0 + a_L \cdot L + a_A \cdot A + a_C \cdot C \right)
\]
Example: Prediction of Image Signals using Three Neighboring Samples

3 Solve linear equation system

\[
\begin{bmatrix}
1 & \varrho_{ar} & \varrho_{ver} \\
\varrho_{ar} & 1 & \varrho_{hor} \\
\varrho_{ver} & \varrho_{hor} & 1
\end{bmatrix}
\cdot
\begin{bmatrix}
 a_L \\
 a_A \\
 a_C
\end{bmatrix}
=\begin{bmatrix}
 \varrho_{hor} \\
 \varrho_{ver} \\
 \varrho_{al}
\end{bmatrix}
\Rightarrow
\begin{align*}
a_L &= 0.5892 \\
a_A &= 0.8255 \\
a_C &= -0.4262
\end{align*}
\]
Example: Prediction of Image Signals using Three Neighboring Samples

3 Solve linear equation system

\[
\begin{bmatrix}
1 & \varphi_{ar} & \varphi_{ver} \\
\varphi_{ar} & 1 & \varphi_{hor} \\
\varphi_{ver} & \varphi_{hor} & 1 \\
\end{bmatrix}
\begin{bmatrix}
a_L \\
a_A \\
a_C \\
\end{bmatrix}
= 
\begin{bmatrix}
\varphi_{hor} \\
\varphi_{ver} \\
\varphi_{al} \\
\end{bmatrix}
\Rightarrow
\begin{align*}
a_L &= 0.5892 \\
a_A &= 0.8255 \\
a_C &= -0.4262 \\
\end{align*}
\]

4 Determine constant offset

\[
a_0 = \mu_S (1 - a_L - a_A - a_C)
\Rightarrow
a_0 = 1.4198
\]
Example: Prediction of Image Signals using Three Neighboring Samples

3 Solve linear equation system

\[
\begin{bmatrix}
1 & \varrho_{ar} & \varrho_{ver} \\
\varrho_{ar} & 1 & \varrho_{hor} \\
\varrho_{ver} & \varrho_{hor} & 1 \\
\end{bmatrix}
\cdot
\begin{bmatrix}
a_L \\
a_A \\
a_C \\
\end{bmatrix}
=
\begin{bmatrix}
\varrho_{hor} \\
\varrho_{ver} \\
\varrho_{al} \\
\end{bmatrix}
\Rightarrow
\begin{align*}
a_L &= 0.5892 \\
a_A &= 0.8255 \\
a_C &= -0.4262
\end{align*}
\]

4 Determine constant offset

\[
a_0 = \mu_S \left(1 - a_L - a_A - a_C\right)
\Rightarrow
a_0 = 1.4198
\]

⇒ Predictor is given by

\[
\hat{X} = \text{round}\left(a_0 + a_L \cdot L + a_A \cdot A + a_C \cdot C\right)
\]

\[
= \text{round}\left(1.4198 + 0.5892 \cdot L + 0.8255 \cdot A - 0.4262 \cdot C\right)
\]
Example: Prediction of Image Signals using Three Neighboring Samples

$p_S(x)$

original

$\sigma = 47.85$

$H = 7.45$
Example: Prediction of Image Signals using Three Neighboring Samples

\[ p_S(x) \]

original

\[ \sigma = 47.85 \]
\[ H = 7.45 \]

\[ p_U(x) \]

simple predictor \((\hat{X} = L)\)

\[ \sigma = 11.34 \]
\[ H = 5.05 \]
Example: Prediction of Image Signals using Three Neighboring Samples

- Original image
  - $p_S(x)$
  - $\sigma = 47.85$
  - $H = 7.45$

- Simple predictor ($\hat{X} = L$)
  - $p_U(x)$
  - $\sigma = 11.34$
  - $H = 5.05$

- Optimal affine predictor (3 neighbours)
  - $p_U(x)$
  - $\sigma = 6.78$
  - $H = 4.52$
Potential Improvements for Image Coding

1. Use more samples as observation set
   - May improve the affine predictor
   - Requires transmission of more prediction parameters

2. Adapt predictor during coding
   - Image signals have instationary properties: Direction of edges plays an important role
   - Split image into blocks and choose predictor for each block
   - May be sufficient to choose between pre-defined predictors
   - Horizontal predictor for parts with horizontal edges
   - Vertical predictor for parts with vertical edges
   - Some further predictors

3. Non-linear Predictors
   - Non-linear predictors may be able to deal with different edge directions
Potential Improvements for Image Coding

1. **Use more samples as observation set**
   - May improve the affine predictor
   - Requires transmission of more prediction parameters

2. **Adapt predictor during coding**
   - Image signals have instationary properties: Direction of edges plays an important role
   - Split image into blocks and choose predictor for each block
   - May be sufficient to choose between pre-defined predictors
     - Horizontal predictor for parts with horizontal edges
     - Vertical predictor for parts with vertical edges
     - Some further predictors
Potential Improvements for Image Coding

1. Use more samples as observation set
   - May improve the affine predictor
   - Requires transmission of more prediction parameters

2. Adapt predictor during coding
   - Image signals have instationary properties: Direction of edges plays an important role
   - Split image into blocks and choose predictor for each block
   - May be sufficient to choose between pre-defined predictors
     - Horizontal predictor for parts with horizontal edges
     - Vertical predictor for parts with vertical edges
     - Some further predictors

3. Non-linear Predictors
   - Non-linear predictors may be able to deal with different edge directions
Examples for Non-Linear Predictors

**LOCO Predictor used in JPEG-LS**

- Each sample $X$ is predicted according to

$$
\hat{X} = \begin{cases} 
    \min(L, A) & : C \geq \max(L, A) \\
    \max(L, A) & : C \leq \min(L, A) \\
    L + A - C & : \text{otherwise}
\end{cases}
$$

![Diagram](https://via.placeholder.com/150)
Examples for Non-Linear Predictors

**LOCO Predictor used in JPEG-LS**

- Each sample $X$ is predicted according to

\[
\hat{X} = \begin{cases} 
\min(L, A) & : C \geq \max(L, A) \\
\max(L, A) & : C \leq \min(L, A) \\
L + A - C & : \text{otherwise}
\end{cases}
\]

**Motion Vector Prediction in Video Coding Standards**

- Motion vector $m$ is predicted by component-wise median

\[
\hat{m}_x(X) = \text{median}(m_x(A), m_x(B), m_x(C))
\]

\[
\hat{m}_y(X) = \text{median}(m_y(A), m_y(B), m_y(C))
\]
Summary of Lecture

Predictive Lossless Coding

- Entropy coding of prediction error signals $u_n = s_n - \hat{s}_n$
- Simple and effective way to exploit dependencies between neighbouring samples
- Complexity reduction relative to more general conditional entropy coding

Optimal Prediction

- Given by conditional mean for an observation set
- Complex due to requirement of large tables (similar to conditional entropy coding)

Affine and Linear Prediction

- Simple structure of predictor, low-complex implementations possible
- Optimal prediction parameters are given by solution of Yule-Walker equations
- For instationary sources (such as audio, image, video signals):
  - Determine predictor for smaller sets of samples (still large enough)
  - Determine optimal predictor or choose between set of predefined predictors
Exercise: Lossless Image Compression Challenge (Part I)

Implement an encoder and decoder for lossless coding of 8-bit color images:

1. We use the PPM format as raw data format:
   - The encoder should read the original images in PPM format.
   - The decoder should write the reconstructed images in PPM format.

Example images (24 PPM images of the Kodak set) are provided on the course web-site (and in the KVV).

2. Use coding techniques that you learned for efficiently compressing the 8-bit color images.
   - A combination of prediction and entropy coding of the prediction errors is suggested.
   - Start with a simple (but working) approach and try to improve your codec step by step.

structure of “ppm” files:

```
P6 // ascii (fixed)
width height // ascii
255 // ascii (max. value)
<binary data> // binary
```

binary data:

- pixels in raster-scan order (line by line)
- each pixel consists of three 8-bit values
  - R: red component (0..255)
  - G: green component (0..255)
  - B: blue component (0..255)
- the values R, G, B for a pixel follow each other (before the values for the next pixel)

suggestion:

- Store the red, green, and blue components of an image into separate arrays
- Code the color components independently