SCHOOL OF COMPUTER TECHNOLOGY

AASD 4001 Mathematical Concepts for Machine Learning

Lecture 4

Reza Moslemi, Ph.D., P.Eng.

Session 3

- Matrix Factorization
- Mathematics of Digital Signal Processing

Matrix Factorization

Factorization:

- $x^2 - y^2 = (x - y)(x + y)$
- $x^2 + y^2 + 2xy = (x + y)^2$

The same idea can be applied to matrices and is called matrix factorization!

- There are various methods for matrix factorization, each with its own properties and applications
  - LU (lower-upper) Decomposition
  - Cholesky Decomposition
  - QR Decomposition
  - Singular Value Decomposition (SVD)
  - Etc.

What is most relevant to us in this course, especially for recommender systems, is SVD.

Matrix Factorization (cont'd)

What is Singular Value Decomposition (SVD)?

the singular value decomposition of a complex m-by-n matrix M is given by:

- $\boldsymbol{M} = \boldsymbol{U\Sigma V^T}$

where m-by-m $\boldsymbol{U}$ and n-by-n $\boldsymbol{V}$ are orthogonal matrices, and $\boldsymbol{\Sigma}$ is a m-by-n rectangular diagonal matrix with non-negative real numbers on the diagonal.

orthogonal matrix:

- a square matrix whose columns and rows are orthonormal vectors
- In simple terms: $\boldsymbol{V^T V} = \boldsymbol{V V^T} = \boldsymbol{I}$ or $\boldsymbol{V^T} = \boldsymbol{V^{-1}}$

Matrix Factorization (cont'd)

An example of SVD:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & \sqrt{5} & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ -\sqrt{0.2} & 0 & 0 & 0 & -\sqrt{0.8} \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -\sqrt{0.8} & 0 & 0 & 0 & \sqrt{0.2} \end{bmatrix}$$

Obviously, manual calculation of the above SVD-related matrices is cumbersome.

- Fortunately, numpy linear algebra library has built-in SVD method and we will shortly explore an SVD example with that
- It is important to know the concept behind the libraries we are using

Matrix Factorization (cont'd)

How does SVD relate to ML and recommendation systems in particular?

Consider the following rating matrix (user-item, user-movie here) :

	Movie #1	Movie #2	Movie #3	Movie #4
User #1	5	3	-	1
User #2	4	-	-	1
User #3	1	1	-	5
User #4	1	-	-	4
User #5	-	1	5	4

Where each user has rated some movies 0-5 (the "–" in the table means that the user has not rated that movie).

- e.g.: Netflix released such a database for completion on Kaggle (appox. 500,000 user and 17,000 movies).

Matrix Factorization (cont'd)

Let's assume that there are n users, m movies, and k ($k \leq n$, to be chosen by the algo. designer) latent features, based on which the users have rated the movies.

If we form an R matrix consisting of the data shown in the movie database, we can use SVD-like decomposition to write:

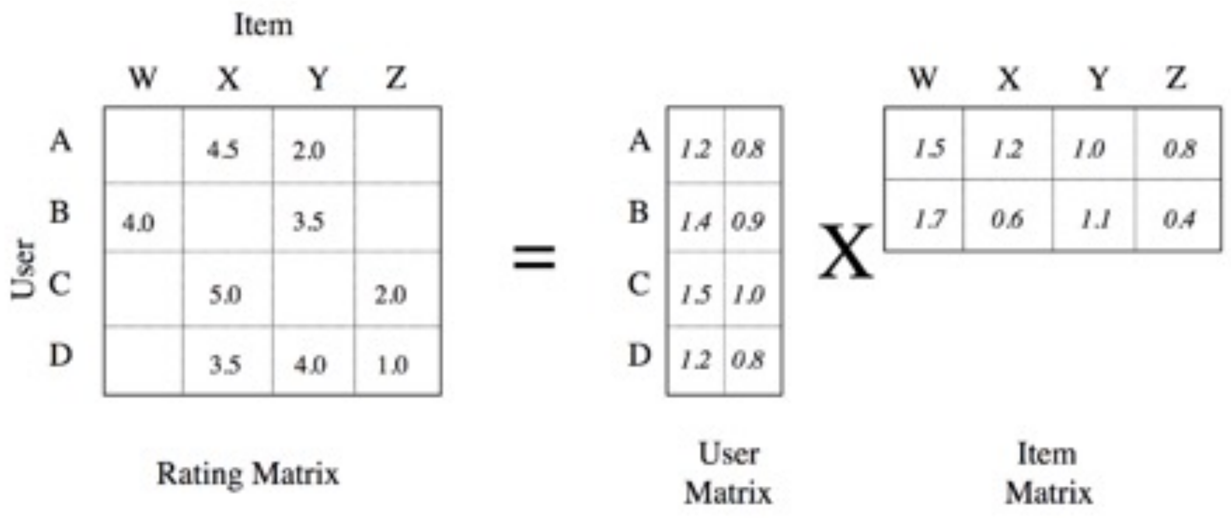
$$\boldsymbol{R} \approx \boldsymbol{P} * \boldsymbol{Q}^{\boldsymbol{T}} = \widehat{\boldsymbol{R}}$$

Each row of the n-by-k $\boldsymbol{P}$ matrix denotes the association btw a user and the features.

Each row of the m-by-k $\boldsymbol{Q}$ matrix denotes the association btw a movie and the features.

Matrix Factorization (cont'd)

As an example:



$$\underset{\text{Rating Matrix}}{\begin{array}{c|cccc} & \text{W} & \text{X} & \text{Y} & \text{Z} \\ \hline \text{A} & & 4.5 & 2.0 & \\ \text{B} & 4.0 & & 3.5 & \\ \text{C} & & 5.0 & & 2.0 \\ \text{D} & & 3.5 & 4.0 & 1.0 \end{array}} = \underset{\text{User Matrix}}{\begin{array}{c|cc} \text{A} & 1.2 & 0.8 \\ \text{B} & 1.4 & 0.9 \\ \text{C} & 1.5 & 1.0 \\ \text{D} & 1.2 & 0.8 \end{array}} \times \underset{\text{Item Matrix}}{\begin{array}{cccc} \text{W} & \text{X} & \text{Y} & \text{Z} \\ \hline 1.5 & 1.2 & 1.0 & 0.8 \\ 1.7 & 0.6 & 1.1 & 0.4 \end{array}}$$

If we could find P and Q, each prediction for any user rating any movie would be:

- $\hat{r}_{ij} = p_i q_i^T = \sum_{a=1}^{k} p_{ia} q_{aj}$

Matrix Factorization (cont'd)

But how do we find P and Q matrices?

- SVD only works for matrices without missing values, but the rating matrix has a lot of missing values
- We can form an optimization problem to solve that issue

One approach is to initialize the P and Q matrices with some random values, calculate $\hat{\boldsymbol{R}}$ and calculate how different it is from actual $\boldsymbol{R}$ (the error).

In simple terms, we need to minimize the following loss function containing squared errors:

$$min_{\boldsymbol{P},\boldsymbol{Q}} \sum_{i,x} \left(r_{xi} - \boldsymbol{p}_{\boldsymbol{i}} . \boldsymbol{q}_{\boldsymbol{x}}^{\boldsymbol{T}}\right)^2$$

Matrix Factorization (cont'd)

How to minimize the given cost function?

- In session 4, we learnt about the gradient descent algorithm for loss function minimization.

For the loss function $\sum_{i,x}\left(r_{xi} - \boldsymbol{p_i}.\boldsymbol{q_x^T}\right)^2$, the gradients are given by:

- $\nabla p_{x,k} = \sum_{i,x} -2\left(r_{xi} - \boldsymbol{p_i}.\boldsymbol{q_x^T}\right) q_{ik}$
- $\nabla q_{i,k} = \sum_{i,x} -2\left(r_{xi} - \boldsymbol{p_i}.\boldsymbol{q_x^T}\right) p_{xk}$

and the iterative update formula for P and Q matrices is given by:

- $\boldsymbol{P} \leftarrow \boldsymbol{P} - \eta.\nabla\boldsymbol{P}$
- $\boldsymbol{Q} \leftarrow \boldsymbol{Q} - \eta.\nabla\boldsymbol{Q}$

Matrix Factorization (cont'd)

In order to avoid overfitting, it is common to augment the previous loss function with an extra term of the following form (aka regularization):

$$min_{\boldsymbol{P},\boldsymbol{Q}} \sum_{i,x} \left(r_{xi} - \boldsymbol{p_i}.\boldsymbol{q_x^T}\right)^2 + \lambda \left[\sum_x \|p_x\|^2 + \sum_i \|q_i\|^2\right]$$

Which results in the following expressions for the gradients:

- $\nabla p_{x,k} = \sum_{i,x} -2\left(r_{xi} - \boldsymbol{p_i}.\boldsymbol{q_x^T}\right) q_{ik} + 2\lambda p_{i,k}$
- $\nabla q_{i,k} = \sum_{i,x} -2\left(r_{xi} - \boldsymbol{p_i}.\boldsymbol{q_x^T}\right) p_{xk} + 2\lambda q_{i,k}$

which will be used in the update formula.

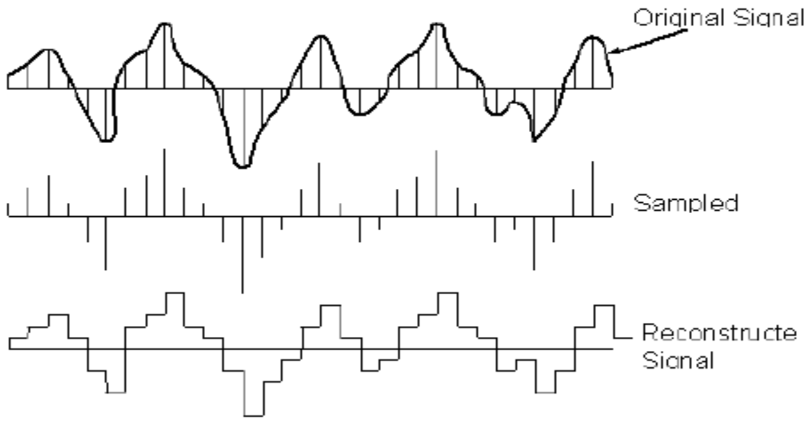
Let's move to jupyter notebook and open "Matrix Factorization.ipynb" and practice.

Mathematics of Digital Signal Processing

Digital signal processing (DSP):

- The use of digital processing units (e.g. a computer) to perform a wide variety of signal processing operations.
- The digital signal is a sequence of numbers that represent samples of a continuous variable in a domain such as time, space, or frequency.

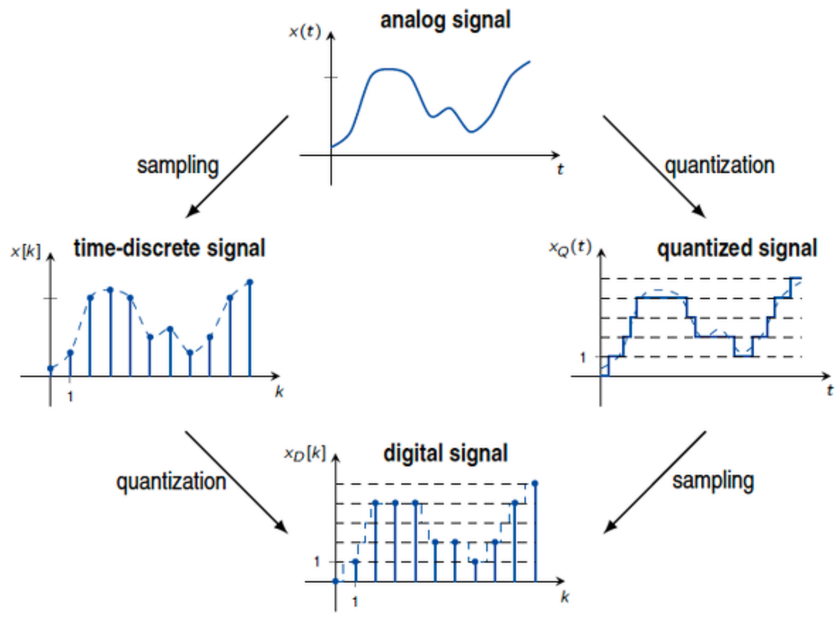
Applications:



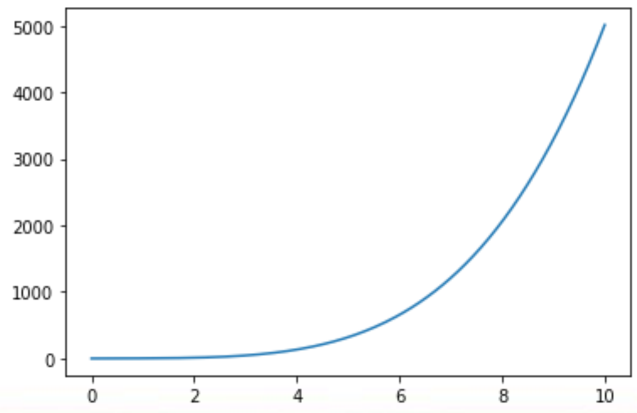
Mathematics of Digital Signal Processing (cont'd)

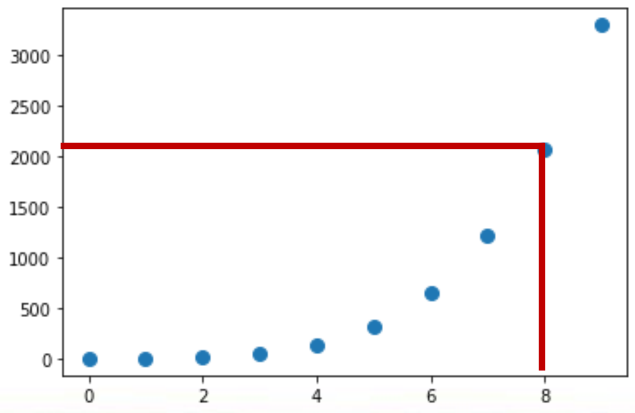
Obtaining the digital signal for original analog signal

- Sampling: converting a continuous time-varying signal into a discrete-time signal, a sequence of real numbers
- Quantization replaces each real number with an approximation from a finite set of discrete values.



$$x(t) = t + t^2 + 1/2t^4 \longrightarrow x[n]$$





Mathematics of Digital Signal Processing (cont'd)

Continuing with the previous example:

$x(t) = t + t^2 + 1/2t^4$

$$\Big\downarrow$$

$x[n]_{n=0,1,2,3,4,5} = \{0.00, 2.50, \ldots\}$

Exercise:

Consider a continuous signal $x(t) = 0.5t^2 + \sin(t)$. Calculate $f[n]$ for n = 0-5 up to 2 decimal points.

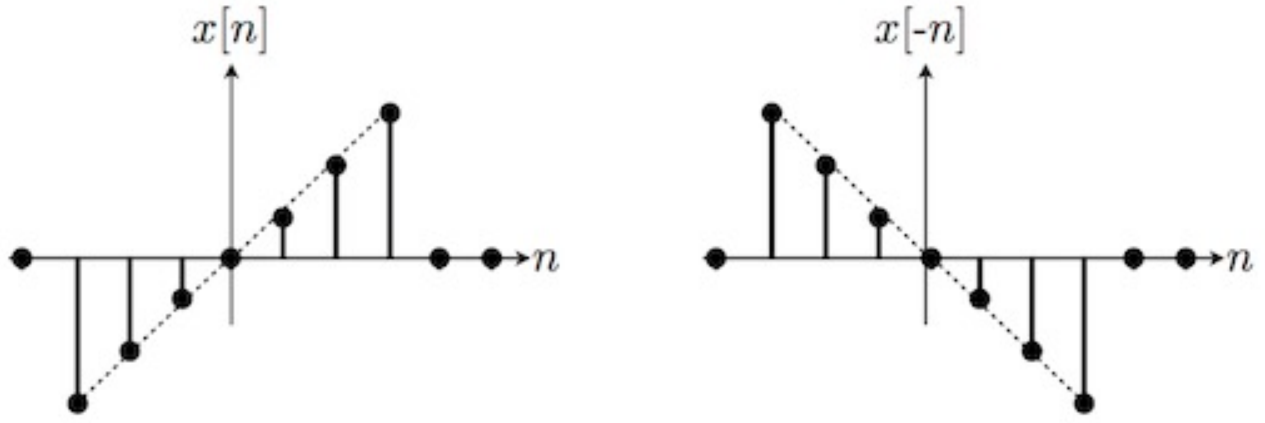
$x[n]_{n=0,1,2,3,4,5} = \{0.00, 1.34, \ldots\}$

Mathematics of Digital Signal Processing (cont'd)

Basic operations (processing) on digital signals:

Flipping (time reversal)

- It flips the signal over the y (vertical) axis.
- A technique to focus wave energy to a selected point in space and time, localize and characterize a source of wave propagation, and/or communicate information between two points.

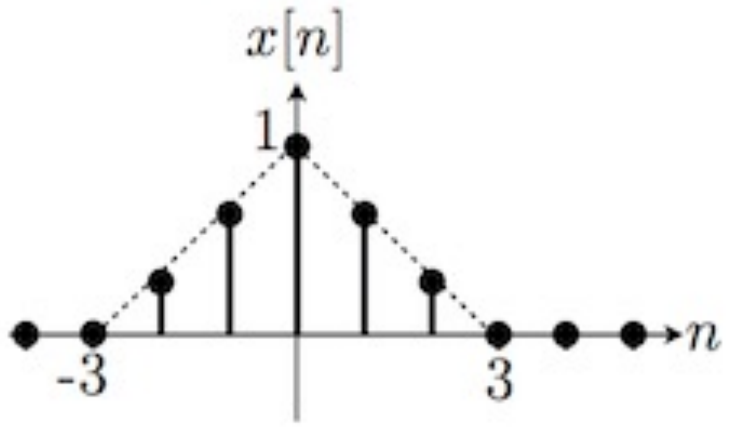


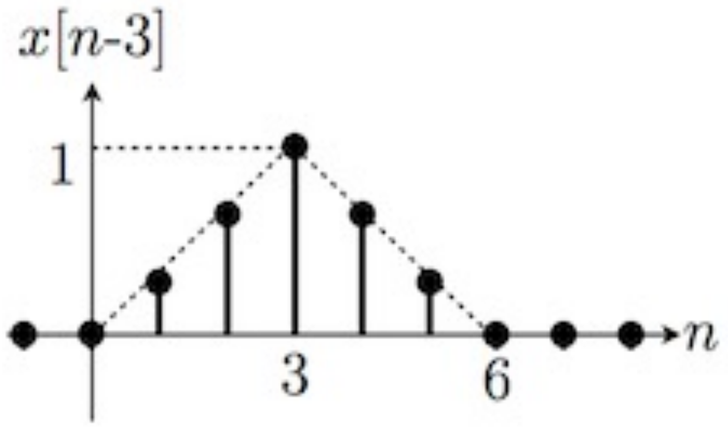
Mathematics of Digital Signal Processing (cont’d)

Basic operations (processing) on digital signals:

Time Shifting

- Time shifting is, as the name suggests, the shifting of a signal in time. This is done by adding or subtracting an integer quantity of the shift to the time variable in the function. Subtracting a fixed positive quantity from the time variable will shift the signal to the right (delay) by the subtracted quantity, while adding a fixed positive amount to the time variable will shift the signal to the left (advance) by the added quantity.





Mathematics of Digital Signal Processing (cont'd)

Basic operations (processing) on digital signals:

Time Scaling

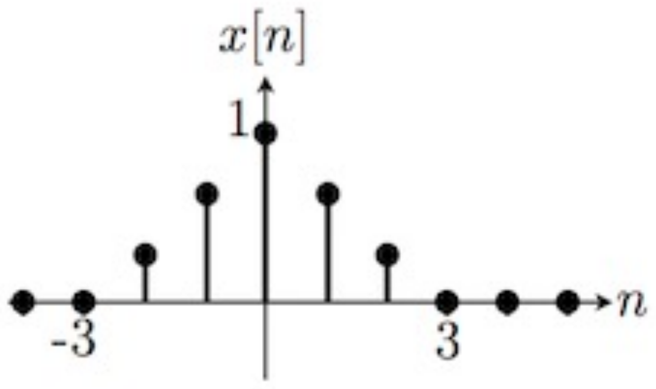
- Time scaling compresses or dilates a signal by multiplying the time variable by some quantity. If that quantity is greater than one, the signal becomes narrower and the operation is called decimation. In contrast, if the quantity is less than one, the signal becomes wider and the operation is called expansion or interpolation, depending on how the gaps between values are filled.

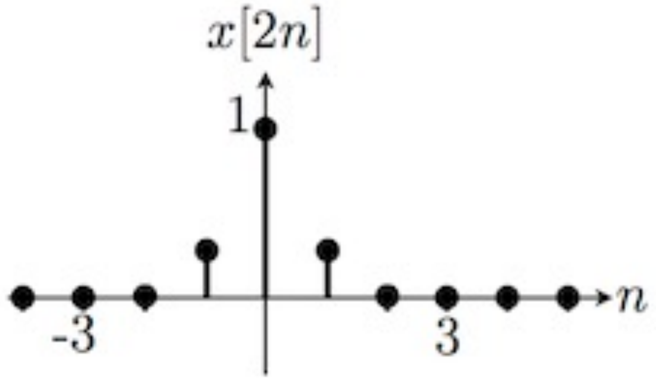
Mathematics of Digital Signal Processing (cont'd)

Basic operations (processing) on digital signals:

Decimation

- In decimation, the input of the signal is changed to be $f(cn)$. The quantity used for decimation c must be an integer so that the input takes values for which a discrete function is properly defined. The decimated signal $f(cn)$ corresponds to the original signal $f(n)$ where only each n sample is preserved (including $f(0)$), and so we are throwing away samples of the signal.
- It is the process of reducing the sampling rate. In practice, this usually implies lowpass-filtering a signal, then throwing away some of its samples. (Also downsampling or compaction)



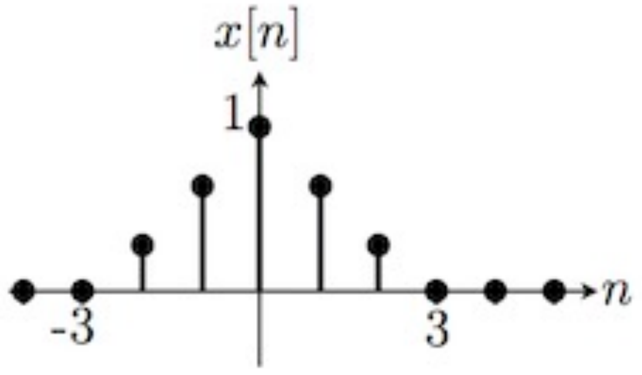


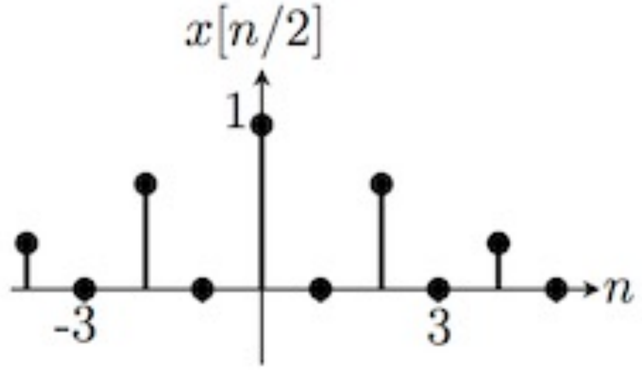
Mathematics of Digital Signal Processing (cont'd)

Basic operations (processing) on digital signals:

Expansion

- In expansion, the input of the signal is changed to be $f\left(\frac{n}{c}\right)$. We know that the signal $f(n)$ is defined only for integer values of the input n. Thus, in the expanded signal we can only place the entries of the original signal f at values of n that are multiples of c. In other words, we are spacing the values of the discrete-time signal $c-1$ entries away from each other. Since the signal is undefined elsewhere, the standard convention in expansion is to fill in the undetermined values with zeros.
- It produces an approximation of the sequence that would have been obtained by sampling the signal at a higher rate (upsampling).



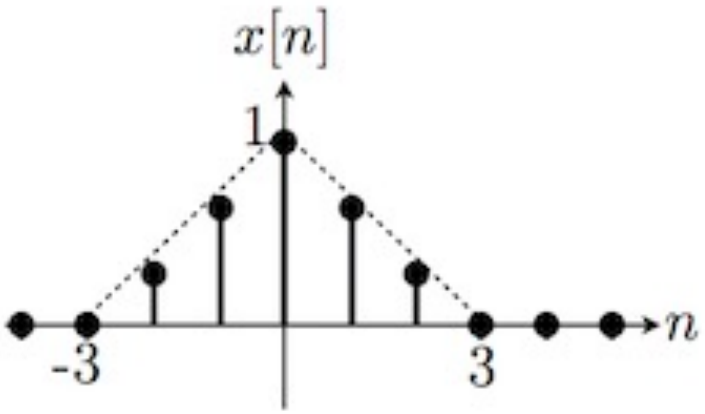


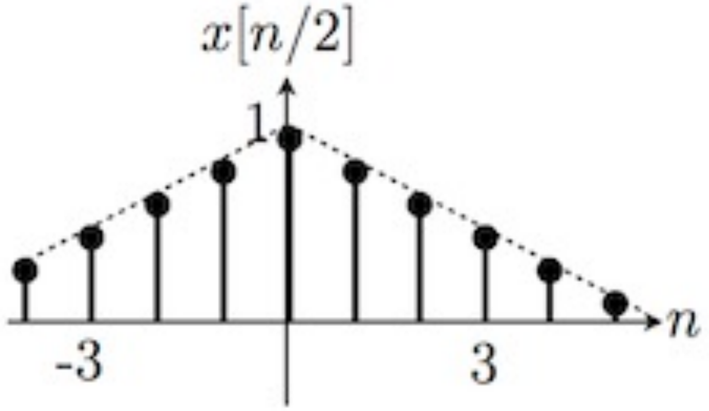
Mathematics of Digital Signal Processing (cont'd)

Basic operations (processing) on digital signals:

Interpolation

- In practice, we may know specific information that allows us to provide good estimates of the entries of $f\left(\frac{n}{c}\right)$ that are missing after expansion. This process of inferring the undefined values is known as interpolation.



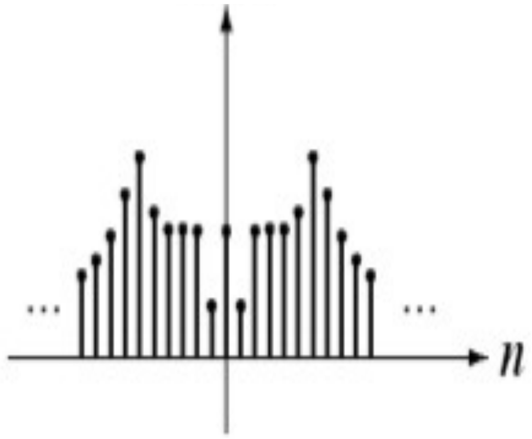


Mathematics of Digital Signal Processing (cont'd)

Even and Odd signals

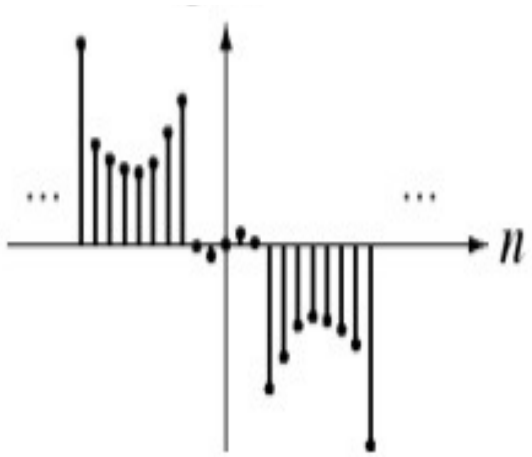
Even signal:

- Signal is flipped about the y-axis
- $x[n] = x[-n]$



Odd signal:

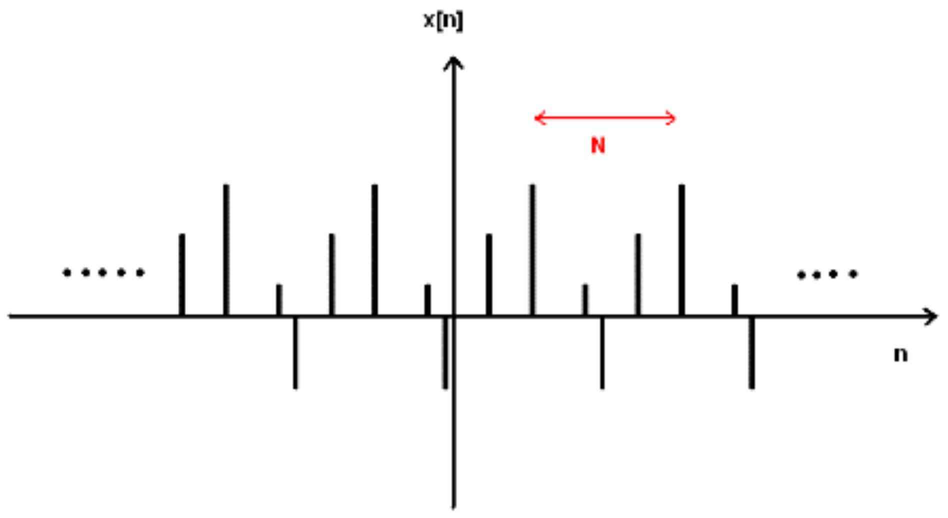
- Signal is flipped around the origin
- $x[n] = -x[-n]$
- At n=0, $x[0] = -x[0] = 0.$



Mathematics of Digital Signal Processing (cont'd)

Periodic signals:

- $x[n] = x[n+N]$



Mathematics of Digital Signal Processing (cont'd)

Complex numbers

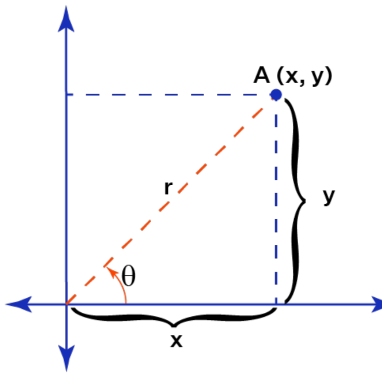
- Complex numbers shorten the equations used in DSP, and enable techniques that are difficult or impossible with real numbers alone. For instance, the Fast Fourier Transform is based on complex numbers.

Cartesian form of a complex number: $z = x + jy$

- x is the real part, Re(z)
- y is the imaginary part, Im(z)
- Imaginary unit: $j = \sqrt{-1}$

Polar form of a complex number: $z = r(\cos\theta + j\sin\theta)$

- r is the magnitude of z, $|z|$
- θ is the angle (phase) of z
- $r = \sqrt{x^2 + y^2}$ $\qquad \theta = \tan^{-1}\left(\frac{y}{x}\right)$
- $x = r\cos\theta$ $\qquad y = r\sin\theta$



Mathematics of Digital Signal Processing (cont'd)

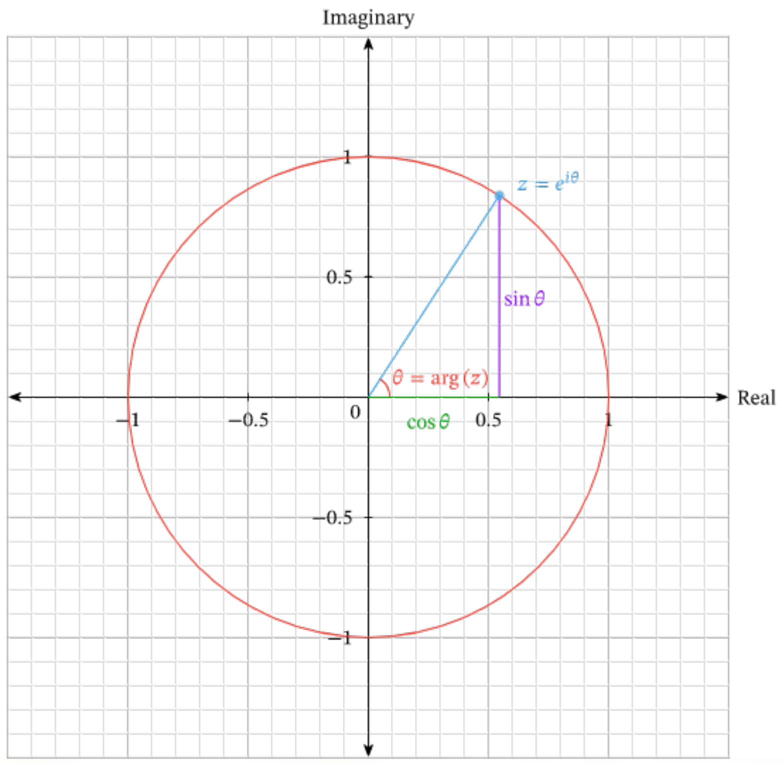
Complex numbers

Exponential form of a complex number: $z = re^{j\theta}$

- θ must be strictly in radians (NOT degrees)

Euler's formula:

- $e^{j\theta} = \cos\theta + j\sin\theta$
- $z = r(\cos\theta + j\sin\theta) = re^{j\theta}$


Imaginary
Real
$z = e^{i\theta}$
$\sin\theta$
$\theta = \arg(z)$
$\cos\theta$
1
0.5
0
−0.5
−1
1
0.5
−0.5
−1