Applying AI to Wearable Data

Introduction to Wearable Data

Why Wearables

Wearables have been around for a long time as fitness devices. Fitbit's first wrist wearable came out in 2009. But before that, there were chest straps for measuring your heart rate and hip devices for counting steps.

More recently, wearables are starting to be used in making medical decisions and for clinical research. For example, in this **ABC news article**, a young woman discovered she had kidney failure by following clues from her Apple Watch.

This story highlights all the advantages of wearable devices. They can monitor people continuously and unobtrusively and discover intermittent abnormalities that point to serious medical conditions before they progress. And in this case, they alerted this woman of her condition before she developed symptoms that might have put her in the hospital.

Wearable Devices

Today the landscape of wearable devices is vast and only increasing. In part because of the miniaturization of electronics and sensors, they are much more capable than before. For example, the Apple Watch came out in 2015 and tried to track activity, heart rate, exercise, sleep. The newer model even has an ECG sensor on it for monitoring your heart rhythm. There are shoe sensors, “underwearables” - things like compression shorts, and even wearables for your baby. Pampers has a smart diaper.

But these sensors don’t produce meaningful metrics automatically. For example, the sensors on a FitBit don’t produce a pulse rate or a step count. They sense some underlying signal, for step count, it’d be motion. Then, complex algorithms will process the raw motion data into various activity classifications or step counts.

What You Will Learn

This course will teach you how to build signal processing and machine learning algorithms to process raw wearable data and produce clinically meaningful metrics. We’ll talk about the healthcare aims that can be achieved with wearables. The skills you'll develop and the algorithms you’ll learn in this class will be transferable to a broad spectrum of devices and sensors.

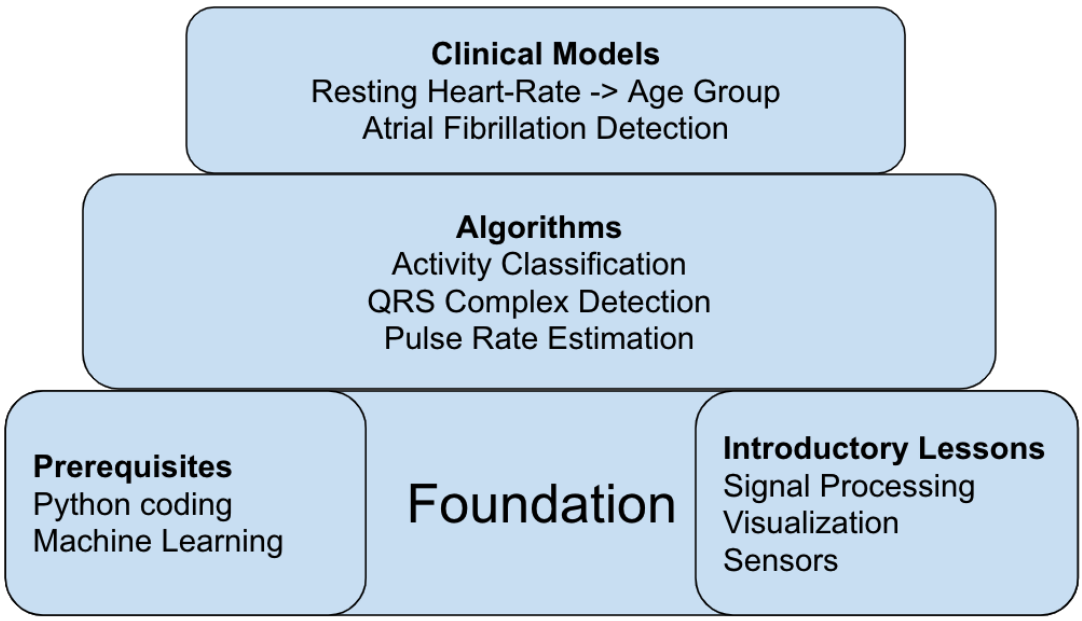
The course focuses on three main sensors for a wrist-wearable:

- the IMU sensor measures motion.
- the PPG sensor measures blood flow at the wrist.
- the ECG sensor measures the electrical activity of the heart.

We’ll begin with an introduction to how these sensors work. Then we will deep-dive into the IMU sensor and learn how to use it to do activity classification. This can help build useful context around how the wearer spends their time, how much exercise they get, how mobile they are, and so on. Next, we’ll look at the ECG sensor and learn how we can use it to detect some cardiovascular conditions. Finally, in the final project, you will use the PPG sensor to derive the pulse rate. And learn techniques to produce good results even while the wearer is exercising, which as you’ll soon learn causes all sorts of problems. You will be working intimately with time-series signals in this course, so before we jump in, we’ll do a quick review of basic time-series signal processing and learn some plotting tools to help us explore the data.

Key Topics

Key Topics in this Course



Course Outline

Lesson 1 - Introduction to Wearable Data

- Welcome / Introduction
- Wearables in a Medical Context

Lesson 2 - Introduction to Signal Processing

- Sampling
- Python plotting in time-domain
- Resampling / Interpolation
- Fourier Transform
- Python plotting in the frequency domain
- Harmonics

Lesson 3 - Introduction to Wearable Sensors

- Wearable Demo
- IMU Sensor
- Accelerometer deep-dive
- PPG Sensor
- ECG Sensor

Lesson 4 - Activity Classifier

- Introduction to Activity Classifiers
- Feature Extraction
- Model Building
- Hyperparameter Tuning

Lesson 5 - ECG Processing

- About the ECG signal
- Pan-Tompkins algorithm
- Atrial Fibrillation detection algorithm

Final Project

- Motion Compensated Pulse Rate
- Resting Heart Rate vs. Age and Sex in a Disease Population

Apple Heart Study Overview

If wearables are ever to be used in a medical context, research and clinical trials need to be done with wearable devices. We're going to look into one such study, the Apple Heart Study (AHS). Apple, with Stanford University, recruited 419,000 individuals over 8 months to participate over the internet. Participants just had to download an app over the App Store, provided they already owned an Apple Watch and an iPhone. The goal of the study was to try to detect an abnormal heart rhythm called Atrial Fibrillation (AF). AF is one of the most common arrhythmias affecting up to 6 million people in the US and it's significant because it's associated with a 4 - 5x increase in the risk of stroke.

Study Procedure

The way the AHS worked was the watch would sometimes notify participants of an irregular pulse rhythm. Once notified, the researchers would send these participants a wearable ECG chest patch for 7 days that would confirm whether they were experiencing AF. 34% of the notified participants in this study were confirmed to have AF from the chest patch. AF can be intermittent, so even if someone has it, there's no guarantee it would show up during the chest patch period.

Apple Heart Study Interpretation

Summary

I think this is a landmark study in the use of wearables for clinical research, but I don't think we should overvalue that 34% of participants were confirmed to have AF or how the Apple Watch's use as a screening or diagnostic tool. The reason for this is because the study deviated from typical clinical trials that try to prove the effectiveness of devices or drugs in a few significant ways:

- Not everyone received an ECG chest patch, so we don't know the false-negative rate.
- The inclusion criteria (Apple Watch and iPhone users) created a biased study population.

If wearable research is biased towards a more affluent population, the interventions or discoveries made may be more specifically tailored for this population and may be less effective on a less affluent population. This is similar to why researchers think long and hard about how to enroll minority populations in their clinical trials.

Despite the caveats above, this study broke new ground in significant ways:

- Recruited almost half a million people for the study in 8 months.
- These types of studies lose a lot of participants to follow-up (e.g., over 2000 participants were notified of having irregular pulse rates but only collected data from 450 ECG chest patches.)
- How participants would react to a smartwatch notifying them of an abnormal heart rhythm while it was happening.

I think the study is super exciting in that it's a first attempt to use the capability of wearables to do long term monitoring in a medical context.

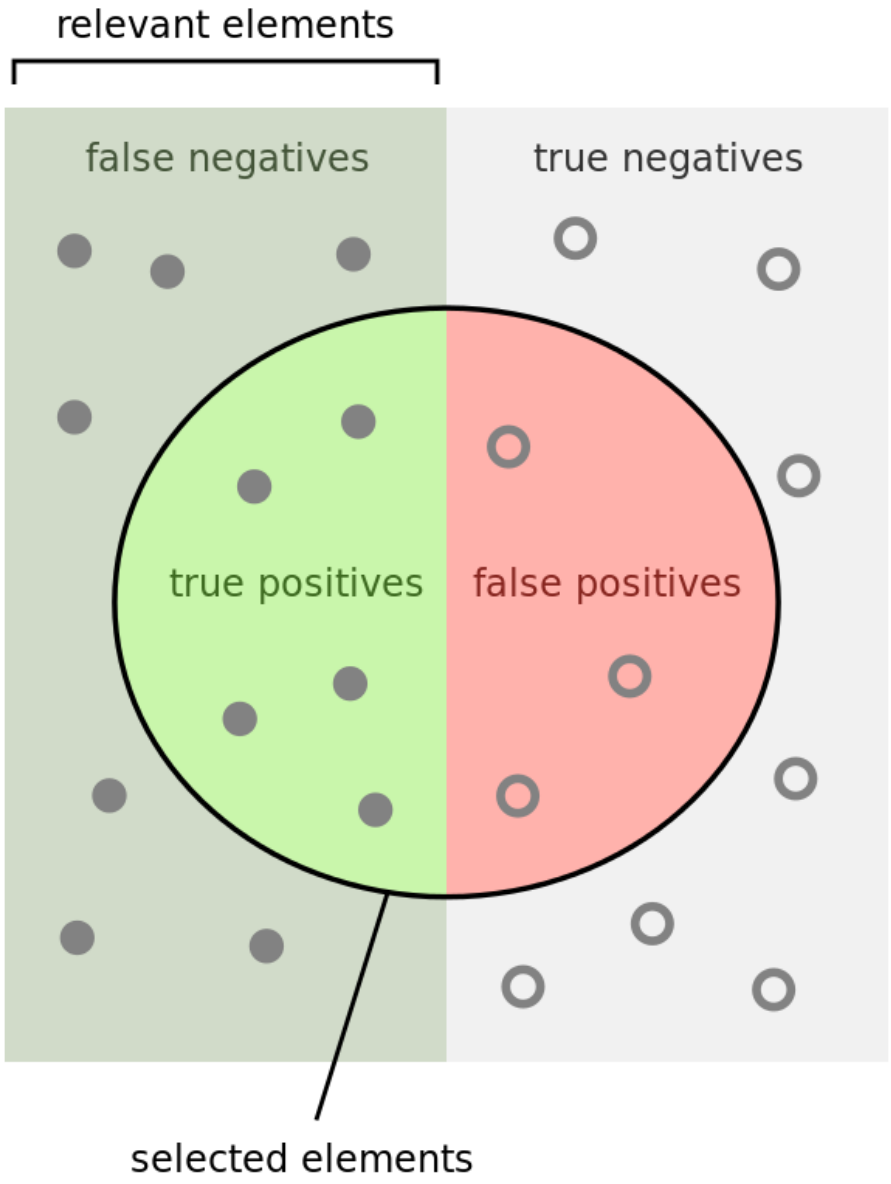
There's obviously a lot of hype around studies like this and I want us to be excited about the right things.

Classification Accuracy

From activity classification to AF detection, many tasks in wearable healthcare involve classification. In the case of the AHS, participants were classified by the "irregular pulse" notification into AF and non-AF groups. One way of discussing **classification accuracy** in binary classification is by looking at precision and recall.

- **precision**: the proportion of all true positive cases that are detected positive by the classifier
- **recall**: the proportion of all positives detected by the classifier that are true positives

This is demonstrated graphically below:



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{selected elements}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$

Precision and RecallSource: Walber. Precision and recall. Nov 2014. CC-BY-SA-4.0 Link

Further Research

Wearables are part of Digital Health, learn more about it **here**.

The wikipedia page for **precision and recall**

The Framingham Study is a landmark study in cardiovascular health. Many interesting papers came out of it, and I highly recommend exploring them. Start **here with the Wikipedia page**

If you would like to read up on the AHS on your own, you can find the clinical trial study record detail **here**.

Relevant Papers

- **Apple Heart Study** - Perez MV, Mahaffey KW, Hedlin H, et al. "Large-scale assessment of a smartwatch to identify atrial fibrillation." *N Engl J Med* 2019;381:1909-1917. **Link**.
- **Apple Heart Study Response** - Campion Edward W., Jarcho John A.. (2019) "Watched by Apple." *N Engl J Med* 381:20, 1964-1965. **Link**.
- **Framingham Study** - Wolf PA, Abbott RD, Kannel WB. "Atrial fibrillation as an independent risk factor for stroke: the Framingham Study." *Stroke* 1991;22:983-988. **Link**.
- **Wearable Health** - Montgomery, K., Chester, J., & Kopp, K. (2018). "Health Wearables: Ensuring Fairness, Preventing Discrimination, and Promoting Equity in an Emerging Internet-of-Things Environment." *Journal of Information Policy*, 8, 34-77. doi:10.5325/jinfopoli.8.2018.0034 **Link**

New Vocabulary

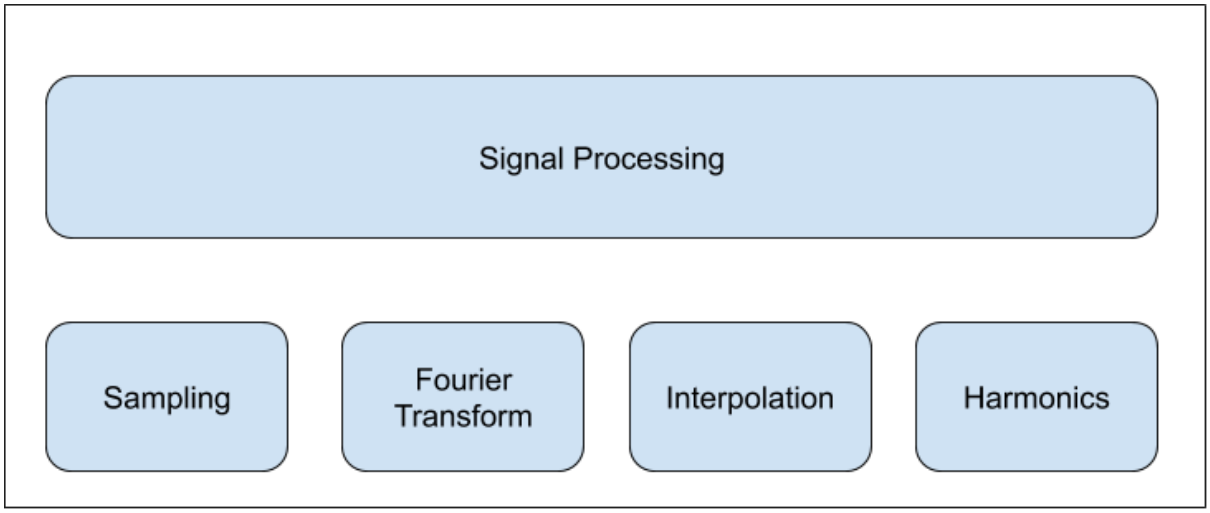
- **Inclusion Criteria**: Characteristics that potential study subjects must have for them to be included in the study.
- **Exclusion Criteria**: Characteristics that disqualify potential study subjects from participating in a clinical study. e.g., Some common ones are being under 18 or pregnant.
- **Classification Accuracy**: A metric for evaluating the performance of a classifier -- the fraction of classifications that are correct. For rare events

(like atrial fibrillation), this metric is unsuitable. For example, a classifier that classifies every data point as healthy would have a classification accuracy of 99%, as around 1 percent of the population has atrial fibrillation, but would be relatively useless.

- **Precision**: The fraction of positive classifications that are correct.
- **Recall**: The fraction of positive elements that are classified correctly as positive.

Intro to Digital Sampling & Signal Processing

Introduction



We will be covering four major areas of signal processing theory in this lesson

This lesson will be a whirlwind tour through sampling theory, signal processing, the Fourier transform, and related topics that you will need to know to complete this course. Along the way, we will learn how to plot and visualize our signals using basic python plotting functions. We will be using the following packages:

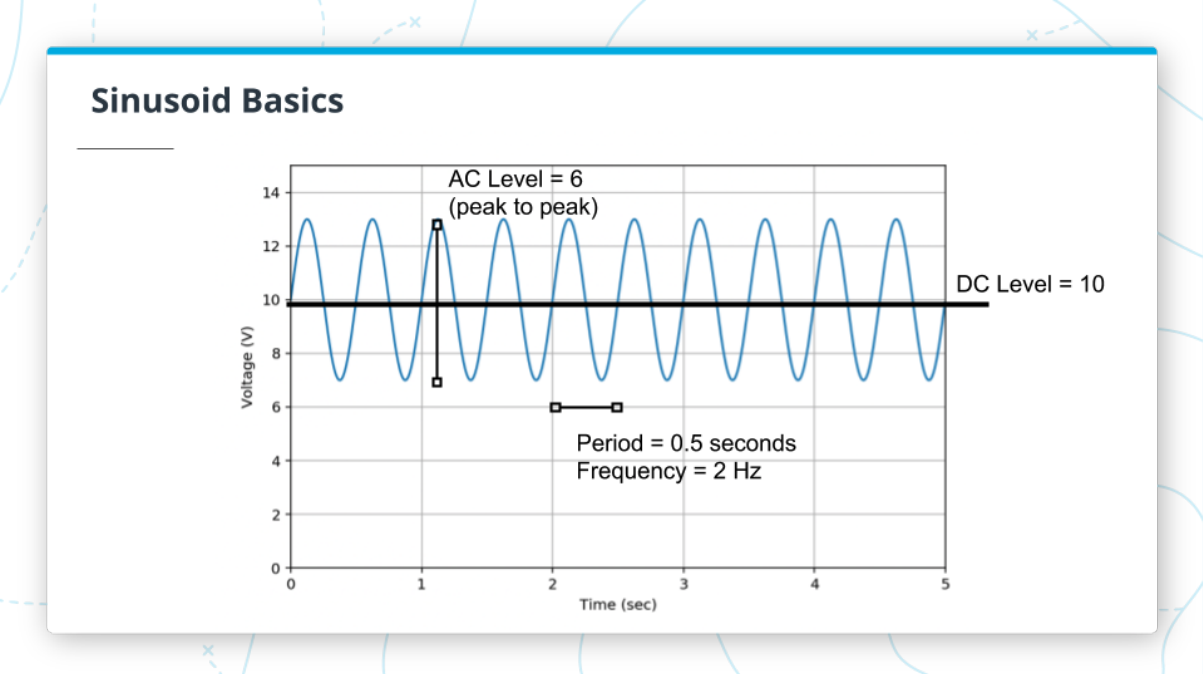
- **numpy**
- **scipy**
- **matplotlib**

- **scikit-learn**

If you are familiar with these libraries and concepts already, then this will be mostly a review. If this is all brand new to you, don't worry, this lesson assumes zero prior knowledge and doesn't overwhelm you with the theory. In fact, you won't see a single equation in this lesson. You will, however, see a lot of code.

Refresher on Signals

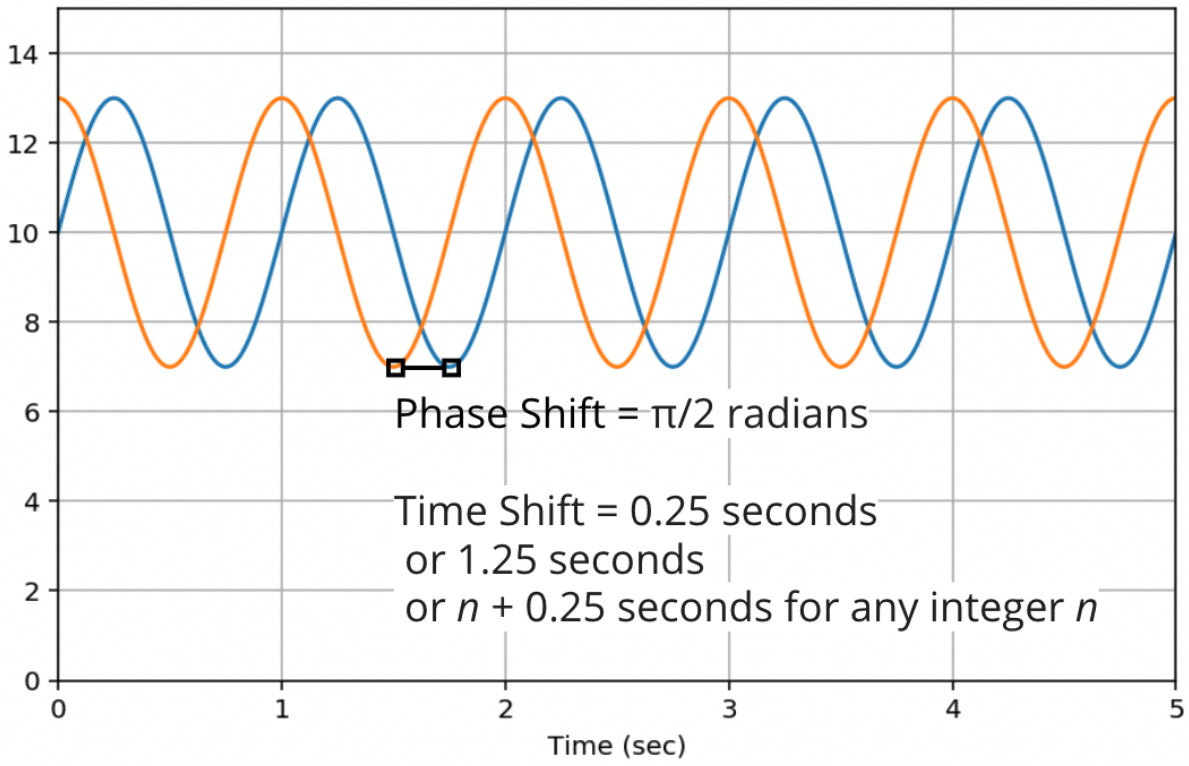
A signal is simply a series of numbers (e.g. [3, 4, 6, 2, 4] is a signal!). Typically these numbers represent a voltage that changes with respect to some physical phenomenon. When the voltage signal changes in time, it's called a time series signal. Signal processing helps us analyze this sequence of numbers so we can learn more about the physical phenomenon it represents.



Sinusoid Basics

We can use the figure above to explore some properties of signals. The time-invariant part of the signal is the DC level or mean value (black line), here it's at 10 Volts. The AC component (blue line) is the part of the signal that varies with time. There are many ways to measure the AC component -- e.g., standard deviation, variance, or interquartile range. In this case, we measured the peak-to-peak amplitude, which is 6 volts.

This signal is also periodic, meaning that it repeats itself over and over again. The **period** is the amount of time it takes to make one repetition, which in this case is half a second. The **frequency** is the inverse of the period, or the number of repetitions per second, which is 2 **hertz (Hz)**.



Phase Change

The last property of a periodic signal is the **phase shift**, which is similar to a time shift. If two signals are time shifted by one full period, there is no difference between them. For this reason, we express this shift by the fraction of the period that they are shifted. If a signal is shifted by a quarter of a period, we can say the phase shift is 90 degrees (360 being a full period). You can also measure phase shift in radians and there are 2 pi radians in a full period. So this phase shift would be half pi radians.

Making a sinusoid

$$y(t) = A\sin(2\pi f t + \varphi) + C$$

A - amplitude (AC component)

f - frequency (Hz)

φ - phase (radians)

C - DC shift

```
import numpy as np

ts = np.arange(0, 5, 1/100)
sinusoid = 3 * np.sin(2 * np.pi * 1 * ts + np.pi) + 10
```

New Vocabulary

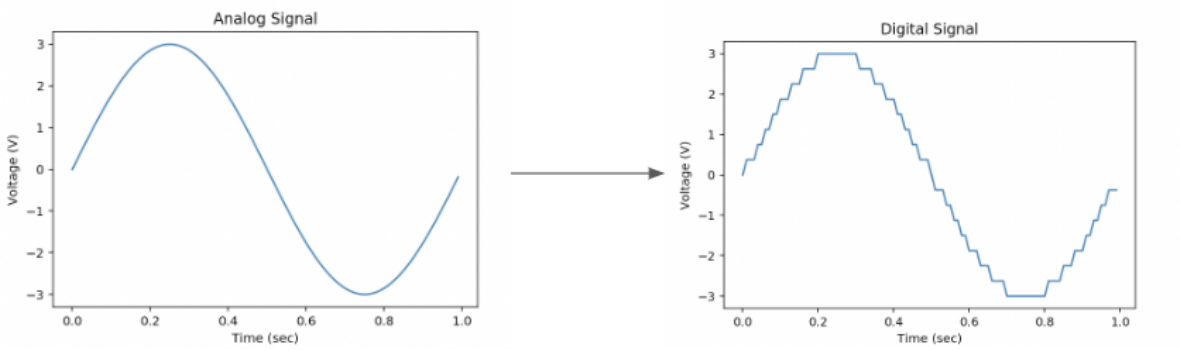
- **Period**: The amount of time it takes to make one repetition.
- **Frequency**: The amount of repetitions in a given time period, usually 1 second is the time period.
- **Hertz (Hz)**: The units of the sampling rate. 1Hz means 1 sample per second.
- **Phase Shift**: The shift between two similar periodic signals expressed in fractions of a period (multiplied by 2π radians or 360°).

Digital Sampling

The goal of digital sampling is to take an analog continuous-time signal, typically a voltage, and to quantize it and discretize it in time so that we can store it in a finite amount of memory and use the magic of computers to process it. The component that does this is called an **analog-to-digital converter** or an ADC, this is an example of a **transducer, you will learn more about transducers in a future lesson. It is important to learn about the few fundamental ways the ADC changes the analog signal. In our head, it's easy sometimes to pretend that we're dealing with an ideal analog signal, but this can get us into trouble, and it's important to know more detail about how signals are sampled to avoid pitfalls later on.

An ADC encodes a range of physical values to a set of discrete numbers. In this example, the analog signal varies over time between -3V and +3V and we are using a 4-bit ADC, which means that the ADC has 4 bits to encode the range from -3 to +3 (ie. the **bit-depth** of our sensor is 4). The 4 bits indicate that there are 16 discrete values and we can see the effect of this quantization in the digitized signal.

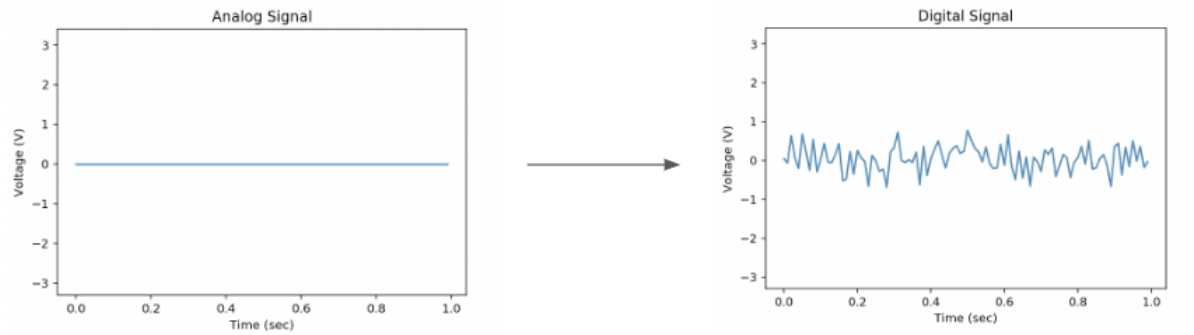
Quantization Noise



But typically, ADCs have many more bits and you won't see quantization noise because it will be overpowered by other noise sources.

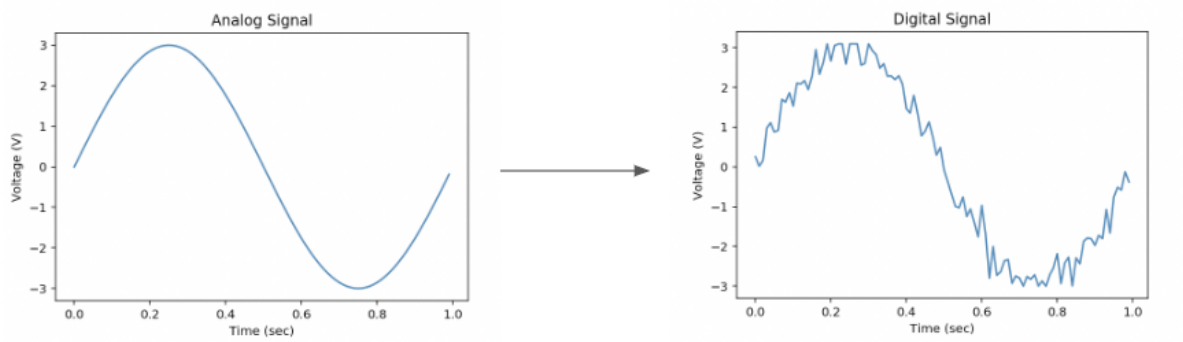
- Latent thermal energy in the system.
- Electronic noise from within the sensor.
- Electronic noise from the surroundings and the building itself. All these types of noise contribute to what we call the **noise floor**. Even when the incoming signal is perfectly flat, you will see some noise in the output. If you ever see a flat line at 0 in the output, it's because your sensor is broken.

Noise Floor



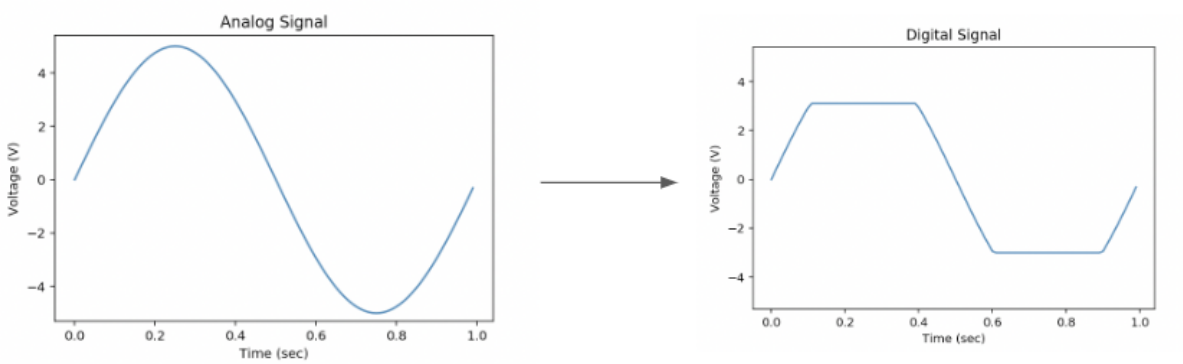
This noise is additive, so you'll see it on top of whatever incoming signal you have.

Additive Noise

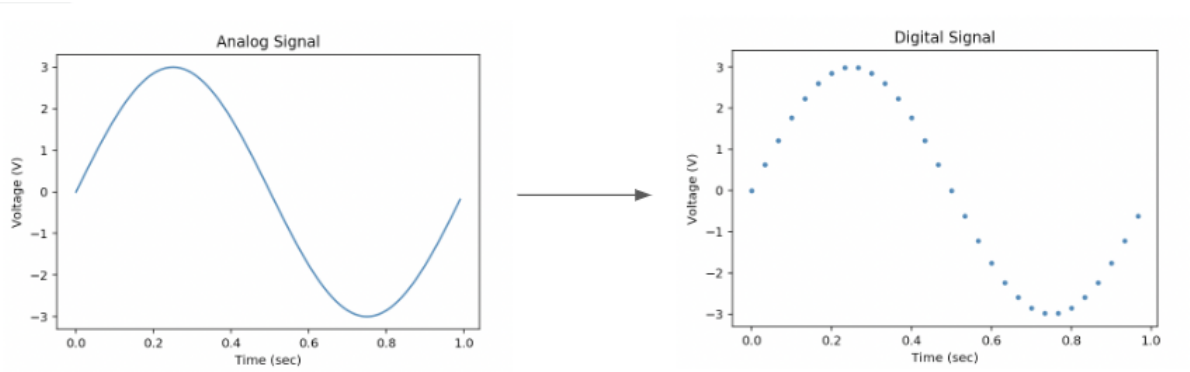


ADCs have a fixed range on the input. So for this example, our ADC was limited to -3V and +3V. This is known as the **dynamic range** of our sensor. When the input signal exceeds the dynamic range of the sensor, in this case from -4 to +4, everything greater than 3 will be clipped and set to 3 and everything smaller than -3 will be clipped to -3. We call this effect clipping, oversaturation, or undersaturation.

Signal Clipping



And finally, it's important to remember that digital signals are sampled periodically in time. We often plot them as these continuous signals by connecting the dots in between, but they are better represented as a sequence of individual points. In this example, there are 30 samples in this second, so we would say the **sampling rate** is 30 samples per second or 30 Hz.



New Vocabulary

- **Transducer**: Part of a sensor that converts a physical phenomenon into an electrical one (e.g., voltage)
- **Analog-to-Digital Convert (ADC)**: A device (usually embedded in the sensor) that converts an analog voltage into an array of bits.
- **Bit depth**: The number of bits an ADC uses to create a sample. A 16-bit ADC produces a 16-bit number for each sample.
- **Noise floor**: The total amount of noise in the sensor, including electrical interference from the environment and other parts of the device, thermal noise, and quantization noise.
- **Dynamic range**: The physical range of the sensor. Values outside of this range will show up as clipping in the digital signal.
- **Sampling rate**: The frequency at which a sensor measures a signal.

Further Research

- Sampling Signal Processing Wikipedia

Interpolation

We just learned about **interpolation** which is a technique that allows us to work with multiple signals that are sampled differently in time. We saw 2 signals that are both 1 Hz sine waves, but the one that is sampled at 60 Hz has many more data points than the one sampled at 25 Hz. After plotting and verifying the lengths of the signals, it might appear that `s2_interp` and `s1` are the same, but it is most certainly not! By plotting the original and the interpolated signal together we can see that linear interpolation estimates

points in between existing points by using a weighted average of the original points.

Previously we had only discussed uniformly sampled signals where the signal is sampled at fixed intervals in time, but sometimes we may encounter signals that are sampled haphazardly in time. This is troubling because a lot of signal processing techniques that we are about to learn require that the signal is sampled uniformly. We can fix this again with linear interpolation. When we compare the 2 signals, one uniformly and one not uniformly sampled, we can see that they follow the same continuous signal but the location of those samples are at different times. Using the np.interp function you can recover the signal in which the now non-uniformly sampled signal will have a signal point like the uniform signal. But you may notice artifacts at the edge of the resampled signal, and there is more error when the gap between existing samples is larger.

Interpolation in Review

Interpolation is a technique to estimate a signal at points in time between existing samples. We can use this technique to normalize a signal that was sampled non-uniformly. We can also use it to **resample** a signal when comparing signals that are sampled at different sampling rates. Resampling is the process of changing the sampling rate of a discrete signal.

New Vocabulary

- **Interpolation**: A method for estimating new data points within a range of discrete known data points.
- **Resampling**: The process of changing the sampling rate of a discrete signal to obtain a new discrete representation of the underlying continuous signal.

Fourier Transform

We just learned a bit about fourier transform. The theory of fourier transform is that any signal can be represented as a sum of sinusoids. Then we saw how this theory can be put into action by recreating a real accelerometer signal using only the addition of sinusoids. The frequency of the specific sinusoids

that make up a signal can tell us important information that we can use to build algorithms to process that signal.

Summary

We demonstrate the addition of sinusoids and demonstrate how the Fourier transform allows us to describe any signal as a summation of sinusoids. The frequencies of the sinusoids that comprise a signal represent the signal's **frequency components**. The range of frequency components for a signal is called its **bandwidth**.

We then discuss the **Nyquist frequency** and the limits this imposes on the sampling rate and the **bandwidth** of the signals that we sample.

If we try to sample a signal that has higher frequency components than the Nyquist frequency, we will see **aliasing**, which means those high-frequency components will show up at mirrored lower frequencies.

New Vocabulary

- **Frequency component**: The Fourier transform explains a signal as a sum of sinusoids. Each of these sinusoids is a frequency component of the signal.
- **Nyquist frequency**: Half of the sampling frequency. Signal components above this frequency will get aliased in the sampled signal.
- **Bandwidth**: A range of frequencies within a band.
- **Aliasing**: The effect that causes frequency components greater than the Nyquist frequency to become indistinguishable from frequencies below the Nyquist frequency.

Fourier Transform In Practice

We learned how to apply the theory and understanding of what a fourier transform is to actually solve some problems.

Let's make a signal that is composed of two sine waves at 2 Hz and 3 Hz plus some random noise. We will be using the `np.fft` module to compute the Fourier transform with the two main functions below:

- `rfft` - computes the actual Fourier transform coefficients
- `rfftfreq` - tells us the frequencies for which we are computing the Fourier transform

We then examined `freqs` to see that the FFT samples the Fourier transform uniformly from 0 Hz to the Nyquist frequency, which in this case is 25 Hz because our sampling rate is 50 Hz. We also saw the Fourier transform coefficients and only examined the magnitudes. Plotting the FFT, we see that the signal is composed primarily of two frequencies (2 and 3Hz) and a little bit of everything else (from the random noise).

We also saw how zero-padding could be used to visualize sinusoids with frequencies that are not present in `freqs`. To visualize at frequencies not in `freqs`, we need to sample twice as often and we can do this by adding 0s to the end of the signal. However, we also see this rippling, which is an artifact of padding the signal with 0s, which is the trade-off when doing zero-padding.

We just learned about inverse Fourier Transform in which we used the `np.fft` module to compute the inverse Fourier transform with the function `irfft`.

We started with a noisy signal and removed all the frequency components not in the range, in this case, 2Hz and 3Hz. And we saw a recovered signal that looked very close to the signal we saw in the previous video.

But we did the process again from 2.15Hz and 2.95Hz. The recovered signal looked a bit distorted and not what we'd expect. This is because zeroing out Fourier coefficients is not the best way to filter a signal.

We then used `scipy` to **bandpass filter** our signal for us. A bandpass filter will remove all frequency components outside of a given passband. Let's bandpass filter our signal with a passband from 1 Hz to 4 Hz. This way, our desired frequencies of 2.15 Hz and 2.95 Hz are well within the passband. And now, our recovered signal looks very similar to what we want.

In this course, we will always bandpass our signals before processing them.

Fourier Transform In Review

In this class we will be computing the Fourier transform by using a method called the **Fast Fourier Transform**. This is a clever algorithm that is able to

compute the Fourier transform in O(n*log(n)) time instead of quadratic time. We use `numpy`'s implementation of this algorithm with the functions:

- `rfft`
- `rfftfreq`
- `irfft`

We then saw how we could remove the noise by filtering out frequency components outside of the bandwidth of the signal. The process of removing frequencies from a signal outside a specific band is known as **bandpass filtering**. The band of frequencies that we want to preserve is called the **passband**. -- or **bandpass filtering** our signal. We did this first by manipulating the signal in the frequency domain. After seeing the short-falls of this method, we explored using traditional bandpass filtering techniques that process the signal in the time-domain.

Further Resources

3Blue1Brown is a great YouTube channel that explains mathematical concepts with beautiful animations that make intuitive understanding so much easier. He has a few videos on the Fourier transform, which are absolutely illuminating. I highly recommend exploring this channel, starting with this video, **But what is the Fourier Transform? A visual introduction**.

New Vocabulary

- **Frequency-domain**: A representation of a signal over frequency instead of time. Instead of representing the signal as a series of numbers in time, the signal is represented by the frequency components that make it up.
- **Bandpass filter**: A function that preserves frequency components of a signal within a band and suppresses the frequency components outside that band.

Plotting Signals in the Frequency Domain

When we look at signals in the frequency domain, we lose information about the time-domain. Previously this hadn't been a problem because we were looking at signals whose frequency components did not change over time. They were **stationary**. However, most signals we deal with will not be

stationary. In this case, it is better to visualize the frequency components of a signal over time using either a:

- **Short-Time Fourier Transform**
- **Spectrogram**

New Vocabulary

- **Frequency component**: The Fourier transform explains a signal as a sum of sinusoids. Each of these sinusoids is a frequency component of the signal.
- **Stationarity**: A property of a signal where the statistics of a process generating a signal do not change in time. Generally, if the frequency components in a signal change in time, this signal is not stationary.

Harmonics

Real periodic signals are rarely sinusoidal. Still, we like to use the Fourier transform to learn about the periodicity of these signals. All periodic signals are composed of a **fundamental frequency**, which is the lowest frequency of the periodic signal, and integer multiples of this frequency called **harmonics**. In this lesson, we see this for ourselves and how the fundamental frequency relates to the signal in time-domain.

Recap: Intro to Digital Sampling & Signal Processing

We covered a lot of material in this lesson. We started with the basics of what a signal is and how to digitally sample one. Then we covered techniques to process digital signals, including interpolation, the Fourier Transform, and harmonics. Along the way, we learned how to visualize signals using matplotlib, and how the spectrogram or STFT can help us more accurately see the Fourier coefficients of a signal as they change in time. The breadth of concepts that we went over could comprise full university classes. We did a lot of exercises to help you feel comfortable using these topics practically, but a lot of this material might be unlike other disciplines of math that you've seen before. Don't hesitate to rewatch the videos and explore the further resources

to build your intuition. And I'm hopeful that as we use these concepts in the upcoming lessons, things will start to click more.

Further Resources

Physionet

Physionet is a great resource of freely available biomedical signals. You can try many of the techniques you learn in this class on datasets in **Physionet**. This **European ST-T Database** from Physionet was used in the previous exercise.

Plotting

- **Matplotlib** - the plotting library we use most in this course.
- **Seaborn** - a wrapper around `matplotlib` that makes it easier to do higher-level statistical visualization. We will use this a few times in the course.
- **Altair** - Another powerful visualization library in Python
- **Plotly** - You can use `plotly` to create and save visualization in HTML / javascript. This is especially useful when you want to make offline, shareable plots that you can interact with in the browser.

Interpolation

- **Interpolation**
- **Linear Interpolation**

Fourier Transform

3Blue1Brown is a great YouTube channel that explains mathematical concepts with beautiful animations that make intuitive understanding so much easier. He has a few videos on the Fourier transform, which are absolutely illuminating. I highly recommend exploring this channel, starting with **this video**.

Spectrograms

Plotting a spectrogram or visualizing the short-time Fourier transform are ways of balancing the trade-off between time resolution and frequency resolution. Surprisingly, this trade-off is related to the quantum uncertainty principle (see **this 3Blue1Brown video**). This tutorial is a great explanation of

this trade-off as well as a description of the **wavelet transform**, which is another solution to this problem.

Harmonics

Harmonics explain why different instruments sound different despite playing the same note. Check out **this video** from YouTube channel **12tone** for an explanation.

3Blue1Brown also has **a video** explaining why certain notes sound good together by looking at multiples of their frequencies.

Glossary

- **Transducer**: Part of a sensor that converts a physical phenomenon into an electrical one (e.g., voltage)
- **Analog-to-Digital Convert (ADC)**: A device (usually embedded in the sensor) that converts an analog voltage into an array of bits.
- **Bit depth**: The number of bits an ADC uses to create a sample. A 16-bit ADC produces a 16-bit number for each sample.
- **Noise floor**: The total amount of noise in the sensor, including electrical interference from the environment and other parts of the device, thermal noise, and quantization noise.
- **Dynamic range**: The physical range of the sensor. Values outside of this range will show up as clipping in the digital signal.
- **Sampling rate**: The frequency at which a sensor measures a signal.
- **Hz**: The units of the sampling rate. 1Hz means 1 sample per second.
- **Nyquist frequency**: Half of the sampling frequency. Signal components above this frequency will get aliased in the sampled signal.
- **Frequency component**: The Fourier transform explains a signal as a sum of sinusoids. Each of these sinusoids is a frequency component of the signal.
- **Aliasing**: The effect that causes frequency components greater than the Nyquist frequency to become indistinguishable from frequencies below the Nyquist frequency.
- **Bandwidth**: A range of frequencies within a band.

- **Interpolation**: A method for estimating new data points within a range of discrete known data points.
- **Resampling**: The process of changing the sampling rate of a discrete signal to obtain a new discrete representation of the underlying continuous signal.
- **Frequency domain**: A representation of a signal over frequency instead of time. Instead of representing the signal as a series of numbers in time, the signal is represented by the frequency components that make it up.
- **Time-domain**: The typical representation we are used to for signals where the signal is represented by values in time.
- **Bandpass filter**: A function that preserves frequency components of a signal within a band and suppresses the frequency components outside that band.
- **Passband**: The band of a bandpass filter where frequency components will be preserved.
- **Stationarity**: A property of a signal where the statistics of a process generating a signal do not change in time. Generally, if the frequency components in a signal change in time, this signal is not stationary.

Introduction to sensors

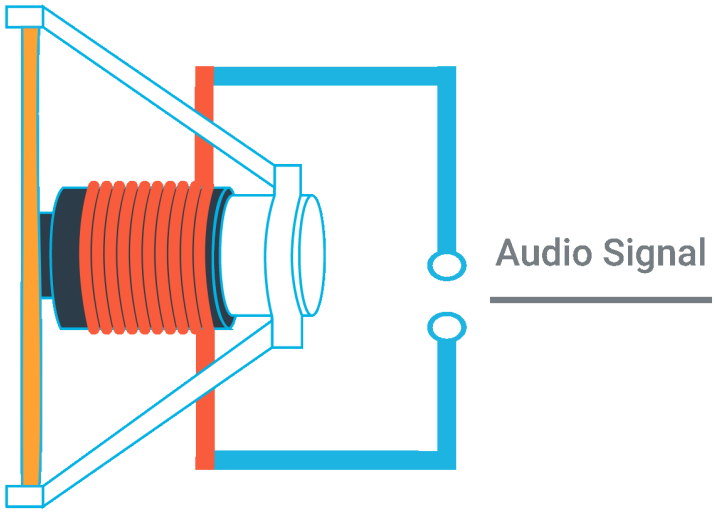
Introduction

Sensors are how we measure physical quantities in the world around us. The name of the game for building any kind of sensor is to convert a physical phenomenon into an electrical one - be it a current or voltage - which can then be digitized and stored on a computer. The component that does this is called the transducer.

This is easy to see in this example of how a microphone works. A microphone is basically a drum attached to a coil of wire around a magnet. The drum moves in response to vibrations in the air or sound. As you may have learned in your physics class when the coil of wire moves back and forth over the magnet, the changing magnetic field induces a current in the wire. The A/C

current creates an A/C voltage in the circuit that is then measured using an ADC.

Microphone Diagram



Specifics of how a sensor works is important for truly understanding the signal. You need to know how your accelerometer works to really understand what you're seeing when you're plotting an accelerometer trace or when thinking about how to design your algorithm.

In this lesson, we'll learn how the three sensors work:

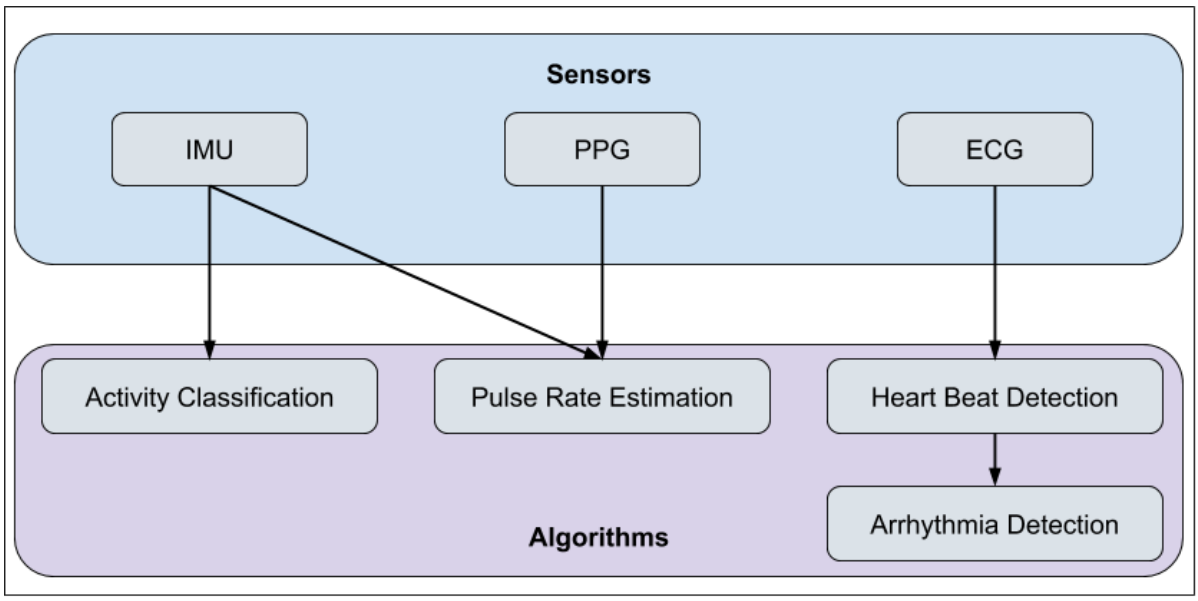
- Inertial Measurement Unit (IMU)
- Photoplethysmogram (PPG)
- Electrocardiogram (ECG)

You'll understand the physical quantity being measured, become familiar with what these signals look like, and learn about common challenges and noise sources that arise when working with these signals.

Lesson Concepts

This lesson will describe the sensors and raw signals they collect. Sensors enable a set of algorithms that can compute derived metrics from each of these signals or multiple in combination.

Sensor & Algorithms



Outline

- Wrist Wearable Demo
- IMU Sensor
  - IMU Overview
  - Accelerometer Deep Dive
- PPG Sensor
- ECG Sensor

Inertial Measurement Unit (IMU)

The **inertial measurement unit (IMU)** is an umbrella term for three specific sensors that describe motion, namely the accelerometer, gyroscope, and magnetometer.

- The **accelerometer** measures linear acceleration.
- The **gyroscope** measures angular velocity
- The **magnetometer** measures absolute orientation.

While an accelerometer might tell you that the device is moving to the left really fast, it won't tell you which way left actually is. The magnetometer will tell you that moving to the left means going East. Each of these sensors has 3

channels of measurements, each in a perpendicular direction in 3D space and would be labeled x, y, and z. This would be like measuring the acceleration up and down, left and right, and forward and backward. Device manufacturers can orient their accelerometers however they want, so we can't assume that the z-direction is the vertical direction.

We won't be dealing with gyroscopes and magnetometers in this course, but we will look at accelerometers. The animation below is a model of an accelerometer. There is a mass attached to springs and the mass moves a plate between two capacitors inside a circuit. This changes the capacitance depending on the location of the plate. Because the displacement of the mass on a spring is proportional to the force it sees, the changing voltage in the circuit is proportional to force and acceleration. If you combine three of these circuits together perpendicularly, you can measure acceleration in 3 dimensions.

Accelerometer Diagram

This also means that accelerometers are affected by gravity. For example, if we rotate our accelerometer 90 degrees, we can see gravity pulls the mass down and the accelerometer will see a voltage associated with 1 **g-force** or the magnitude of the acceleration due to Earth's gravity. The accelerometer only measures 0 if it's in free-fall.

This also means that when the device is stationary, you can measure its orientation. If all the acceleration is in the downward z-direction, then it's flat on a table. Or if it's all in the x-direction, then it's tilted on its side. We'll take a more in-depth look at this another phenomena in the next lesson where we look at accelerometer traces in detail.

Not all accelerometers are implemented using this capacitor attached to a mass on a spring model, but they follow similar principles. For example, some accelerometers use a force sensitive resistor or a **piezoelectric crystal** to modulate a voltage in response to an acceleration.

New Vocabulary

- **Inertial Measurement Unit (IMU)**: A collection of sensors that measure motion.
- **Accelerometer**: A sensor that measures linear acceleration.
- **Gyroscope**: A sensor that measures angular velocity.
- **Magnetometer**: A sensor that measures magnetic forces.
- **g-force**: The amount of acceleration on a body measured in units of acceleration due to gravity on earth (or roughly 9.8m/s^2).

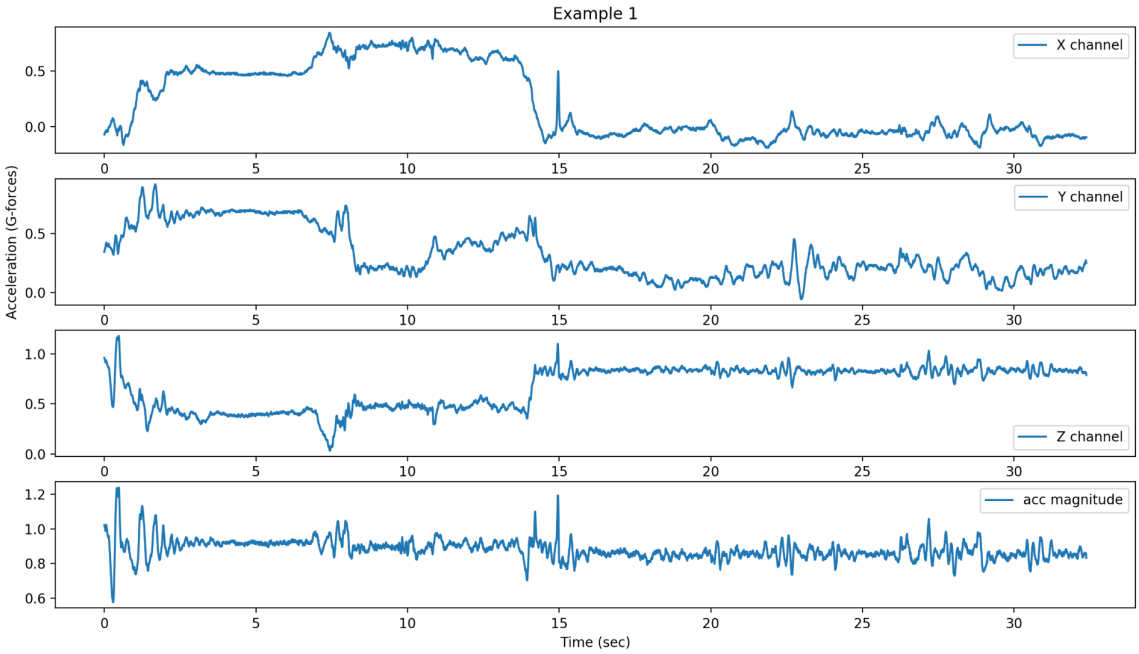
Accelerometer Deep-Dive

Let's take a more in-depth look at the accelerometer. As a reminder, when we talk about accelerometer magnitude, this means the vector magnitude of the force on the accelerometer in 3D space, as given by this formula.

Accelerometer magnitude is the square root of the sum of the squared value of movement in each of the 3 axes.

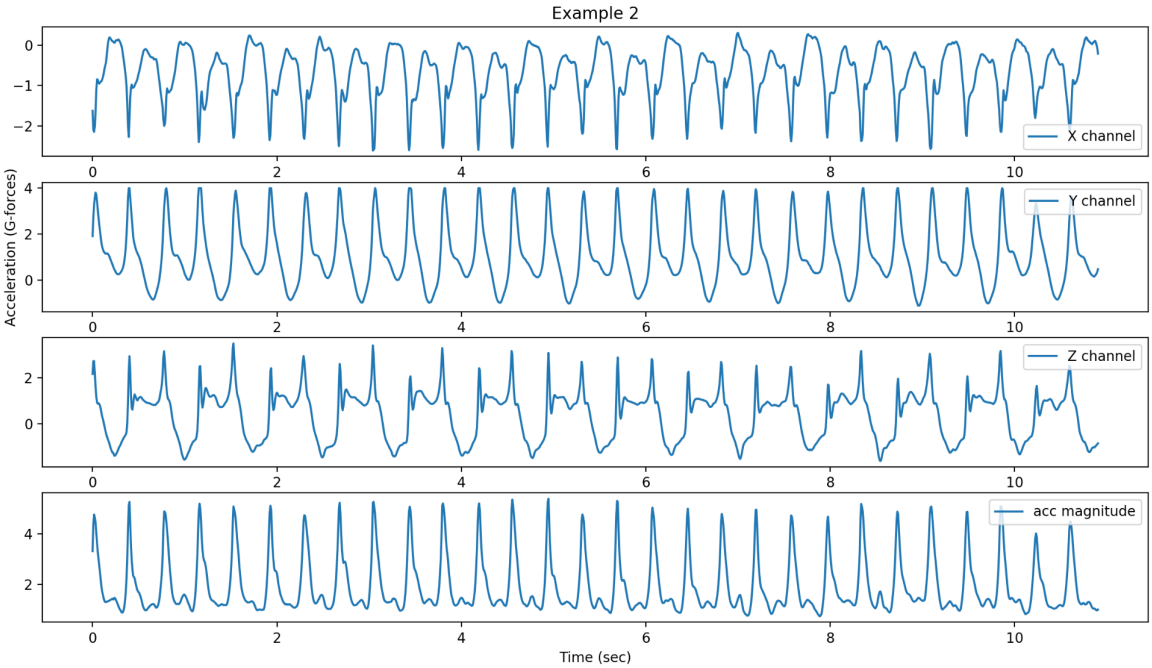
$$ACC_{mag} = \sqrt{ACC_x^2 + ACC_y^2 + ACC_z^2}$$

Take a look at this accelerometer trace when the wearer is at rest.



Accelerometer Magnitude - Wearer at Rest

The accelerometer magnitude is close to 1 g, which means the total force on the accelerometer is only due to gravity. But if you look at the individual channels, they shift around. In the beginning, they all see the same amount of gravitational force, which means the device is tilted along each axis equally. But then after 15 seconds, the x and y channels drop and the z-channel rises to nearly 1g. This indicates the device has moved and is now lying flat, so only the z-channel feels the force of gravity. By strategically placing accelerometers on the body, you can use this technique to figure out things like posture or sleeping position.



Accelerometer Magnitude - Wearer is Jogging

This is a trace of someone who is jogging. Notice the very periodic shapes in each channel that indicate the cadence of the arm swing, which can be used to figure out how fast someone is walking. By analyzing the specific characteristics of the waveform, you can decompose each stride into various components.

For example, this paper decomposes the accelerometer trace from an ankle into various phases of the gait cycle.

Patterson M., Caulfield B. A novel approach for assessing gait using foot mounted accelerometers; Proceedings of 5th International ICST Conference on Pervasive Computing Technologies for Healthcare; Dublin, Ireland. 23–26 May 2011; pp. 218–221.

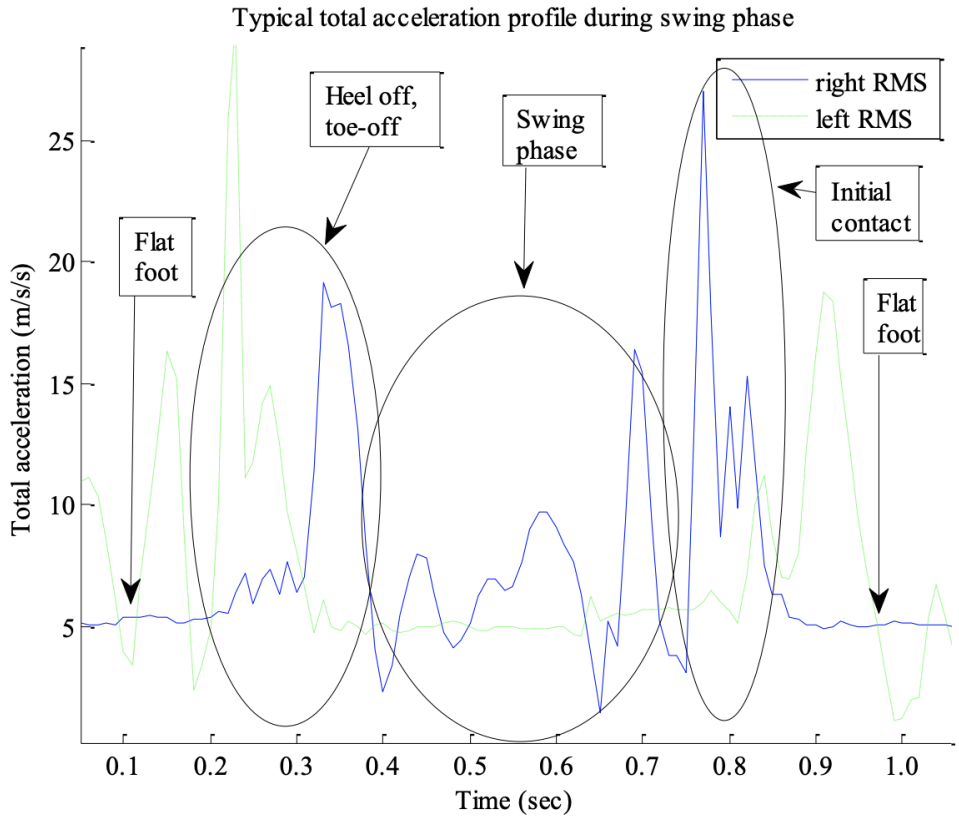
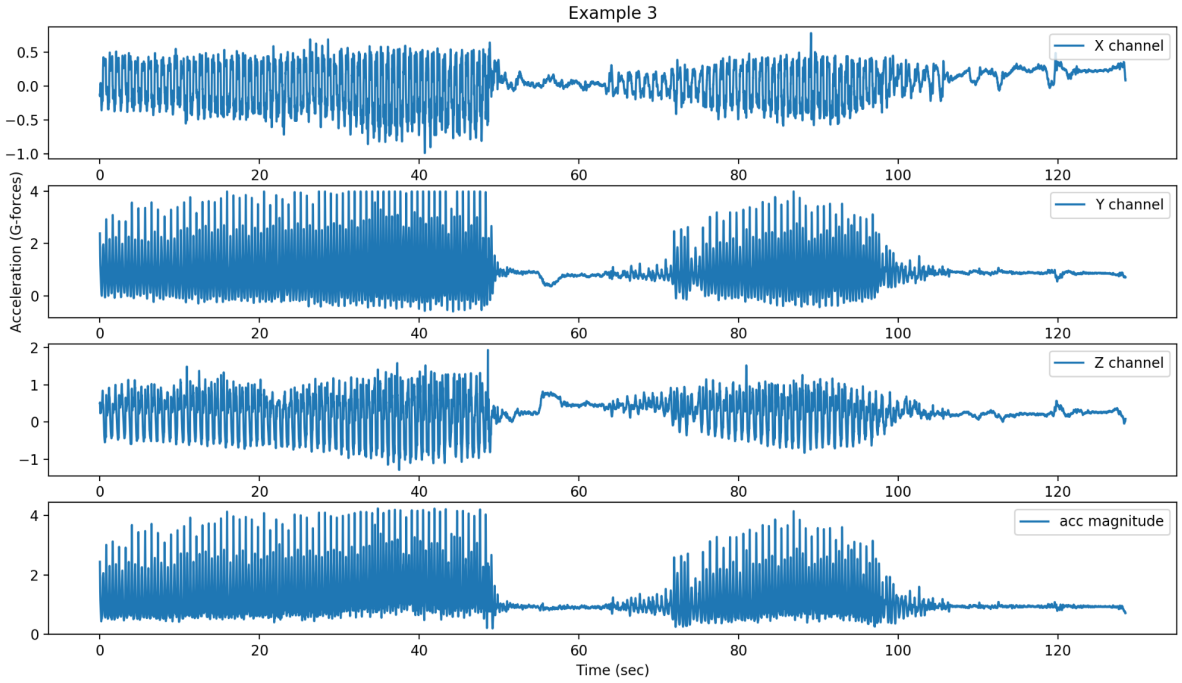


Figure 1. Total acceleration curve from an accelerometer mounted on the foot of a healthy subject.

Discriminating the gait phases like this is an important starting point for several applications such as recovery after an injury or treatment, the classification of daily activities, coaching athletes, and distinguishing between normal and pathological gait. Researchers have found that differences in each person's gait mean that it can be used to identify individuals.



Accelerometer Magnitude - Wearer is running with a break in between

Here we see a zoomed-out view of the accelerometer trace over 2 minutes. We can see the wearer distinctly changing activities, going from jogging to still to jogging again. You can even tell that they are probably jogging on a treadmill as the speed slowly ramps up and down. And from the sharp cut-off around 50 seconds, maybe we can tell that they jumped off the treadmill or for some reason their treadmill session was cut short. I didn't collect this dataset, so we can't know for sure, but it's interesting that we can learn this level of detail from just an accelerometer alone.

Exercise 1: Step Cadence

Further Resources

This paper describes using the orientation of accelerometers on the body to determine posture. **Gjoreski, Hristijan & Gams, Matjaz. (2011). Activity/Posture Recognition using Wearable Sensors Placed on Different Body Locations. 10.2316/P.2011.716-067.**.

A review of methods for segmenting the gait cycle.**Taborri J, Palermo E, Rossi S, Cappa P. Gait Partitioning Methods: A Systematic Review. Sensors (Basel). 2016 Jan 6;16(1). doi: 10.3390/s16010066. Review. PubMed PMID: 26751449; PubMed Central PMCID: PMC4732099**

Gait cycle segmentation using an ankle based accelerometer.**Patterson, Matt & Caulfield, Brian. (2011). A novel approach for assessing gait using foot mounted accelerometers. 2011 5th International Conference on Pervasive Computing Technologies for Healthcare and Workshops, PervasiveHealth 2011. 218 - 221. 10.4108/icst.pervasivehealth.2011.246061.**.

An article describing how changes in the gait cycle can be learned behavior. In this case, from prior KGB training.**Araújo R, Ferreirai JJ, Antonini A, and Bloem B. (2015) "Gunslinger's gait": a new cause of unilaterally reduced arm swing. The BMJ.**

Nomenclature for different granularity of gait phases.

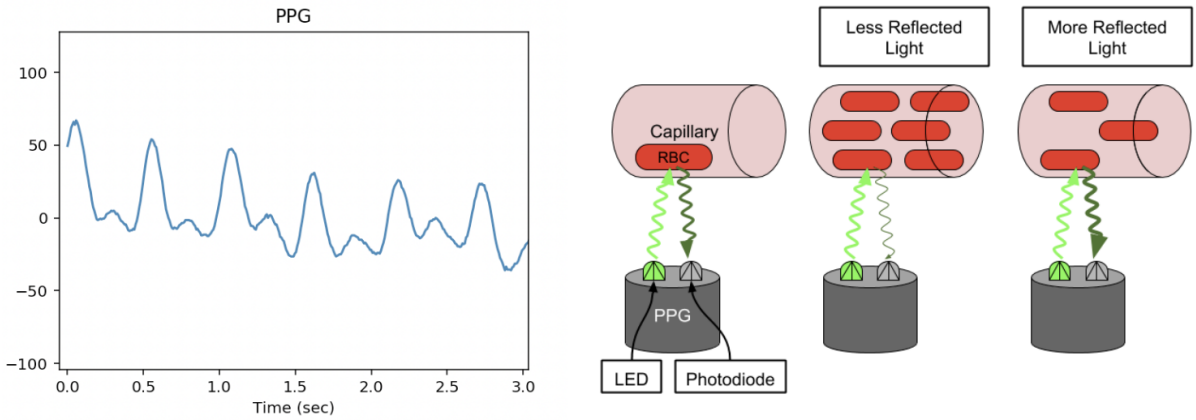
Granularity	Gait Phases								
Two Phases	Stance					Swing			
Three Phases	First Rocker	Second Rocker				Swing			
Four Phases	Heel Strike		Flat Foot		Heel Off	Swing			
Five Phases	Heel Strike		Flat Foot	Heel Off	Toe Off	Swing			
Six Phases (a)	Initial Contact	Loading Response	Mid Stance	Terminal Stance	Pre Swing	Swing			
Six Phases (b)	Loading Response		Mid Stance	Terminal Stance	Pre Swing	Swing 1	Swing 2		
Seven Phases	Loading Response		Mid Stance	Terminal Stance	Pre Swing	Initial Swing	Mid Swing	Terminal Swing	
Eight Phases	Initial Contact	Loading Response	Mid Stance	Terminal Stance	Pre Swing	Initial Swing	Mid Swing	Terminal Swing	
Gait [%]	0					60			100

Source:Taborri J, Palermo E, Rossi S, Cappa P. Gait Partitioning Methods: A Systematic Review. Sensors (Basel). 2016 Jan 6;16(1). doi: 10.3390/s16010066. Review. PubMed PMID: 26751449; PubMed Central PMCID: PMC4732099

Photoplethysmogram (PPG) Sensor

The **photoplethysmogram (PPG)** optically measures blood flow at the wrist. The LEDs in a PPG sensor shine a typically green light into your skin and your

red blood cells absorb that green light. The reflected light is then measured by the **photodetector**. When your heart beats and blood perfuses through the wrist, there are more red blood cells that absorb the green light and the photodetector sees a smaller signal. As the heart fills back up with blood and blood leaves your wrist, the more green light is reflected back and the photodetector reading goes up. This oscillating waveform can be used to detect pulse rate. See the illustration below.



PPG Sensor Diagram

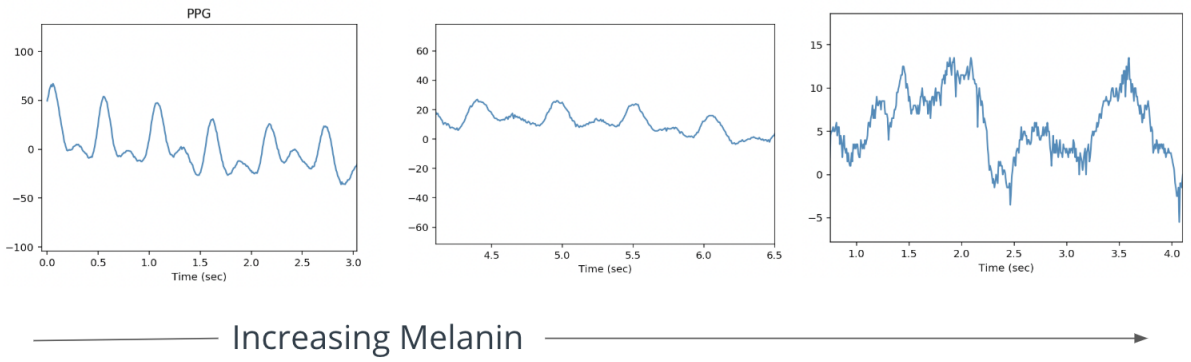
The trough corresponds to a peak in perfusion and occurs when the ventricles of the heart contract, called **systole**. The peak in the waveform occurs when there is the least amount of blood in the wrist or when the heart relaxes and fills with blood, called **diastole**.

Noise Sources

There are many other factors that affect the signal.

Melanin

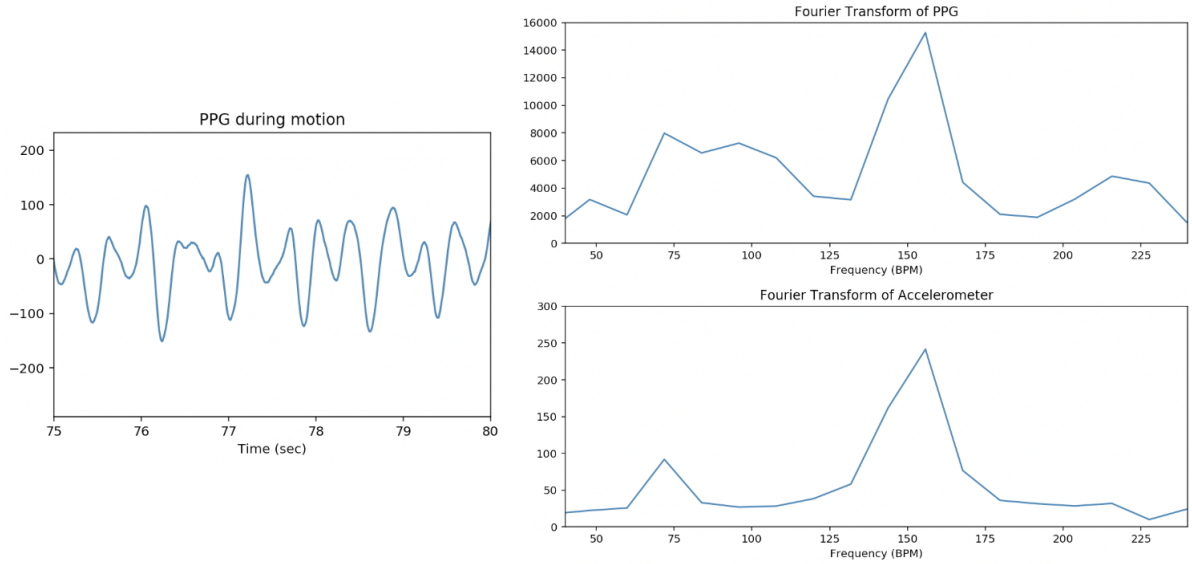
Skin absorbs light from the LEDs as well and darker skin absorbs more light. This causes a DC shift and a reduction in SNR in the PPG signal for people with darker skin.



Effect of Melanin

Arm Motion

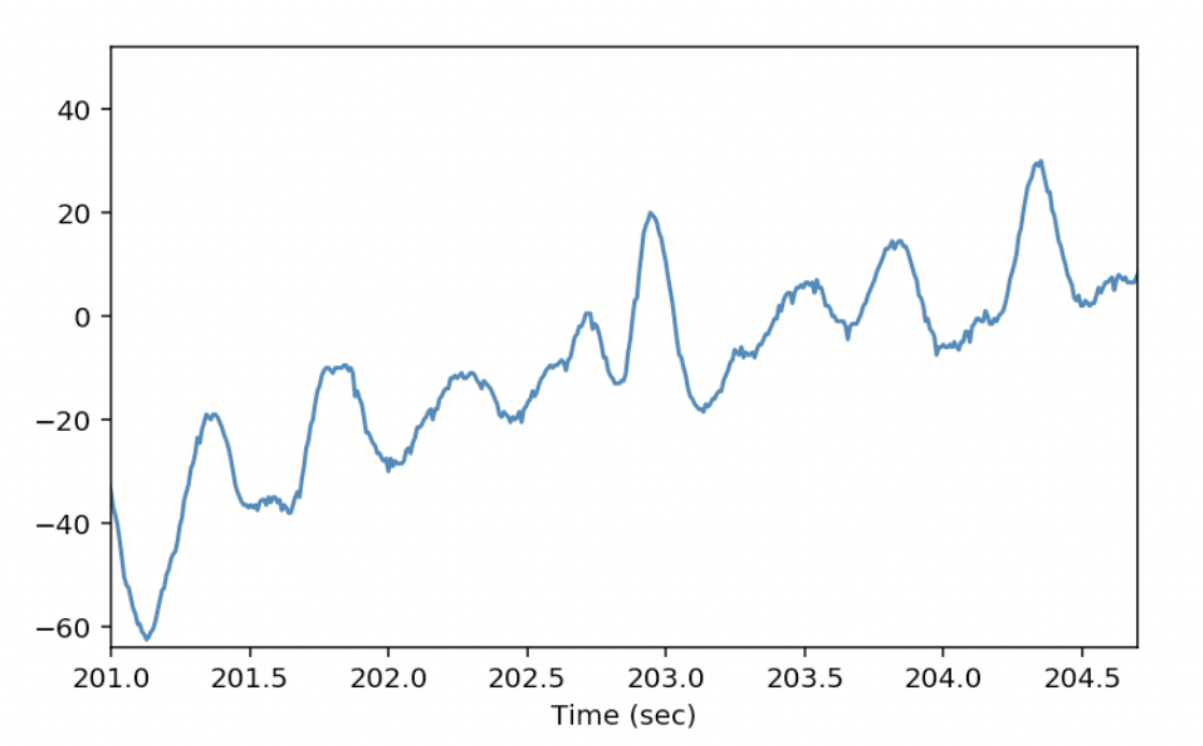
Blood is a liquid and when you move your arm around, the blood inside moves as well. This is what the PPG signal looks like while running. You can see the cadence of the arm swing in the FFT of the PPG signal as well as in the accelerometer.



Effect of Arm Motion

Arm Position

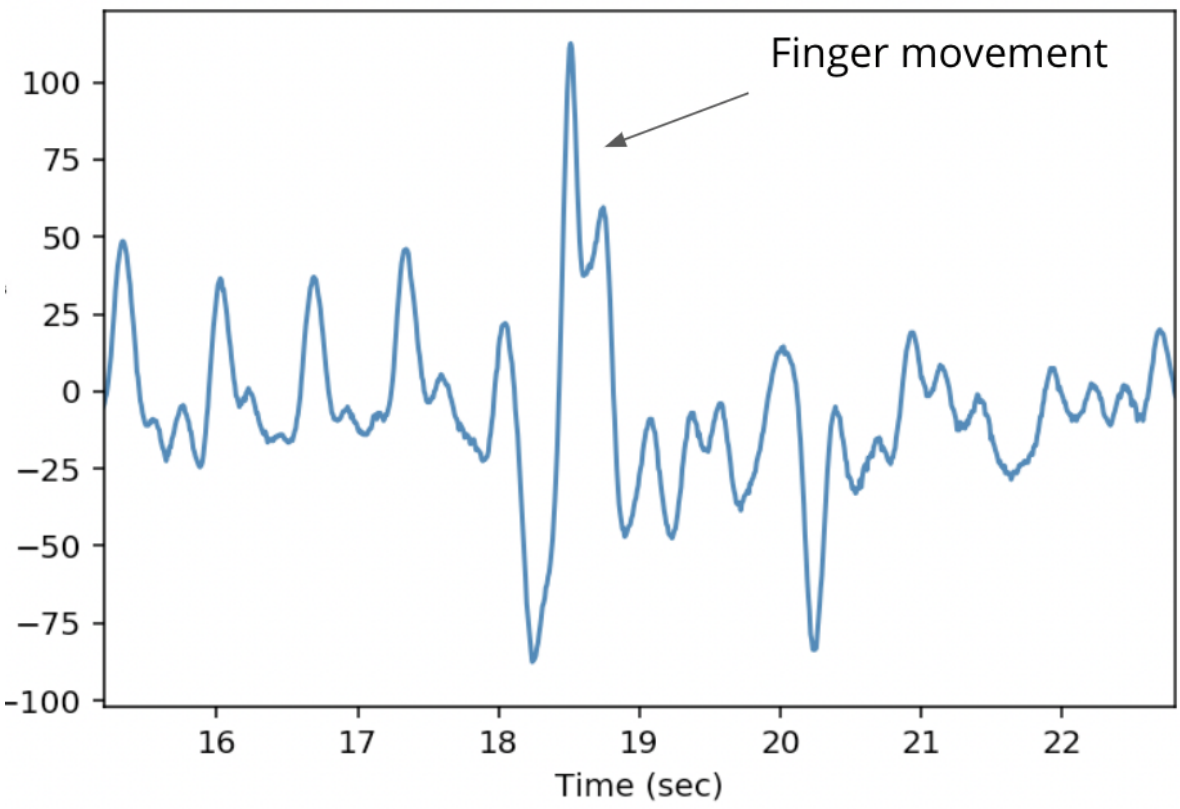
Even while at rest, if the position of the arm changes, blood flows into or out of the wrist. If you hang your arm down, more blood will flow into it and the DC level of the signal will slowly drop. If you raise your arm up, blood will flow out of your arm and the DC level will increase.



Effect of Posture Change

Finger Movement

Moving your fingers around will cause a significant disturbance in the PPG signal because, as the tendons in your wrist move, they cause other structures to move around and may even shift the position of the sensor. All this motion will change the path that light takes to travel through the wrist and change the photodetector reading.



Effect of Finger Movement

All these sources of noise make the PPG signal tricky to deal with, but now that you are aware of them, you can design algorithms that are robust to these events.

New Vocabulary

- **Photoplethysmogram (PPG)**: the optical sensor used to measure pulse rate on a wearable device.
- **Photodetector**: A sensor that measures light.
- **Diastole**: The phase of the cardiac cycle where the heart relaxes and fills with blood.
- **Systole**: The phase of the cardiac cycle where the ventricles contract and pump blood through the arteries.

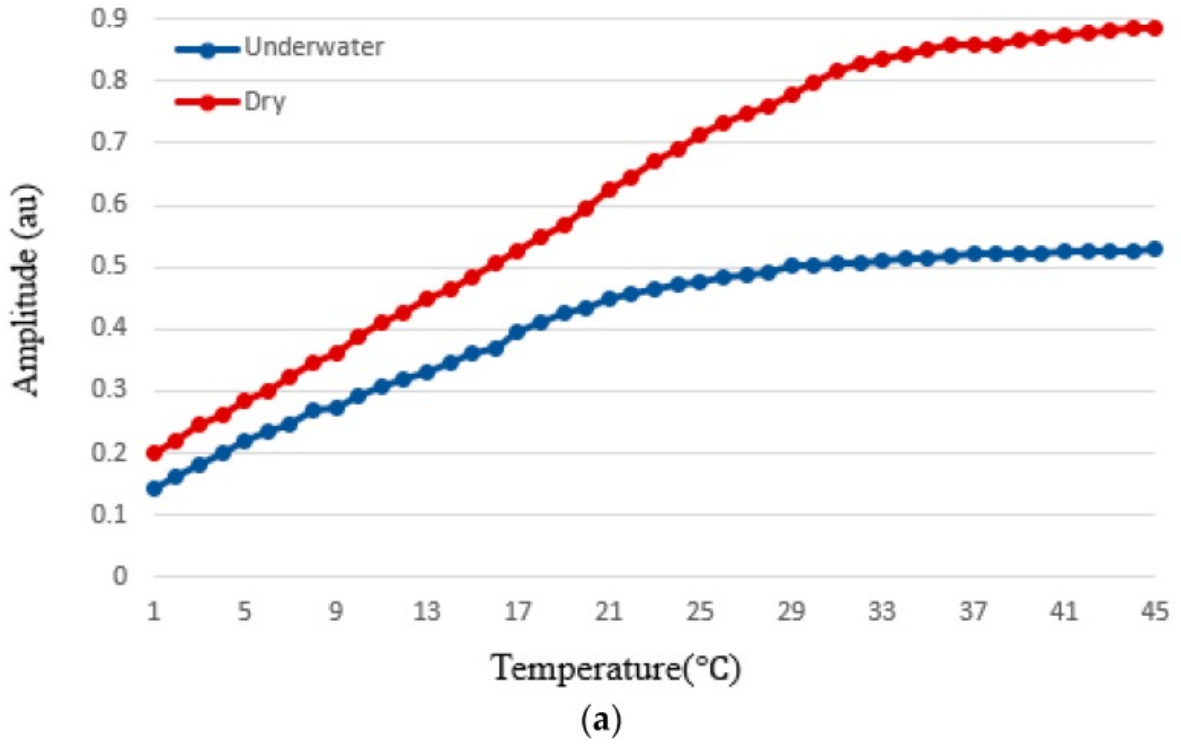
Signal Quality Evaluation

The single most important factor that can make or break any algorithm you build is the underlying quality of the signal. As an algorithms engineer you may

have the rare opportunity to influence the design of the hardware that will acquire the signal that will then be the input to your algorithms. Being able to provide actionable quantitative feedback for various hardware prototypes will provide endless returns down the line when you start building algorithms based on that hardware.

We can see an example of this kind of analysis in **this paper** where the researchers analyzed the effects of different mediums (water vs. air) and temperature on the PPG signal quality and pulse rate estimation from a smartphone sensor.

First let's look at the pulse rate algorithm accuracy.

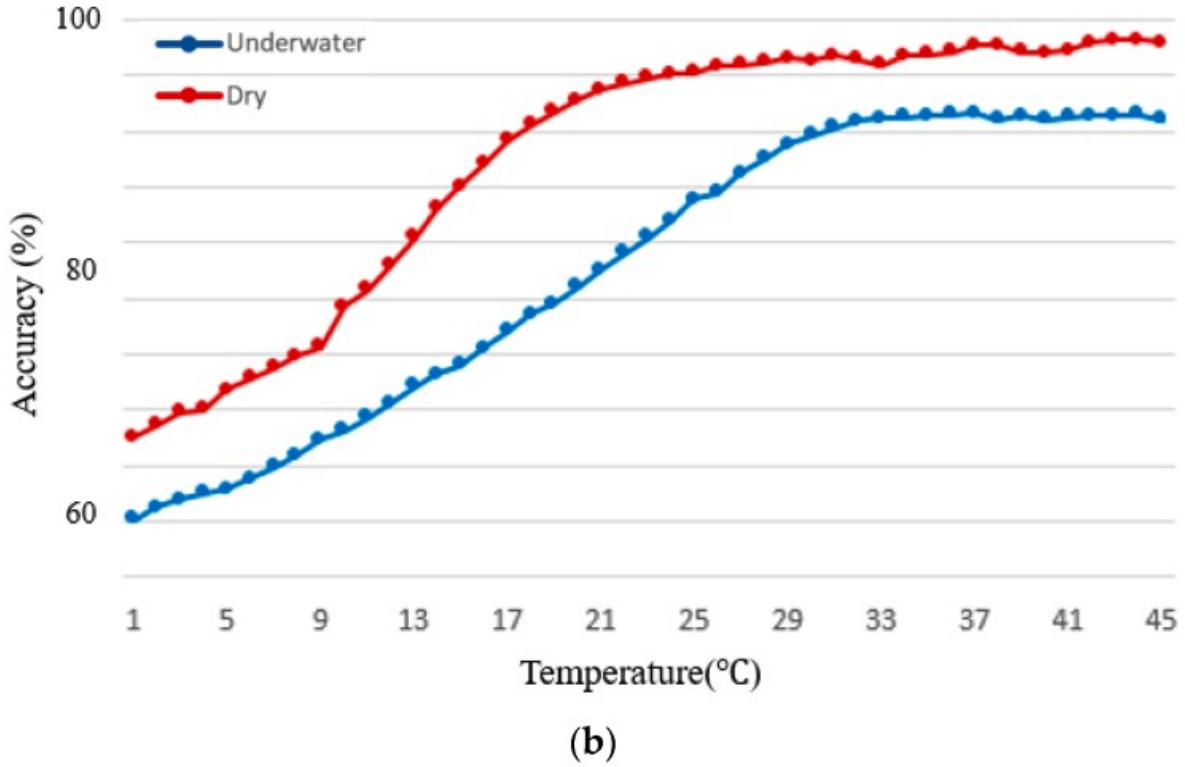


Pulse rate accuracy acquired from smartphone PPG signals for varying temperature in dry (red) and underwater (blue) environments.

Pulse rate accuracy deteriorates at cooler temperatures and is worse underwater compared to in air.

Using a pulse rate algorithm's accuracy is a great way to evaluate signal quality, especially if the primary thing you want to do with that signal is estimate pulse rate and if you have the algorithm on hand. However, using an algorithm to estimate signal quality can conflate the algorithm accuracy and its idiosyncrasies with the signal quality. Sometimes you may want to look

directly at the signal characteristics themselves. Here, the researchers chose to look at signal amplitude.



Smartphone PPG signal amplitude for varying temperature in dry (red) and underwater (blue) environments.

Again we observe that we get higher amplitudes in dry and warmer environments. This plot implies that a higher signal amplitude corresponds to higher signal quality. But as we have seen previously, with PPG signals, this is not always the case. Motion artifact and ambient light can cause high amplitude peaks in the signal that definitely do not correspond to signal quality. Optimizing for signal amplitude might mean that you build hardware that is extremely susceptible to motion artifacts.

In the following exercise you will use a more holistic metric to evaluate PPG signal quality called the signal-to-noise ratio that avoids some of these problems.

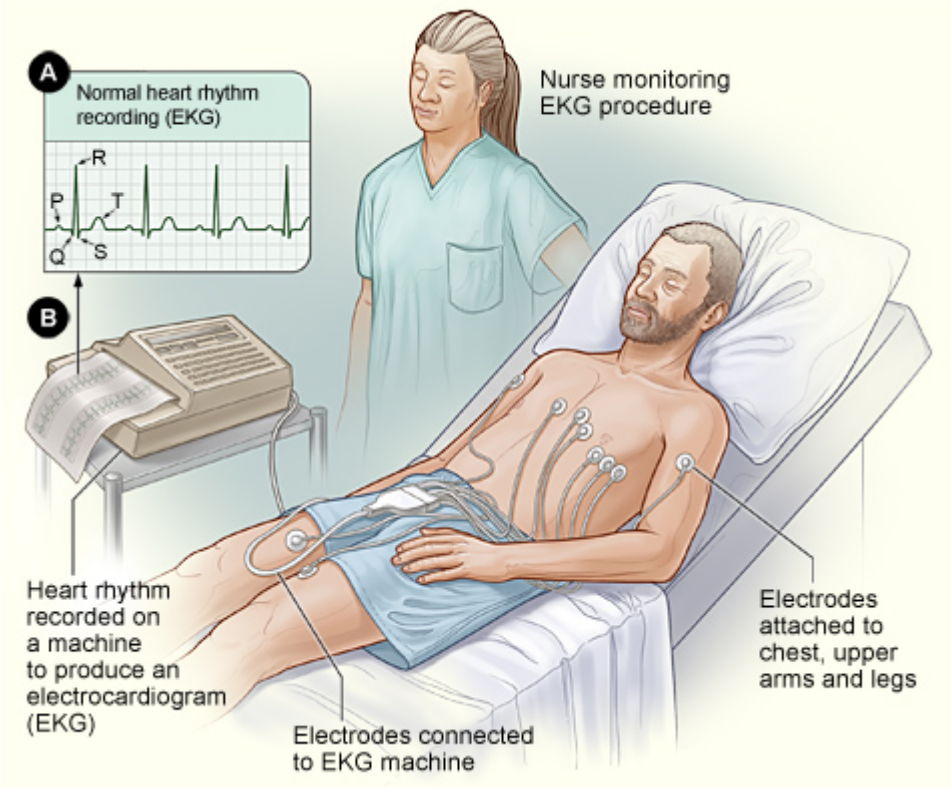
Electrocardiogram (ECG) Sensor

Electrocardiograms (ECG or EKG) measure the voltage across the chest, which is primarily due to the heart rhythm. ECGs are different from the other raw signals produced by a wearable in that cardiologists can use them in their

raw, unprocessed form to diagnose heart problems like a previous heart attack or an irregular heart rhythm.

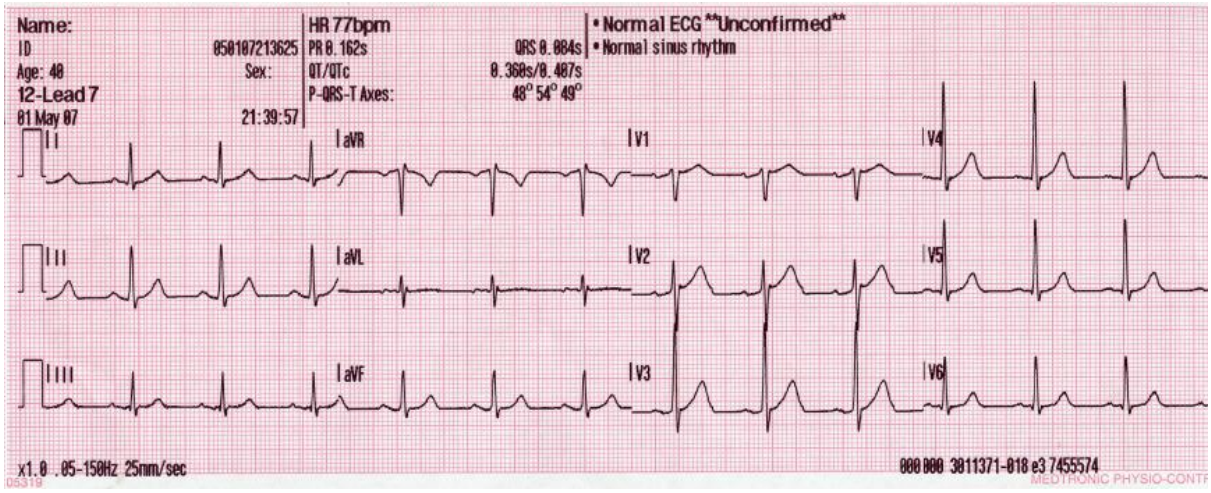
The ECG measures the voltage created by the electrical activity in the heart. A traditional ECG does this with electrodes on the chest. An **electrode** is a conductive pad that comes in contact with skin and a **lead** is the electrical potential difference between two electrodes.

Clinic ECG



Each pair of electrodes measures a slightly different voltage. The standard 12-lead ECG contains 12 different voltages across different axes of the body.

12 Lead ECG



The ECG has been “somewhat” wearable for a while. **Holter monitors** are ECG machines that are connected to a person’s chest in much the same way a tabletop machine is, but the recording device is portable and attached to the wearer’s hip. Holter monitors are worn during stress tests after a patient has been diagnosed or is suspected of having a heart condition or for short-term monitoring. They’re typically not worn for more than a few days.

Beyond that, there are chest patches that only measure 1-lead of the 12 lead ECG. These chest patches can last from a few days to a couple of weeks. In 2011, Alivecor started making a handheld device that paired with an iPhone to take an ECG. Apple Watch and Verily’s Study Watch both have ECG sensors embedded into the hardware. On a wristwatch, the two electrodes are moved to the wrist and fingers on opposing hands. Because of this, the ECG cannot be monitored continuously and passively like the chest patch or Holter monitor or other sensors on a wearable. However, a wrist-watch can be worn for years, while a chest patch only lasts for a few weeks.

In lesson 4, we will go over the physiology of the heart and the characteristics of the ECG wave in greater detail.

Further Resources

You can learn more about the ECG in the **Wikipedia Entry**

New Vocabulary

- **Electrocardiogram (ECG)**: A sensor that measures the electrical activity of the heart.

- **Electrode**: Part of an ECG circuit. Conductive pads that make contact with the skin.
- **Lead**: An electrical potential difference measured across two electrodes of an ECG circuit.
- **Holder monitor**: A mobile ECG device that can measure continuously for typically 24 - 72 hours.

The Apple Heart Study - Revisited

Summary

Wearables allow us to collect more data under more conditions from more people. Typically we'd only measure ECGs from people that are suspected of having heart conditions. Now Apple Watch can measure and look for atrial fibrillation from its large, mostly healthy population. More research needs to be done into AF itself to really know how useful detecting it in this population actually is. For example, what does it mean when someone under 40 years old has AF. Are they actually at an increased risk of stroke? The Framingham study -- which found the association between AF and stroke -- was conducted in the 40s, 50s, and 60s when AF was diagnosed after coming to the hospital and taking an ECG. If a subject's AF was not persistent enough to show up at that time, then it would go undiagnosed. In fact, for a lot of the stroke victims, it was unknown that they even had AF until after they came to the hospital for the stroke itself.

Although the population of Framingham study was at a 4-5X increase in the risk of stroke from AF, it is unclear if the population that the Apple Watch is discovering AF for is also at this increased risk. What does it mean to the public health system when you're alerting all these people of AF just because you can monitor it more closely than you ever could before? And how should patients and their doctors respond when they discover AF? The Apple Heart Study paper mentions these concerns explicitly:

> ...further studies are needed to better understand the public health implications of identifying irregular pulse in persons younger than 40 years of age.

and

> ...uncertainty remains about the benefits of diagnosing and treating asymptomatic atrial fibrillation...

We have to address these concerns before we can really understand the value of using the Apple Watch as a screening or diagnostic tool.

Recap

Summary

Every time you interact with a new sensor type, it's a good idea to become familiar with the sensor in this way. You should try to understand the physics that make the sensor work, explore real data from the sensor, and research what common noise sources are and the artifacts they produce in the signal. Seeing how your own body reacts with a sensor is a great way to get experience and build intuition when possible. Finally, examining the existing literature on your sensor and the conditions that you are using it in is critical for learning a larger picture of the sensor's characteristics.

In this lesson, we learned how the three sensors we're using in this course -- the IMU, PPG, and ECG sensors -- work. You now know what physical quantities are measured by each sensor, know what each sensor looks like, and learned about common challenges and noise sources that arise when working with these signals.

Outline

- Wrist Wearable Demo
- IMU Sensor
  - IMU Overview
  - Accelerometer Deep Dive
- PPG Sensor
- ECG Sensor

Further Resources

IMU

- This paper describes using the orientation of accelerometers on the body to determine posture. **Gjoreski, Hristijan & Gams, Matjaz. (2011). Activity/Posture Recognition using Wearable Sensors Placed on Different Body Locations. 10.2316/P.2011.716-067.**.
- A review of methods for segmenting the gait cycle. **Taborri J, Palermo E, Rossi S, Cappa P. Gait Partitioning Methods: A Systematic Review. Sensors (Basel). 2016 Jan 6;16(1). doi: 10.3390/s16010066. Review. PubMed PMID: 26751449; PubMed Central PMCID: PMC4732099**
- Gait cycle segmentation using an ankle based accelerometer. **Patterson, Matt & Caulfield, Brian. (2011). A novel approach for assessing gait using foot mounted accelerometers. 2011 5th International Conference on Pervasive Computing Technologies for Healthcare and Workshops, PervasiveHealth 2011. 218 - 221. 10.4108/icst.pervasivehealth.2011.246061.**.
- An article describing how changes in the gait cycle can be learned behavior. In this case, from prior KGB training. **Araújo R, Ferreirai JJ, Antonini A, and Bloem B. (2015) "Gunslinger's gait": a new cause of unilaterally reduced arm swing. The BMJ.**

PPG

- To learn more about the PPGs, you can look at this **Wikipedia Entry**
- A dissertation on PPG processing techniques that includes an introduction that has great background information on the PPG signal in general. **Schäck, Tim. Photoplethysmography-Based Biomedical Signal Processing Darmstadt, Technische Universität Darmstadt, Jahr der Veröffentlichung der Dissertation auf TU prints: 2019 Tag der mündlichen Prüfung: 21.01.2019.**.

ECG

- You can learn more about the ECG in the **Wikipedia Entry**
- Note: We'll cover the ECG in more detail in lesson 4.

Glossary

- **Inertial Measurement Unit (IMU)**: A collection of sensors that measure motion.
- **Accelerometer**: A sensor that measures linear acceleration.
- **Gyroscope**: A sensor that measures angular velocity.
- **Magnetometer**: A sensor that measures magnetic forces.
- **g-force**: The amount of acceleration on a body due to gravity.
- **Photoplethysmogram (PPG)**: the optical sensor used to measure pulse rate on a wearable device.
- **Photodetector**: A sensor that measures light.
- **Diastole**: The phase of the cardiac cycle where the heart relaxes and fills with blood.
- **Systole**: The phase of the cardiac cycle where the ventricles contract and pump blood through the arteries.
- **Electrocardiogram (ECG)**: A sensor that measures the electrical activity of the heart.
- **Electrode**: Part of an ECG circuit. Conductive pads that make contact with the skin.
- **Lead**: An electrical potential difference measured across two electrodes of an ECG circuit.
- **Holter monitor**: A mobile ECG device that can measure continuously for typically 24 - 72 hours.

Activity Classification

Intro to Activity Classifiers

In this lesson, we are going to build an activity classifier using data from an accelerometer from a wrist wearable. Activity classifiers can be useful directly in that people like to keep track of the activities they are doing over the day. But they can also be used in more clinical contexts. For example, if a company is doing a drug trial and wants to know if their drug makes study subjects

more or less active, they can look at the activity classifier output and see if subjects are spending more time walking around or if they are mostly idle.

To build this activity classifier, you will learn how to featurize a high-rate time-series signal. How do you take a few seconds of a 200Hz signal -- so that's 1000s of data points -- and reduce it to a handful of features that traditional machine learning models know how to deal with. We'll then use a random forest model to train our activity classifier and use leave-one-subject out cross-validation to evaluate its performance. We'll then talk about our model's hyperparameters and do hyperparameter optimization. To be successful in any modeling task, however, we need to dive into our data and become familiar with its intricacies, so we're going to begin with some data exploration.

Outline

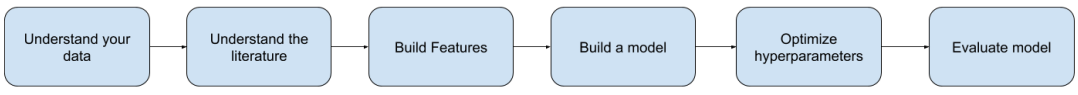
Understanding Your Data

- Wrist PPG Dataset
- Data Exploration and Visualization Understanding The Literature
    - Feature Engineering and Extraction Modeling Performance Evaluation Hyperparameter Optimization

Concepts

We will follow an algorithm development process, as outlined below.

Algorithm Development Process



Before jumping into a new domain, we first need to understand the data and the literature. For this lesson, we will be building an algorithm that first featurizes the raw high-rate IMU time series into a manageable number of data points. We can then put these features into a classical machine learning model, optimize the hyperparameters, and evaluate our performance.

Data Exploration

We just went over the dataset we will be using throughout this lesson from PhysioNet. We learned from reading the **Experimental Protocol** that there is PPG, ECG, 3-channel accelerometer data. And the 8 participants are doing the 4 following activities:

1. Walking
2. Jogging
3. Biking Lower Resistance
4. Biking Higher Resistance

For this lesson we will be focused on accelerometer data for building our activity classifier.

Summary

The dataset we will be using comes from the **Wrist PPG During Exercise** dataset on Physionet. Physionet is a great resource for biomedical time-series signals. It has many datasets with various signals in various disease conditions and all the data is available under an open license. The Wrist PPG dataset contains data from 8 subjects. The protocol contained four activities -- walking on a treadmill, running on a treadmill, low-intensity cycling on an exercise bike, and high-intensity cycling. While the dataset contains ECG and PPG signals as well, we will only be using the accelerometer signal for this project.

We explore the summary statistics of this dataset and look at the raw data. Like most real datasets, this one is not perfect. It is unbalanced in terms of numbers of subjects as well as the total number of data points.

Further Resources

- **Wrist PPG Dataset**
- This is a great blog post by **Casie Kozrykov** who taught me statistics at Google! In it, she describes the dangers of overfitting your brain when you explore your data.
- **Your dataset is a giant inkblot test**

- Check out **this StackOverflow discussion** on the value of data exploration. From one of the responses:

  > Two weeks spent training a neuralnet can save you 2 hours looking at the input data

- And finally, a **blog post** from a machine learning practitioner on the data exploration.

Feature Extraction

In the previous exercise we thought about how to distinguish between the types of activities based on the data but because of the limited set of data it's important to recognize that we've also just overfit ourselves to the dataset. When we did the data exploration, we looked at the entire dataset and now our brains might pick up on patterns that are only specific for that dataset. And it almost happened to me and we looked at how we might have distinguished biking from walking but there was 1 subject's walking data that looked much more like biking data rather than walking. It is safer to use literature based features, particularly when on a limited dataset. Other researchers use different datasets so it is very unlikely to be overfit to your dataset.

Summary

With the previous exercise, we've started to think about the data and how we might build features to separate the signals into activity classes. And while that's a great exercise, it's important to recognize that we've also just overfit ourselves to the dataset. When we did the data exploration we looked at the entire dataset and now our brains might pick up on patterns that are only specific for that dataset. This almost happened to me, in fact. One of my observations was that the x and z channel overlap for walking and not for biking. However, this isn't true for S6. There could easily have been a dataset where S6 did not participate, and I would have fully believed that x and z acc channels overlap when people are walking. If I had put this feature in my model, it would have not generalized to S6 and the performance of my model would have significantly degraded.

For small datasets like this, looking at the literature of existing features is a great way to avoid overfitting because the researchers who have come up with these features have looked at different datasets from ours. We are going to use features described in:

- Mehrang S., Pietilä J., Korhonen I. An Activity Recognition Framework Deploying the Random Forest Classifier and A Single Optical Heart Rate Monitoring and Triaxial Accelerometer Wrist-Band. Sensors. 2018;18:613. doi: 10.3390/s18020613. **Link**
- Liu S, Gao RX, Freedson PS. Computational methods for estimating energy expenditure in human physical activities. Med Sci Sports Exerc. 2012;44:2138–2146. doi: 10.1249/MSS.0b013e31825e825a. **Link**

We describe some of these features in an IPython notebook and leave some of the implementation to you in the following exercise!

Feature Extraction Continued...

We computed a partial list of our features but this needed to be done for you for each of the accelerometer channels in `activity_classifier_utils.py`.

Summary

Sampling a sensor hundreds of times per second means that raw sensor data has huge dimensionality. There are 7680 points of accelerometer data at 256 Hz over 10 seconds. Trying to model data points of this size will not be successful. We need to do dimensionality reduction.

By selecting features from the literature, we can be confident that we are not using features that overfit to our dataset and that the activity classifier we build will generalize to other studies and devices.

Check out the notebooks for this lesson and `activity_classifier_utils.py` to see how we implement the features and compute them for our dataset.

Further Resources

This blog post goes through a very similar process as this lesson. It starts by explaining some signal processing techniques (like we did earlier in the course). The author uses those techniques to build features in much the same way we just did. And then, he uses those features to build an activity

classification model, just as we are about to! **Machine Learning with Signal Processing Techniques**

Literature

The algorithm we built was inspired by these two papers.

- Mehrang S., Pietilä J., Korhonen I. An Activity Recognition Framework Deploying the Random Forest Classifier and A Single Optical Heart Rate Monitoring and Triaxial Accelerometer Wrist-Band. Sensors. 2018;18:613. doi: 10.3390/s18020613. **Link**
- Liu S, Gao RX, Freedson PS. Computational methods for estimating energy expenditure in human physical activities. Med Sci Sports Exerc. 2012;44:2138–2146. doi: 10.1249/MSS.0b013e31825e825a. **Link**

Activity Classification

Now that we've explored the data, examined the literature, chosen our features, and pre-processed all the data. Now it's time to finally build the classifier!

First off, we will do feature extraction to train on 10 second long non-overlapping windows. And we used sklearn to build a random forest classifier to classify our data. Then we defined the hyperparameters with 100 trees where each tree has a maximum depth of 4.

Now we are ready to build and train the model!

But as we just trained on the whole dataset we can't easily evaluate it. But one way to evaluate the performance of a multi-class classifier is to look at a confusion matrix. The confusion matrix shows how many data points were misclassified and what they were misclassified as.

Summary

We've explored the data, examined the literature, chosen our features, and pre-processed all the data. Now it's time to finally build the classifier!

In this lesson, we finally train our features to build a random forest model. We talk about model performance and use **cross-validation** to estimate our accuracy. We end up with a model with an overall **classification accuracy** of

73%, which is the percent of correct classifications made by the model. But don't fret, we'll do better in the next video!

Notebook Review

If you wanted to interact with the notebook in the video, you can access it **here** in the repo `/activity-classifier/walkthroughs/activity-classifier/` or in the workspace below.

The dataset that will be used throughout this lesson can be found at the top of the lesson directory at `/activity-classifier/data/`.

Menu

Expand

Further Resources

Random forests are boosted decision tree models. You need to understand a decision tree before learning what a random forest model is. Start with the `sklearn` **tutorial** on decision trees. Then check out these videos on youtube for a visual explanation:

- **Decision Trees Part 1**
- **Decision Trees Part 2**
- **Random Forest Part 1**
- **Random Forest Part 2**

See this **list of classification accuracy metrics** that can be computed in `sklearn`.

Follow **this series of blog posts** for an understanding of how these accuracy metrics work on multiclass problems like ours.

Glossary

- **Cross-validation**: A technique for estimating model performance where multiple models are trained and tested each on a separate partition of the entire dataset.
- **Classification accuracy**: The percent of correct classifications made by a model.

Hyperparameter Tuning and Regularization

We ended the last concept with a classification accuracy of 77%. However, there are a few more ways we can turn to improve the performance.

We at first used our best guesses but now we can explore the space and see if we can improve the performance. We found that reducing the maximum tree depth to 2, we have significantly increased our classification accuracy, from 77% to 89%. By reducing the depth to 2, we are **regularizing** our model. Regularization is an important topic in ML and is our best way to avoid overfitting. This is why we see an increase in the cross-validated performance.

But, we used the entire dataset many times to figure out the optimal hyperparameters. In some sense, this is also overfitting. Our 90% classification accuracy is likely too high, and not the generalized performance. In the next concept, we can see what our actual generalized performance might be if we use our dataset to optimize hyperparameters.

Cross-Validation and Feature Importance

Using the Nested Cross Validation technique, we'd ideally pick the best hyperparameters on a subset of the data, and then evaluate it on a hold-out set which is similar to train-validation-test set split but we don't have enough data to do so. When you don't have enough data to separate your dataset into 3 parts, we can nest the hyperparameter selection in another layer of cross-validation.

We then walked through how to actually apply this technique on our dataset. Our performance dropped because we are now not overfitting our hyperparameters when we evaluate model performance.

We have just learned that another way to regularize our model and increase performance (besides reducing the tree depth) is to reduce the number of features we use. The `RandomForestClassifier` can tell us how important the features are in classifying the data. We found the 10 most important features determined by the `RandomForestClassifier` and trained the model on just those 10 features. The trained model no longer misclassified `bike` as `walk` and this improved our classifier performance by 15%, just by picking the most important features!

Hyperparameter Tuning in Review

Machine learning models use data to fit their internal parameters. However, all models also have parameters that configure how they work and aren't modified during training, these parameters are called **hyperparameters**. We can train these hyperparameters by creating many different models, each with different hyperparameters and evaluating each model's performance. Just like we can overfit model parameters, we can also overfit its hyperparameters. To avoid this, we can estimate performance using **nested cross-validation**.

Some hyperparameters affect how easily the model will overfit the data, sometimes at the expense of complexity. In our case, this hyperparameter was the depth of the trees in the forest. When we limited the tree depth to just 2, we saw the cross-validation error decrease substantially.

Another hyperparameter that we can modify is the number of features that we choose to model. Random forest models use some features more than others to classify the data. In `sklearn` we can ask the `RandomForestClassifier` which features were more important than others. By building a new model that only uses the 10 best features, we were able to improve our performance to 93%.

Further Resources

Nested cross-validation can be a tricky concept to wrap your head around. Here are three different explanations from three different authors. Maybe one of the following resources will explain it in a way that clicks for you:

- **Nested CV - Weina Jin**
- **Nested CV - Elder Research**
- **Nested CV - Stack Exchange: Cross Validated**

Our code implementing nested CV was pretty verbose so that you could see all the steps. As with almost everything in ML, `sklearn` can do it for us as well and you can learn more about Nested CV in `sklearn` through the **documentation**.

Is overfitting our hyperparameters really a problem in practice? **Yes (or so says this 2010 paper)**

An explanation of the difference between hyperparameters and regular parameters with this **article** from Machine Learning Mastery.

If you want to learn more about Regularization through this **article** from Towards Data Science.

Glossary

- **Hyperparameter**: A parameter of the model that dictates how the model learns. This is not trained during the training process of the model itself.
- **Regularization**: Regularization is a technique to reduce overfitting of a model by discouraging complexity in the model.
- **Nested cross-validation**: A technique to determine model performance when hyperparameters are also optimized.

Activity Classification Recap

Summary

We've just done our first modeling task with wearable data. It took a while to get here. We had to learn what an accelerometer was and how it collected information about movement, and we had to learn about signal processing to build the features that we used in our model. We were able to build a pretty successful three-class classifier using a random forest and by doing proper hyperparameter optimization. But this problem is about as easy as it gets.

If you're looking for a challenge, go to the original dataset and see that it actually has four classes -- running, walking, high-intensity biking, and low-intensity biking. You could try building a classifier that can distinguish between high-intensity and low-intensity biking. You'll also find a PPG signal in the dataset. That might help you discriminate between these two classes. You might also find that it's impossible. Maybe there's not enough information in these sensors to solve this problem, or maybe the dataset wasn't collected as rigorously as we need. Ultimately, low-resistance and high-resistance is subjective, and the dataset description even notes:

> ...each participant was free to set the pace of the treadmill and pedal rate on the bike so they were comfortable and also to change these settings or stop the exercise at any time.

More often than not, this is the reality of real-world data science problems.

Outline

Understanding Your Data

- Wrist PPG Dataset
- Data Exploration and Visualization Understanding The Literature
    - Feature Engineering and Extraction Modeling Performance Evaluation Hyperparameter Optimization

Further Resources

Data Exploration

- **Wrist PPG Dataset**
- This is a great blog post by **Casie Kozrykov** who taught me statistics at Google! In it, she describes the dangers of overfitting your brain when you explore your data. **Your dataset is a giant inkblot test**.
- Check out **this StackOverflow discussion** on the value of data exploration. From one of the responses:

> Two weeks spent training a neuralnet can save you 2 hours looking at the input data

- And finally, a **blog post** from a machine learning practitioner on the data exploration.

Feature Creation

This blog post, **Machine Learning with Signal Processing Techniques**, goes through a very similar process as this lesson. It starts by explaining some signal processing techniques (like we did earlier in the course). The author uses those techniques to build features in much the same way we just did. And then, he uses those features to build an activity classification model, just as we are about to!

The algorithm we built was inspired by these two papers.

Mehrang S., Pietilä J., Korhonen I. An Activity Recognition Framework Deploying the Random Forest Classifier and A Single Optical Heart Rate

Monitoring and Triaxial Accelerometer Wrist-Band. Sensors. 2018;18:613. doi: 10.3390/s18020613.

Liu S, Gao RX, Freedson PS. Computational methods for estimating energy expenditure in human physical activities. Med Sci Sports Exerc. 2012;44:2138–2146. doi: 10.1249/MSS.0b013e31825e825a.

Model Building

Random forests are boosted decision tree models. You need to understand a decision tree before learning what a random forest model is. Start with the `sklearn` **tutorial** on decision trees. Then check out these videos on youtube for a visual explanation:

Decision Trees Part 1 **Decision Trees Part 2** **Random Forest Part 1** **Random Forest Part 2**

See this **list of classification accuracy metrics** that can be computed in `sklearn`.

Follow **this series of blog posts** for an understanding of how these accuracy metrics work on multiclass problems like ours.

Hyperparameter Optimization

Nested cross-validation can be a tricky concept to wrap your head around. Here are three different explanations from three different authors. Maybe one of the following resources will explain it in a way that clicks for you:

- **Nested CV - Weina Jin**
- **Nested CV - Elder Research**
- **Nested CV - Stack Exchange: Cross Validated**

Our code implementing nested CV was pretty verbose so that you could see all the steps. As with almost everything in ML, `sklearn` can do it for us as well and you can learn more about Nested CV in `sklearn` through the **documentation**.

Is overfitting our hyperparameters really a problem in practice? **Yes (or so says this 2010 paper)**

An explanation on the difference between hyperparameters and regular parameters with this **article** from Machine Learning Mastery.

If you want to learn more about Regularization through this **article** from Towards Data Science.

Glossary

- **Hyperparameter**: A parameter of the model that dictates how the model learns. This is not trained during the training process of the model itself.
- **Regularization**: Regularization is a technique to reduce overfitting of a model by discouraging complexity in the model.
- **Nested cross-validation**: A technique to determine model performance when hyperparameters are also optimized.
- **Cross-validation**: A technique for estimating model performance where multiple models are trained and tested each on a separate partition of the entire dataset.
- **Classification accuracy**: The percent of correct classifications made by a model.

ECG Signal Processing

Introduction to ECG Processing

Summary

In this lesson, you will build an algorithm that processes the ECG signal to find the heartbeats. You will then build an algorithm that takes these heartbeat locations and detects atrial fibrillation. We will learn about the physiology of the heart during normal function as well as during atrial fibrillation. This will help us build better features for our algorithms.

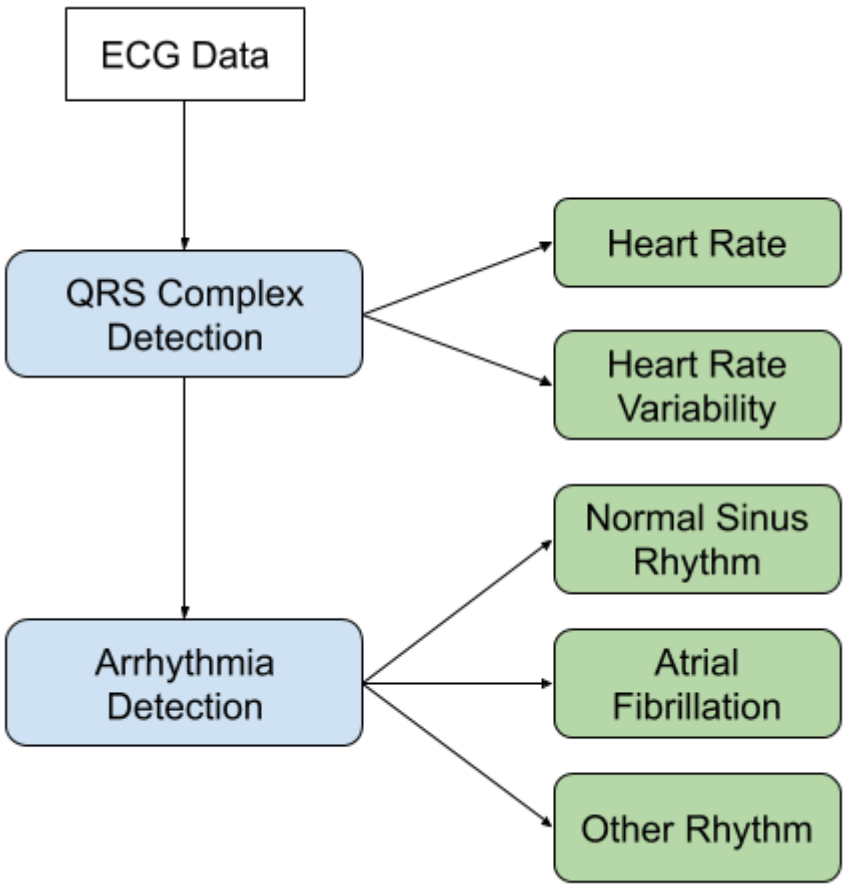
Lesson Outline

- Heart Physiology
- QRS Complex Detection
  - Pan-Tompkins Algorithm
  - Extending Pan-Tompkins

- Atrial Fibrillation Physiology
- Arrhythmia Detection
  - Computing in Cardiology Challenge 2017
  - Data Exploration
  - Feature Extraction
  - Modelling

Lesson Concepts

Below is a diagram describing the concepts that we will cover in this lesson. Specifically, we will build a system where two algorithms are cascaded together. Each algorithm is capable of producing clinically meaningful results, but the QRS complex detection algorithm's output is fed into the arrhythmia detector.



Lesson Concepts

Heart Physiology

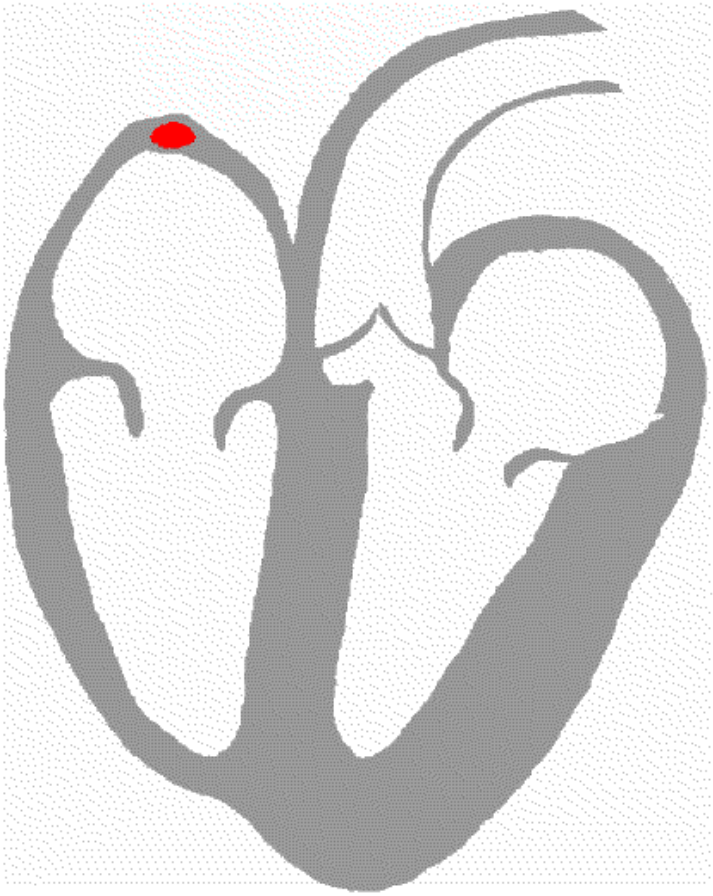
Summary

Before we can understand all the parts of the ECG wave, we need to first learn more about how the heart works. The heart is made up of four chambers, two atria and two ventricles. The **atria** pump blood into the ventricles and then the **ventricles** pump blood throughout the body. Each heart cell is polarized, meaning there is a different electrical charge inside and outside of the cell. At rest, the inside of the cell is negatively charged compared to the outside. When the cell **depolarizes**, positive charges outside of the cell flow inside and makes the interior of the cell positively charged relative to the outside. This depolarization causes the cell to contract. The movement of charges across a heart cell's membrane is the source of the electrical activity that gets measured by an ECG.

The conduction of electrical impulses through the heart follows a regular pattern orchestrated by the **cardiac conduction system**. Each heartbeat starts with an impulse in the sinus (SA) node. The **sinus node** is the natural pacemaker of the heart and is responsible for setting the heart's natural rhythm. The impulse then propagates throughout the atria and causes the atria to contract. Then, the impulse enters the **atrioventricular (AV) node**, which delays the propagation of the signal to the ventricles. This gives the atria time to pump blood into the ventricles. After a brief pause, the signal passes through the AV node and into the ventricles, causing the ventricles to contract. This is the largest electrical disturbance in the cardiac cycle. After the ventricles contract, they **repolarize**, meaning the ions move back across the cell wall to their initial resting state.

Each part of this sequence is responsible for creating a distinct marker-- or wave --in the ECG signal. The P-wave corresponds to the depolarization of the atria. The pause between the P-wave and the Q-wave is caused by the AV node delaying the electrical impulse to the ventricles. When the ventricles contract, we see the large QRS complex. And finally, after some time, the ventricles repolarize, causing the T-wave. The atria repolarize around the same time as the ventricles finish contracting so that activity is not visible in an ECG because it is obscured by the S-wave.

This cycle repeats every heartbeat and results in a very regular and uniform looking signal.



Cardiac cycle animation Source: By Kalumet - selbst erstellt = Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=438140

Glossary

- **Atria**: The upper chambers of the heart that pass blood to the ventricles.
- **Ventricles:** The main chambers of the heart that pump blood throughout the body

- **Depolarization:** The movement of charges across a cell membrane that causes the inside of the cell to become less negatively charged.
- **Repolarization:** The movement of charges across the cell membrane that restore the negative resting charge inside the cell.
- **Cardiac conduction system:** A group of specialized cardiac cells that send signals to the heart, causing it to contract. The main components that we discussed in this course were the sinoatrial (SA) node and the atrioventricular (AV) node. Other components include the bundle of His, left and right bundle branches, and the Purkinje fibers, which propagate the signal from the AV node throughout the ventricles.
- **Sinus node:** The natural pacemaker of the heart. Responsible for generating the impulse that causes the heart to beat.
- **AV node:** Part of the cardiac conduction system that propagates the impulse from the atria to the ventricles after a delay.

Pan-Tompkins QRS Complex Detection

Previously, we learned that ventricular contraction corresponds with the QRS complex. Finding these QRS complexes in the ECG signal is important for a number of algorithmic goals. It directly gives you information about the heart rate and heart rhythm and provides important context when doing more complex processing.

The way we find these QRS complexes is by using an algorithm called the Pan-Tompkins algorithm.

For most event detection tasks like this, we start with preprocessing steps that will boost the signal and suppress the noise. For Pan-Tompkins, we start with the bandpass filter that selects for frequencies in the QRS complex and suppresses frequencies outside of that band. This is followed by a one-sample difference, which is analogous to a derivative operation. This will preserve the steep slopes of the QRS complex and attenuate shallower ones elsewhere in the waveform. Next, we do an element-wise square which non-linearly amplifies the larger portions of our signal and makes everything positive. Finally, we do a moving sum over a fixed window length. This takes advantage of the fact the QRS complex has a fixed width of 150ms. If we tune the moving sum window to 150ms, we can fully take advantage of all the energy in the

QRS complex, while attenuating other spikes that are shorter or longer than 150ms.

The detection steps are fairly simple. We simply find the peaks in our pre-processed waveform and then threshold them to select for the QRS complexes.

Next, we take a look at the code for this algorithm, and we plot the output.

Pan-Tompkings In Code

Recap

We saw a basic implementation of the Pan-Tompkins algorithm and visualized how the pre-processing steps change the waveform. On noisy signals, with small QRS complexes, we saw just how well the pre-processing steps could improve our signal. Still, we saw that sometimes it would not be enough and more complex detection rules need to be implemented.

In the following exercise, we implement more sophisticated detection rules for just this purpose.

The overall goal here isn't necessarily to understand the Pan-Tompkins algorithm per se; it's really to get to a place where we can solve this kind of problem in any domain where you have some knowledge of the underlying physiology and the signal characteristics and the noise characteristics and you can do for your task what Pan-Tompkins algorithm does for QRS complex detection. As you learn more about these tools and operations that can boost signal and suppress noise in your specific domain, you'll be able to build a better intuition for designing your own algorithm on your own problem.

Summary

After improving the detection rules, we substantially improve the precision and recall of our detector substantially. Watch the video to see how we implement the three strategies for more robust detection:

- Refractory Period Blanking
- Adaptive Thresholding
- T-Wave Rejection

The Pan-Tompkins algorithm is a great example of an event detection algorithm. This kind of algorithm is especially common in biomedical time series processing. By diving deep into this algorithm, you've seen how knowledge of the underlying physiology, the signal characteristics, and the noise characteristics can help in the algorithm design. As you learn more about the tools and operations that can boost the signal and suppress noise in a specific domain, you will have better intuition for designing algorithms to solve these types of problems.

Atrial Fibrillation

We've discussed atrial fibrillation previously in the context of the Apple Heart Study and the Framingham Study. Now we'll learn what atrial fibrillation actually is. Atrial Fibrillation is a type of **arrhythmia**, which is an irregular heart rhythm.

Recall, in a normal heart rhythm, the sinus node generates the impulse that causes the atria to contract. This impulse is propagated to the AV node and then throughout ventricles, causing the ventricles to contract. This process results in a very regular rhythm called **sinus rhythm**.

In **Atrial Fibrillation**, instead of the SA node being the sole location that begins the depolarization of the atria, there are multiple locations around the atria that will spontaneously and haphazardly generate an impulse. Each of these impulses causes a partial contraction of the atria, but no single impulse depolarizes the entire atria, so there is no coherent contraction of the atria. Occasionally, one of these impulses will reach the AV node, which will then cause a ventricular contraction. But this occurs at random times, so ventricular contractions occur irregularly.

When can we examine features of the ECG signal to detect atrial fibrillation? First, there is no P-wave because there is no coherent depolarization of the atria to cause an electrical disturbance large enough to create the P-wave. Instead, we see a fibrillating wave in the T-Q segment. Second, the QRS complexes are very irregular.

We mentioned earlier that atrial fibrillation is associated with an increased risk of stroke. Because the atria are not contracting completely, stagnant pools of blood will form in the atria. These pools can form blood clots that are then circulated through the bloodstream. As these clots pass through progressively

smaller and smaller arteries, they may eventually obstruct blood flow to the brain and cause a stroke.

Now we have seen how atrial fibrillation occurs physiologically and why it's a potentially dangerous condition. Next, we will build an algorithm that automatically detects atrial fibrillation and other arrhythmias from the ECG signal.

Glossary

- **Arrhythmia:** An irregular heart rhythm.
- **Sinus Rhythm:** The normal, regular heart rhythm, paced by the sinus node.
- **Atrial fibrillation:** An irregular rhythm caused by multiple, haphazard depolarizations across the atria.

Arrhythmia Detection: Dataset

In the next few concepts, we'll be building an arrhythmia classifier. The data we will use comes from the **Computing in Cardiology (CinC) Challenge 2017 dataset** hosted on Physionet.

The dataset contains thousands of short ECG snippets (30s - 60s) from the AliveCor mobile ECG monitor. The original challenge was to build a 4-class classifier for sinus rhythm, atrial fibrillation, alternative rhythm, and noisy record. We will throw out the noisy records and build a two-class classifier distinguishing between sinus rhythm and another rhythm (atrial fibrillation included).

From exploring our data, we see that we have 1.5x more sinus rhythm records than other rhythm records. Most of the records are 30 seconds long; some are 60 seconds long, and a few are somewhere in between.

We plot the data and visualize the QRS complex detections provided in the dataset. The QRS complex detector still detects QRS complexes during periods of high noise, but these detections are suspect.

Arrhythmia Detection: Features

Summary

As with activity classification, we will featurize our raw signal and throw those features into a classifier to detect an abnormal heart rhythm. Our input signal will be the time between two QRS complexes, known as the RR interval (for the R-wave). This is also called the **inter-beat-interval**. Our input signal will be derived from the output of the Pan-Tompkins algorithm. We will include some of the features in

> Behar, Rosenberg, Yaniv, Oster. Rhythm and Quality Classification from Short ECGs Recorded Using a Mobile Device. Computing in Cardiology Challenge 2017.

and

> Bonizzi, Driessens, Karel. Detection of Atrial Fibrillation Episodes from Short Single Lead Recordings by Means of Ensemble Learning. Computing in Cardiology Challenge 2017.

which were entries in the CinC challenge. Recall that the RR interval time series is irregularly sampled heartbeat because we get a new measurement at each heartbeat, not at some fixed interval in time. As you'll soon see, many of the features we want can be computed on an irregular time series, but some, especially frequency domain features, cannot. In that case, we'll first have to make this uniform using interpolation before we can compute our frequency domain features.

So we can separate our features into time domain features and frequency domain features. The first few are summary statistics like min, max, median, mean, standard deviation. Then we want to compute the number of outliers. And we'll say an outlier is an RR interval that is greater than 1.2 times the average RR interval in that record.

Next, we want to compute the root-mean-square of the difference between adjacent RR intervals. And lastly, we want the percent of RR interval differences that are greater than 50 milliseconds.

Before we compute our frequency domain features, we will need to regularize the RR interval time series so that we can take the FFT of it. We want to interpolate the RR interval time series onto a regular 4 Hz time grid. Then the

features we want will be the largest FFT magnitude component between 0.04 Hz and 0.15 Hz. We also want to know at which frequency this occurs. And we want both of these values for the band between 0.15 Hz and 0.4 Hz as well.

I will leave the feature computation code for you to do in the next exercise and in the next video we will pick this back up with the implementation.

Glossary

- **inter-beat-interval:** The time between successive heart beats. Also called the RR interval.

Exercise 2: Arrhythmia Features

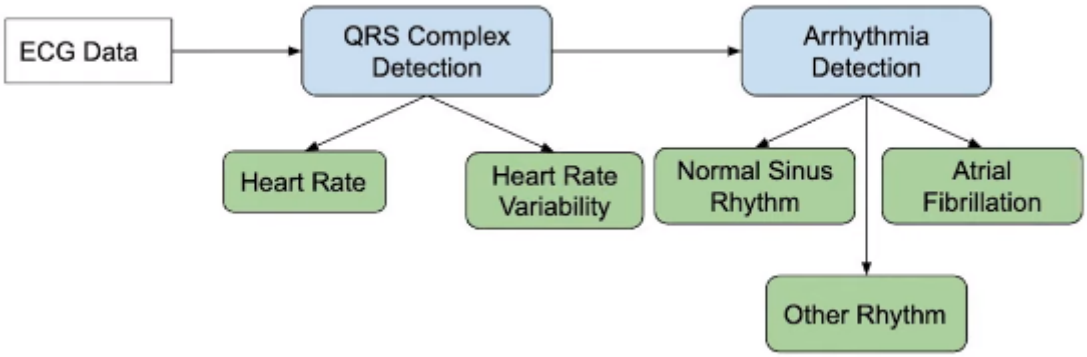
In the video, we go over how the features are computed from the input signal. NumPy operations help us compute the summary statistics easily. We can use NumPy boolean arrays to count the number of outliers and the percent of RR interval differences greater than 50ms. For the frequency domain features, we first resample our RR interval time series to a regular grid using linear interpolation.

Exercise 3: Atrial Fibrillation

Summary

I used a `RandomForestClassifier` with 100 trees and a max depth of 4. With these basic features, it achieved a precision of 73% and a recall of 81%. Further improvements can be made by using a more advanced technique for RR interval time series regularization, and by using more informative features. See the further resources sections for example of more advanced atrial fibrillation detection algorithms.

Recap: ECG Signal Processing



Summary

In this lesson, we saw a good example of how algorithms can be cascaded together. Lower level algorithms operate on the raw data and produce a set of more meaningful and versatile metrics. These metrics can then be inputs into various higher-level algorithms that are more informative about the underlying physiology of the wearer, their health, or their disease state. This allows us to independently iterate on different parts of this stack. Making the underlying algorithms more accurate will generally improve the higher-level algorithms, but this is not always the case. It's important to remember that any change made in the lower level algorithms will have downstream effects.

We also saw how having a solid understanding of the physiological processes that generate our signal can have a profound impact on how we design our algorithm.

Lesson Outline

- Heart Physiology
- QRS Complex Detection
  - Pan-Tompkins Algorithm
  - Extending Pan-Tompkins
- Atrial Fibrillation Physiology
- Arrhythmia Detection
  - Computing in Cardiology Challenge 2017
  - Data Exploration

- Feature Extraction
- Modelling

Further Resources

Dale Dubin's Rapid Interpretation of EKG's is one of the best resources for quickly understanding the ECG signal.

The original Pan-Tompkins algorithm paper.**Pan J, Tompkins WJ. A real-time QRS detection algorithm. IEEE Trans Biomed Eng. 1985;32(3):230–236. doi:10.1109/TBME.1985.325532**

These two papers were the inspiration for the classifier that we built. They used a lot more features and included information from the raw ECG signal as well. See how they built 4-class classifiers and performance was evaluated in this challenge.

- **Behar, Rosenberg, Yaniv, Oster. Rhythm and Quality Classification from Short ECGs Recorded Using a Mobile Device. Computing in Cardiology Challenge 2017.**
- **Bonizzi, Driessens, Karel. Detection of Atrial Fibrillation Episodes from Short Single Lead Recordings by Means of Ensemble Learning. Computing in Cardiology Challenge 2017.**

This paper describes some important features that have been shown to be good for atrial fibrillation detection based on the irregularity of RR intervals.**Sarkar S, Ritscher D, Mehra R. A detector for a chronic implantable atrial tachyarrhythmia monitor. Biomedical Engineering IEEE Transactions on 2008;55(3):1219–1224.**

One of the most famous databases for ECG arrhythmia detection is the **MIT-BIH database**. **This article** is a great overview of the database and its impact.

Glossary

- **Atria**: The upper chambers of the heart that pass blood to the ventricles.
- **Ventricles:** The main chambers of the heart that pump blood throughout the body
- **Depolarization:** The movement of charges across a cell membrane that causes the inside of the cell to become less negatively charged.

- **Repolarization:** The movement of charges across the cell membrane that restore the negative resting charge inside the cell.
- **Cardiac conduction system:** A group of specialized cardiac cells that send signals to the heart, causing it to contract. The main components that we discussed in this course were the sinoatrial (SA) node and the atrioventricular (AV) node. Other components include the bundle of His, left and right bundle branches, and the Purkinje fibers, which propagate the signal from the AV node throughout the ventricles.
- **Sinus node:** The natural pacemaker of the heart. Responsible for generating the impulse that causes the heart to beat.
- **AV node:** Part of the cardiac conduction system that propagates the impulse from the atria to the ventricles after a delay.
- **Refractory Period:** The period after depolarization where a cell cannot depolarize again.
- **Sinus Rhythm:** The normal, regular heart rhythm, paced by the sinus node.
- **Arrhythmia:** An irregular heart rhythm.
- **Atrial fibrillation:** An irregular rhythm caused by multiple, haphazard depolarizations across the atria.
- **inter-beat-interval:** The time between successive heartbeats. Also called the RR interval.
