باسمه تعالی

دانشگاه صنعتی شریف

پروژه درس سیگنال ها و سیستم ها

دانشکده مهندسی برق

نیم سال دوم ۱۴۰۱-۱۴۰۲

امیررضا ولائی ۴۰۰۱۰۲۲۲۲

احسان مریخی ۴۰۰۱۰۱۹۶۷

۱ سیگنال های تک تن و سیگنال های ویژه ها

برای سیگنال های تک تن پیوسته داریم:

$$x(t) = Ae^{j(\omega t+\phi)}$$

اگر فرض کنیم پاسخ ضربه سیستم $h(t)$ باشد، داریم:

$$y(t) = x(t) * h(t) \rightarrow y(t) = \sum_{\theta=-\infty}^{\infty} Ae^{j(\omega(t-\theta)+\phi)}h(\theta) = Ae^{j(\omega t+\phi)} \sum_{\theta=-\infty}^{\infty} Ae^{-j\omega\theta}h(\theta)$$
$$y(t) = \lambda x(t)$$

برای سیگنال های تک تن گسسته داریم:

$$x[n] = Ae^{j(\omega n+\phi)}$$

اگر فرض کنیم پاسخ ضربه سیستم $h[n]$ باشد و $x[n]$ متناوب باشد، آنگاه

$$y[n] = x[n] * h[n] \rightarrow y[n] = \sum_{\theta=-\infty}^{\infty} Ae^{j(\omega(n-\theta)+\phi)}h[\theta] = Ae^{j(\omega n+\phi)} \sum_{\theta=-\infty}^{\infty} Ae^{-j\omega\theta}h[\theta]$$
$$y[n] = \lambda x[n]$$

بنابراین نتیجه میگیریم به صورت کلی سیگنال های تک تن سیگنال ویژه های سیستم هستند؛ به صورتی که اگر وارد سیستم شوند، خروجی ضریبی از ورودی خواهد بود.

۲ ویژگی سیگنال های ویژه

می دانیم که $e^{j\omega}$ ها سیگنال ویژه های یک سیستم LTI هستند. می توانیم $\cos(2\pi 50n)$ و $\sin(2\pi 50n)$ را به صورت زیر بنویسیم:

$$\cos(2\pi 50n) = \frac{e^{j2\pi 50n} + e^{-j2\pi 50n}}{2}$$
$$\sin(2\pi 50n) = \frac{e^{j2\pi 50n} - e^{-j2\pi 50n}}{2j}$$

در نتیجه می توانیم بگوییم که $\cos(2\pi 50n)$ و $\sin(2\pi 50n)$ نیز سیگنال های ویژه هستند و اگر ورودی ترکیب خطی از آن ها باشد، بنا به اصل *Super position* خروجی نیز ترکیب خطی از خروجی های آن ها خواهد بود و تنها با دو ضریب می‌توانیم خروجی را با استفاده از پایه های $\sin(2\pi 50n)$ و $\cos(2\pi 50n)$ بنویسیم.

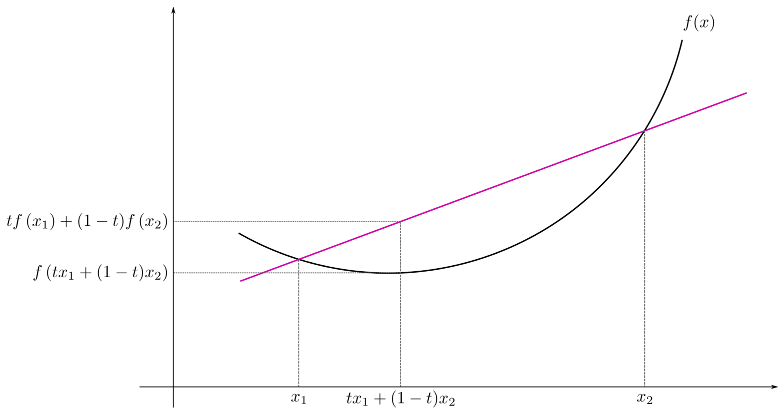
۳ Gradient Descent

الگوریتم (GD) یا Gradient descent یکی از الگوریتم های مهم در مسئله یافتن پاسخ بهینه سازی برای توابع Con-vex هست. هرچند تعمیم هایی از این الگوریتم برای توابع غیر محدب نیز استفاده می‌شود که در آن مسائل بجای یک پاسخ

بهینه برای هر نقطه شروع، ممکن است به نقاط بهینه مختلفی دست پیدا کنیم.
یک تابع در صورتی محدب یا Convex می‌باشد که رابطه ریاضی زیر برقرار باشد:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) \quad \forall x, y \in \mathbb{R}^n, \alpha \in [0,1]$$

یا به زبان ساده، اگر برای هر دو نقطه x, y در تابع f، نقطه میانی z وجود داشته باشد که مقدار تابع در آن نقطه کمتر از میانگین مقدار تابع در x, y باشد، آنگاه تابع f محدب است.



شکل ۱: تابع محدب

می‌دانیم که گرادیان یک تابع در هر نقطه، جهت بیشینه را نشان می‌دهد. بنابراین اگر در هر نقطه، به جای حرکت در جهت گرادیان، در جهت عکس گرادیان حرکت کنیم، مقدار تابع کمتر می‌شود. این روش به روش کاهش گرادیان یا Gradient Descent معروف است.

۱.۳ اثبات افزایش مقدار تابع در جهت گرادیان

اگر تابع چند متغیره $f(x_1, x_2, ..., x_n)$ را در نظر بگیریم، گرادیان آن به صورت زیر تعریف می‌شود:

$$\nabla f = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n}\right]^T$$

که در آن:

$$\frac{\partial f}{\partial x_i} = \lim_{h \to 0} \frac{f(x_1, x_2, ..., x_i + h, ..., x_n) - f(x_1, x_2, ..., x_i, ..., x_n)}{h}$$

آنگاه برای تغییرات کوچک در x، تغییرات متناظر در f به صورت زیر است:

$$f(x + \delta x) \approx f(x) + \nabla f^T \delta x = f(x) + \langle \nabla f, \delta x \rangle = f(x) + \|\nabla f\| \, \|\delta x\| \cos\theta$$

حال اگر $f(x)$ را به سمت چپ معادله ببریم، داریم:

$$f(x+\delta x) - f(x) \approx \|\nabla f\| \, \|\delta x\| \cos\theta$$

میدانیم نرم ۱ یا اندازه هر بردار مقداری ثابت و مثبت است. پس برای مثبت بودن مقدار تغییرات، باید در جهتی حرکت کنیم که $\cos\theta$ مثبت باشد، یا به عبارتی $\pi > \theta$ باشد. برای منفی بودن مقدار تغییرات نیز باید در جهتی حرکت کنیم که θ یا همان زاویه بین $\|\delta x\|$ و $\|\nabla f(x)\|$ بزرگ تر از π باشد.
بیشینه تغییرات منفی زمانی رخ می‌دهد که $\cos\theta$ کمینه باشد یا به عبارتی $\theta = \pi$ باشد یا جهت تغییرات x در خلاف جهت $\|\nabla f(x)\|$ باشد.

۲.۳ الگوریتم Gradient Descent

این الگوریتم برای پیدا کردن نقطه مینیمم توابع محدب (یا نقطه ماکسیسمم توابع مقعر با یکسری تغییر) استفاده میشود. صورت کلی الگوریتم به این شکل است که در هر Itratation، در سمت عکس گرادیان حرکت میکنیم. از قسمت قبل می‌دانیم که اگر در جهت عکس گرادیان تابع محدب حرکت کنیم، مقدار خروجی تابع کمتر می‌شود؛ لذا اگر الگوریتم را به تعداد کافی iteration با ضریب مناسب پشت گرادایان انجام دهیم؛ قطعا (مجدداً تاکید میشود که تابع باید Convex باشد یا به عبارتی ماتریس هسین تابع PD باشد) به نقطه مینیمم تابع خواهیم رسید.
همگرایی در این الگوریتم یا با تعداد iteration و یا با مقدار تفاوت تابع در دو ایتریشن آخر و مقدار ϵ که در اول اجرای الگوریتم تعیین می‌شود کنترل می‌شود و در صورت ارضای هر کدام از شرط ها اجرای آن به اتمام می‌رسد.
برای ضریب پشت گرادیان از تعاریف مختلفی استفاده می‌شود. ساده ترین تعریف استفاده از ضریب ثابت η پشت گرادیان است. یکی از تعاریف دیگر، استفاده از مومنتوم است:

$$m_t = \beta m_{t-1} + g_{t-1}$$
$$\theta_t = \theta_{t-1} + \eta_t g_t$$

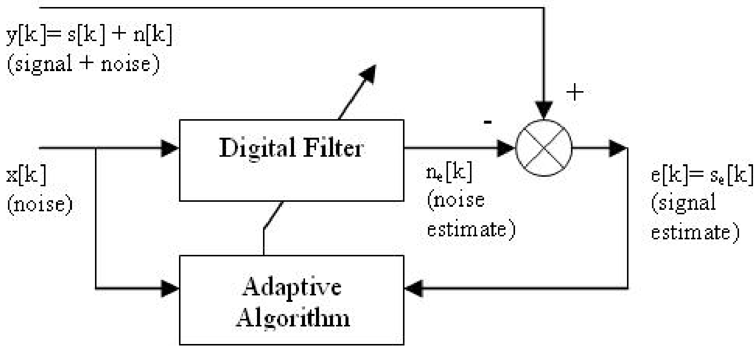
یکی از نکات مهم درمورد η این است که اگر η بزرگی را برای شروع انتخاب کنیم، ممکن است الگوریتم در مرز نزدیک نقطه مینیمم هیچوقت به نقطه مینیمم نزدیک نشود و تغییرات تابع در آن نقطه بسیار زیاد باشد. بالعکس، اگر مقدار η کوچک انتخاب شود، ممکن است لازم باشد iteration های زیادی انجام شود تا الگوریتم به نقطه مینیمم نزدیک شود.

۳.۳ الگوریتم stochastic gradient descent

در این الگوریتم، در هر iteration، یک نمونه از داده ها را انتخاب می‌کنیم و برای آن نمونه، گرادیان را محاسبه می‌کنیم و در جهت عکس آن حرکت می‌کنیم. این الگوریتم برای مسائلی که داده ها بسیار زیاد هستند و یا محاسبه گرادیان برای تمام داده ها بسیار زمان بر است، استفاده می‌شود.
همچنین می‌توانیم به جای انتخاب تمام data set، بخش کوچکی از آن را انتخاب کنیم که به این الگوریتم نیز

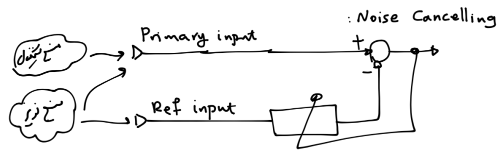
۴ بلوک دیاگرام فیلتر وفقی برای فیلتر نویز برق شهری

بلوک دیاگرام فیلترهای وفقی با کاربرد حذف نویز به صورت کلی به شکل زیر است:



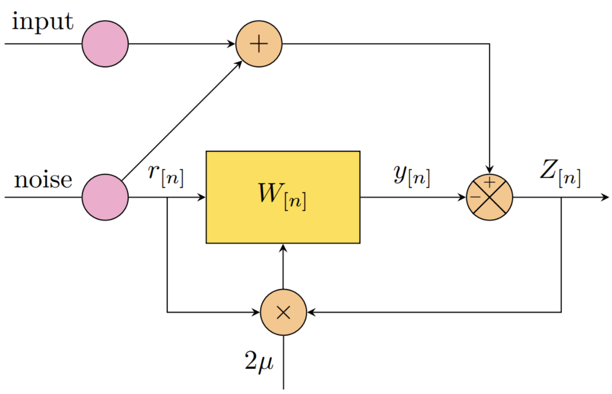
شکل ۲: بلوک دیاگرام فیلتر وفقی برای حذف نویز

که مشابه همان سیستمی است که در جزوه درس استاد بابائی آورده شده است:



شکل ۳: فیلتر وفقی با هدف حذف نویز

اگر بخواهیم با مشخص کردن بلوک Adaptive Algorithm به صورت واضح تر بلوک دیاگرام فیلتر وفقی حذف نویز را نشان دهیم، داریم:



شکل ۴: فیلتر وفقی با هدف حذف نویز

۵ مقدار $\mathbb{E}(e(n))$

مقدار $\mathbb{E}(e(n))$ با توجه به بلوک دیاگرام موجود برابر سیگنال ورودی بدون نویز است. یعنی:

$$\mathbb{E}(e(n)) = \mathbb{E}(d(n) - y(n)) = \mathbb{E}(d(n)) - \mathbb{E}(y(n))$$

به عبارتی، انتظار داریم در صورت همگرایی الگوریتم GD، سیگنال $\mathbb{E}(e(n))$ به سیگنال ورودی بدون محتوی نویز هم‌گرا شود.
به عبارتی، تنها در صورتی که سیگنال اولیه صرفا شامل نویز باشد انتظار داریم که مقدار $\mathbb{E}(e(n))$ صفر باشد (که تا حدی غیرمنطقی نیز هست).

۶ حذف نویز سینوسی 50 هرتز از وویس ریکورد شده با روش LMS

توضیح کلی الگوریتم پیاده سازی شده : در اینجا به جای استفاده از اختلاف سیگنال بدون نویز و سیگنال تخمین به عنوان تابع خطا از خود سیگنال تخمین به عنوان خطا استفاده میکنیم. هدف مینیمم کردن این تابع خطا می‌باشد که از الگوریتم LMS کمک میگیریم تا به این هدف برسیم. در این الگوریتم یه تابع وزن داریم که در هر مرحله نویز رفرنس را آپدیت میکند تا شبیه به نویز روی سیگنال اصلی شود. در هر مرحله تابع وزن خودش را با کمک تابع خطا آپدیت میکند تا به حالت اپتیمال برسد.
تاثیر مقدار μ روی همگرایی و دقت : با انتخاب بزرگتر مقدار μ سرعت همگرایی بیشتر میشود اما دقت آن کمتر میشود. دلیل این اتفاق این است که تابع وزن سریعتر میتواند به مقدار اپتیمال نزدیک شود اما وقتی به آن نزدیک میشود به خاطر بزرگ بودن μ نمیتواند خود را به اپتیمال ترین حالت ممکن برساند. همچنین اگر μ بسیار کوچک انتخاب شود دقت همگرایی زیاد میشود اما سرعت همگرایی کم میشود.

در این بخش از کد با ران گرفتن می‌توانیم ۵ ثانیه وویس گرفته و آن را با نام audio.wav سیو کنیم.

```
%% Question 1
clear all;
% recording voice
% -----------------------------------------------

% setting properties of voice
duration = 5;
Fs = 10^4;
nBits = 24;

% creating record object
s = audiorecorder(Fs, nBits, 1);
recordblocking(s, duration);
% -----------------------------------------------

audiowrite("audio.wav",s.getaudiodata, Fs);
```

شکل ۵: ایجاد و ذخیره فایل صدا

با ران کردن این بخش ابتدا فایل با نام audio.wav خوانده میشود و سپس سیگنال سینوسی ۵۰ هرتز با فاز اولیه رندوم و مقدار 0.5 به آن اضافه میشود.

```
%% adding 50Hz noise
clc;
clear all;

[sig,Fs] = audioread("audio.wav");
% -----------------------------------------------
strPhs = rand * 2*pi;
amp = 0.5;
freq = 50;

t = 0 : 1/Fs : 5-1/Fs;
noise = amp * sin(t * 2*pi * freq + strPhs);

sigNoise = sig + noise';
% -----------------------------------------------

% voice = audioplayer(sigNoise, Fs);
% play(voice);
```

شکل ۶: اضافه کردن نویز به فایل صدا

در این بخش ابتدا نویز با جنس یکسان اولیه اما با فاز متفاوت از آن ایجاد میکنیم. سپس از آن به عنوان نویز رفرنس در الگوریتم LMS استفاده میکنیم.

```
%% removing signal using LMS algorithm
clc;
clear("noise");

freq = 50;


t = 0 : 1/Fs : 5-1/Fs;
refNoise = sin(2*pi*freq * t) + cos(2*pi*freq * t);

step = 0.0005;
weiLen = 50;

% initiating weight function
WInit = rand(1,weiLen);

% applying alorithm
for k = 1:10
    for i = weiLen:length(t)

        noiseEst(i) = WInit * refNoise(i:-1:i-50+1)';
        sigEst(i) = sigNoise(i) - noiseEst(i);
        WInit = WInit + step * sigEst(i) * refNoise(i:-1:i-50+1);
    end
end

subplot(4,1,1);
plot(t,sig);
title('Original Signal','Interpreter','latex');
ylim([-1 1]);
```

شکل ۷: حذف کردن نویز با الگوریتم LMS

۷ حذف نویز سینوسی 50 هرتز از وویس ریکورد شده با روش LMS

ابتدا فایل دیتای داده شده رو لود کرده و فرکانس سمپلینگ و محتوی آن را جدا میکنیم.

```
clear all;
clc;

Data = importdata("400101967.mat");
Fs = Data.fs;
sigNoise = Data.corrupted_signal;
```

شکل ۸: خواندن دیتا

سپس کاملا مشابه با سوال قبل همان الگوریتم را پیاده سازی کرده و با استفاده از نویز رفرنس نویز موجود در دیتا را حذف میکنیم.

```
%% removing signal using LMS algorithm
clear sound;
clc;

% define error noise estimation and filter coeffs

freq = 50;

t = 0 : 1/Fs : (length(sigNoise)-1)/Fs;
refNoise = sin(2*pi*freq * t) + cos(2*pi*freq * t);

step = 0.001;
weiLen = 50;

WInit = rand(1,weiLen);

for k = 1:10
    for i = weiLen:length(t)
        noiseEst(i) = WInit * refNoise(i:-1:i-50+1)';
        sigEst(i) = sigNoise(i) - noiseEst(i);
        WInit = WInit + step * sigEst(i) * refNoise(i:-1:i-50+1);
    end
end
```

شکل ۹: حذف نویز از دیتا