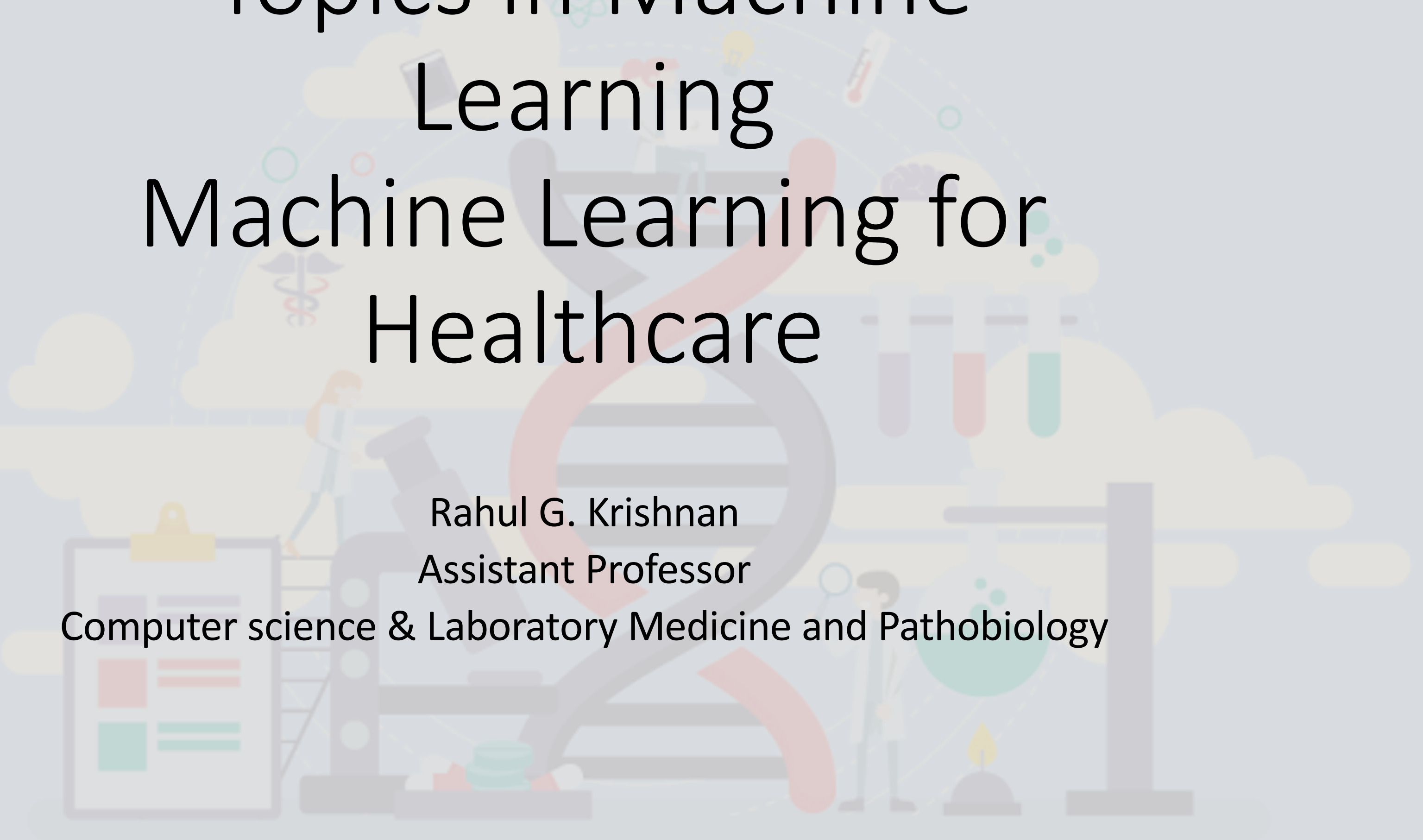
Topics in Machine Learning
Machine Learning for Healthcare

Rahul G. Krishnan
Assistant Professor
Computer science & Laboratory Medicine and Pathobiology

UNIVERSITY OF TORONTO

VECTOR INSTITUTE



Outline

- Announcements
- Time series modeling in healthcare
  - Time-series data in the ICU
  - Time-series data in for chronic disease care and management
  - What can machine learning do?
- Statistical models for time-series data
  - Univariate models for capturing patterns
  - Multi-variate models for high-dimensional data
  - Latent variable models for time-series data
  - Clustering, forecasting and tackling clinical tasks

Announcements

- In two weeks, project proposals [10% of grade] are due:
  - Start forming teams and brainstorming project ideas
  - Create an outline for your project proposal
  - Link: csc2541hf-2021.github.io/assignments/projectproposal
- Sign up for student presentations [15% of grade]
  - Present in pairs, first to TAs, then to the class
  - First come first serve for papers
  - csc2541hf-2021.github.io/assignments/paperpresentation
  - See quercus announcement by Farnam
- Sign up for project presentation slots

Time-series datasets

PhysioNet/MIMIC

Intensive care data

Multiple Myeloma Research Foundation

Chronic disease registries

Parkinson's Disease (PPMI)

Similarities

Multi-modality
Missingness
Sampling biases

Differences

Multi-modality
Missingness
Sampling biases

Tasks for machine learning

- Risk stratification with time-series data
  - All the same techniques we saw previously except our conditioning set x now comprises a time-series
- Pattern discovery in time-series data
  - K-means is easy to apply on static data
  - What about noisy, missing, time-varying data?
- Forecasting
  - Can we use statistical models to predict how a patient might evolve over time
  - Counterfactual reasoning is an important topic
    - Condition on aspects of the data that can change how observations behave over time

Challenges for machine learning

- Clinical decision making is multi-modal
- Frequency of observations and interventions can vary dramatically:
  - Intensive care unit: Observations and interventions happening in real-time
    - High-frequency data
  - Chronic disease management: Observations and interventions happen over the span of months or years
    - Low-frequency data
- Missingness is rampant
  - ICU: sensor noise
  - Chronic disease management: administrative errors, access to health insurance

Preprocessing for time-series data

- For static data:
  - Z-scoring
  - Min-max normalization
- For temporal data:
  - Normalization by standard reference measures (healthy values)
  - Log-transformation
  - Removing the mean of a time-series
  - Normalization to [-1 , 1]
  - Outlier removal
    - Not a good idea to remove if signal is in tails of the distribution
- Imputation for missing data:
  - Feed-forward imputation
  - Linear interpolation
  - Polynomial interpolation
  - We'll see more advanced imputation strategies later in the class

Learning problems with time-series data

- One of the best ways to learn about statistical models for time-series data is to know what you can do with them,
  - Unsupervised learning
    - Forecasting – predict time-series into the future
    - Identify and detect patterns and clusters in time-series
  - Supervised learning:
    - Make predictions from time-series
- Lets turn our attention to focus on forecasting
  - To do forecasting, we often need a *model of the time-series*
  - *We'll start with the task of modeling a single biomarker*

Univariate models of time series data

Time-series regression with time-series features

$$y_t = c + \theta_1 x_{t-1} + \theta_2 x_{t-2} + \ldots + \theta_3 x_{t-k}$$

- Treat time-series modeling as a *linear* regression problem
- x: features (potentially time-varying)
- y: outcome of interest
- But what if we had no other features?

ARIMA [**A**uto**R**Egressive **I**ntegrated **M**oving **A**verage]

$$y'_t = c + \phi_1 y'_{t-1} + \ldots + \phi_p y'_{t-p} + \theta_1 \epsilon_{t-1} + \ldots + \theta_q \epsilon_{t-q} + \epsilon_t$$

- ARIMA(p,d,q) model
  - p: order of autoregressive part
  - d: degree of differencing
  - q: order of moving average

Discuss on board

Pro: Very flexible model of time-series data!
Con: linear additive model

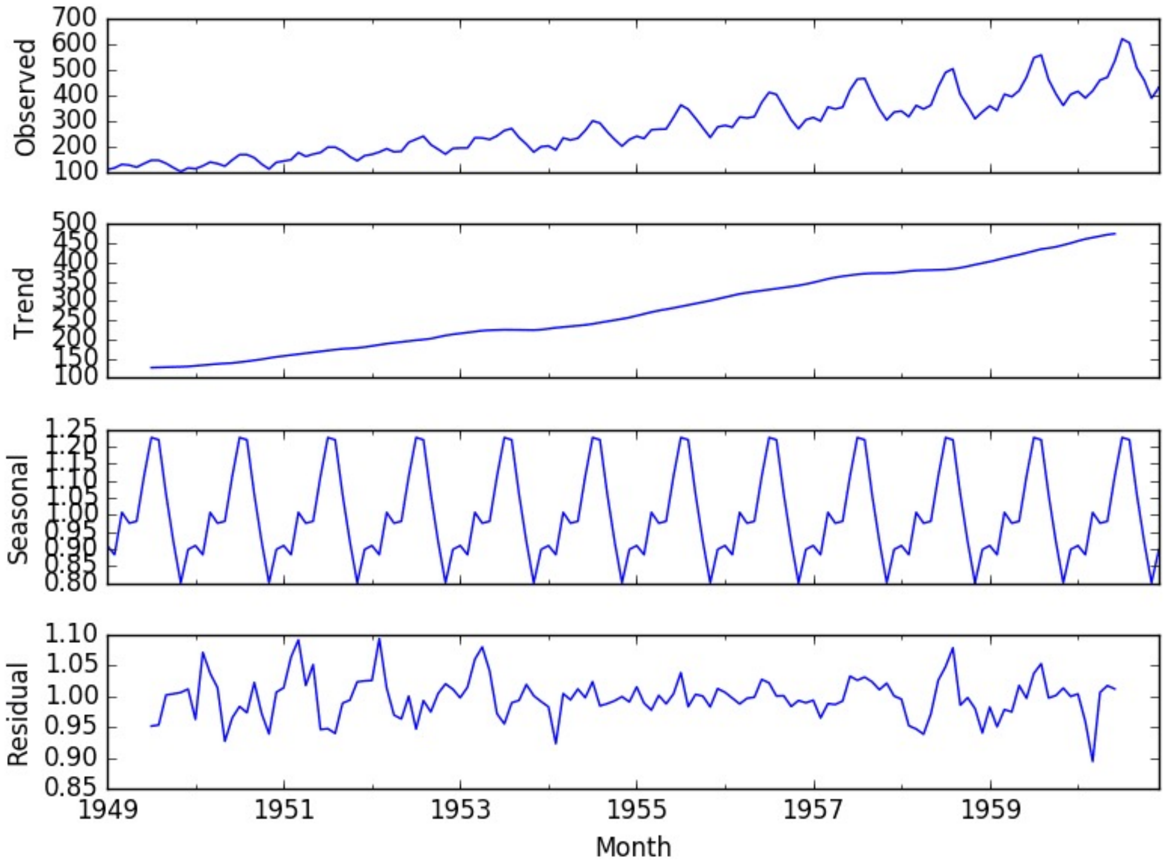
Reference: **Forecasting: Principles and Practice,** Rob J Hyndman and George Athanasopoulos

Nonlinear models of univariate time-series data

$$y_t = f(x_{t-1}, \ldots, x_{t-k}; \theta)$$

- Very general formulation for a broad class of time series problems with nonlinear models
- Theta represent the parameters of this model
- Next, we’ll study a single case study of the use of a such a non-linear model to make predictions from electrocardiogram data

General rule: Decompose time-series



When you think about modeling time-series data, think about trends and patterns that exist and how to design models to capture different variation.

Multivariate models of time series data

What happens if we have more than one time-series to model?

- Simple option: Use D-different univariate time-series models to model the data
  - Discussion – is this a good idea? What are alternatives?

Combining Possibly Related Estimation Problems

By B. Efron and C. Morris

Stanford University *The RAND Corporation*

[Read before the Royal Statistical Society at a meeting organized by the Research Section, on Wednesday, May 9th, 1973, Professor J. Durbin in the Chair]

Summary

We have two sets of parameters we wish to estimate, and wonder whether the James–Stein estimator should be applied separately to the two sets or once to the combined problem. We show that there is a class of compromise estimators, Bayesian in nature, which will usually be preferred to either alternative.

"*The difficulty here is to know what problems are to be combined together—why should not all our estimation problems be lumped together into one grand melée?*"

George Barnard commenting on the James–Stein estimator, 1962.

Keywords: empirical bayes; james–stein; simultaneous estimation; combining estimates; partial exchangeability

1. Introduction

Suppose that the statistician wishes to estimate parameters $\theta_1, \theta_2, .., \theta_k$ where each θ_i is the mean of an independent normal variate x_i,

$$x_i \,|\, \theta_i \sim \mathcal{N}(\theta_i, D). \tag{1.1}$$

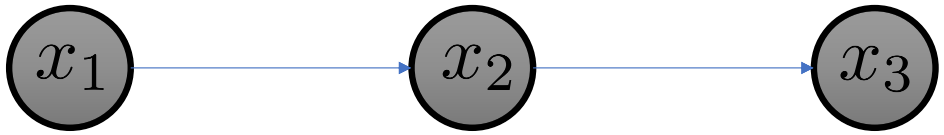
James and Stein (1961) have shown that for $k \geqslant 3$ the estimator

$$\delta_i = \left[1 - \frac{(k-2)\,D}{S}\right] x_i, \tag{1.2}$$

$S = \sum_{i=1}^{k} x_i^2$, is uniformly better than the maximum likelihood estimator $\delta_i^0 = x_i$ when the loss function is the sum of squared errors.

The James–Stein estimator seems to do the impossible. The estimate of each θ_i is made to depend not only on x_i but on the other x_j, whose distributions seemingly are unrelated to θ_i, and the result is an improvement over the maximum likelihood estimator no matter what the values of $\theta_1, \theta_2, ..., \theta_k$. The reaction of the statistical community to this *tour de force* has been generally hostile, the usual suggestion being that this is some sort of mathematical trick devoid of genuine statistical merit. Thus we have the "speed of light" rhetorical question, "Do you mean that if I want to estimate tea consumption in Taiwan I will do better to estimate simultaneously the speed of light and the weight of hogs in Montana?" (For a recent example see Kempthorne's discussion following Lindley and Smith, 1972.)

First-order Markov models



$$p(x_1, x_2, x_3) = p(x_1)p(x_2|x_1)p(x_3|x_2)$$

K-gram models

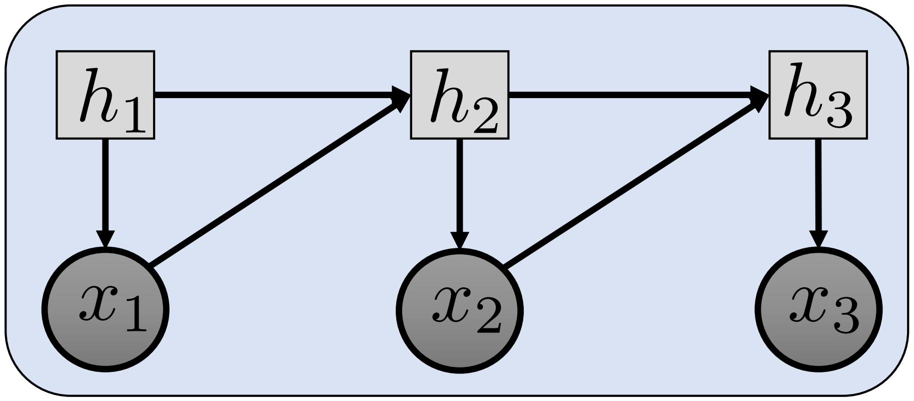
$$\begin{tikzcd}
x_1 \arrow[r] \arrow[rr, bend left] \arrow[rrr, bend right] & x_2 \arrow[r] \arrow[rr, bend left] & x_3 \arrow[r] & x_4
\end{tikzcd}$$

$$p(x_1, x_2, \ldots, x_4) = p(x_1)p(x_2|x_1)p(x_3|x_{1\ldots2})p(x_4|x_{1\ldots,3})$$

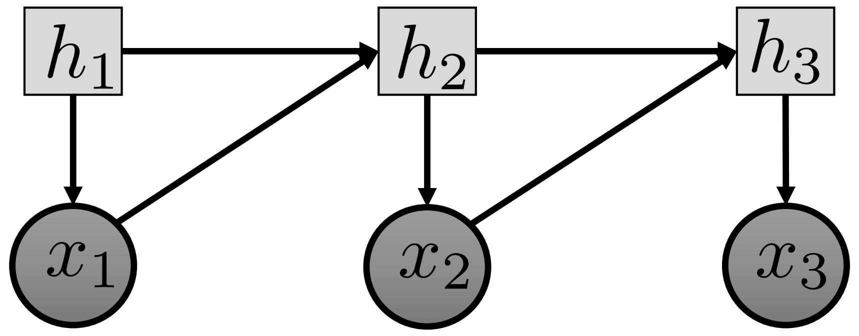
Recurrent Neural Networks

- Auto-regressive sequential models of data
- Forward recurrent neural network model
    - Each **hidden state** summarizes all the variables in the **past**

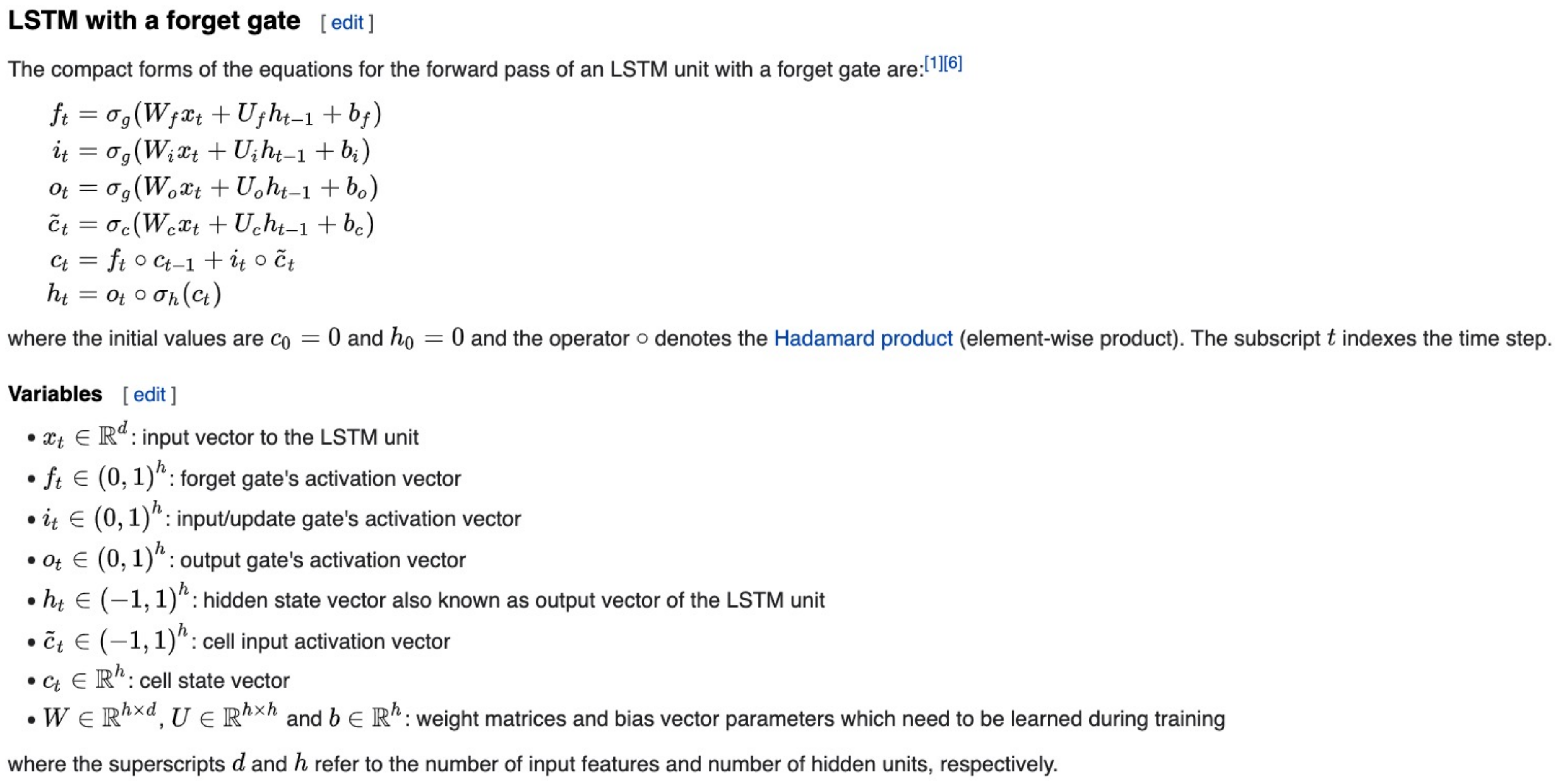
$$p(x_1, x_2, x_3) = p(x_1|h_1)\hat{p}(h_2|h_1)p(x_2|h_2)\hat{p}(h_3|h_2)p(x_3|h_3)$$



Recurrent neural networks in action



- Widely used for time-series modeling
- The parameterization of the functions that control how h behaves dictate the type of recurrent neural networks:
  - Long short-term memory (LSTM)
  - Gated recurrent units (GRU)



LSTM with a forget gate [edit]

The compact forms of the equations for the forward pass of an LSTM unit with a forget gate are:[1][6]

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

where the initial values are $c_0 = 0$ and $h_0 = 0$ and the operator $\circ$ denotes the Hadamard product (element-wise product). The subscript t indexes the time step.

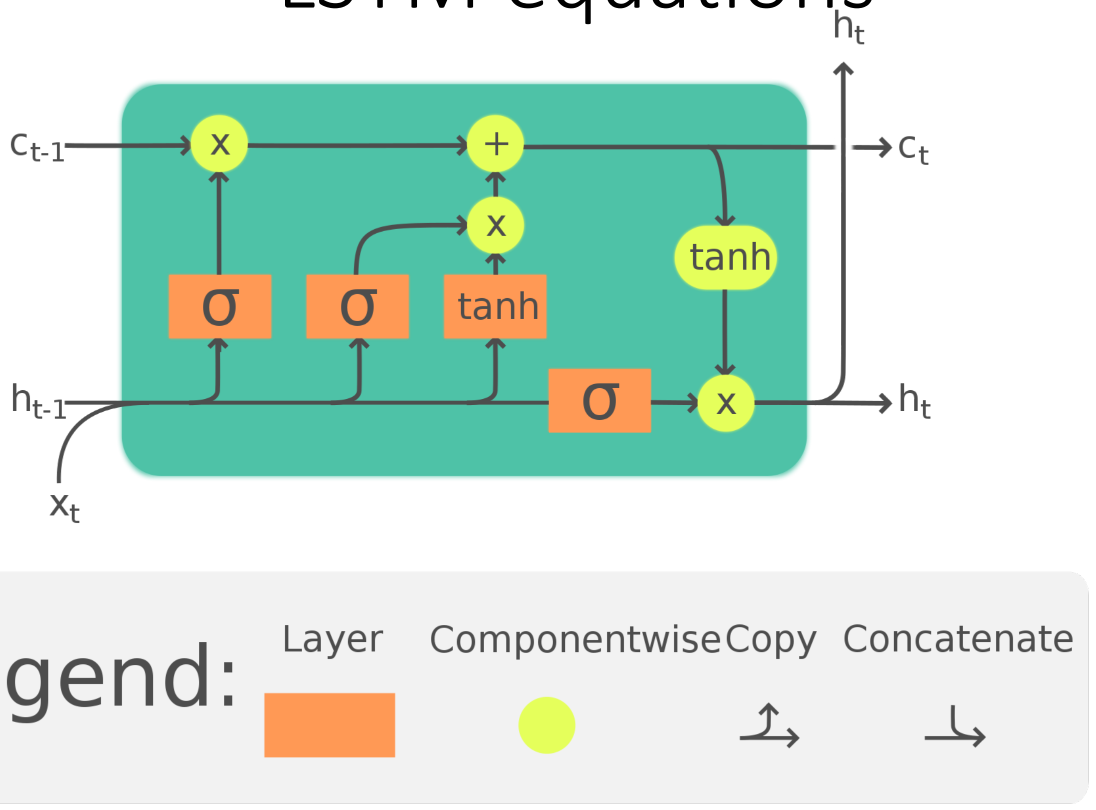
Variables [edit]

- $x_t \in \mathbb{R}^d$: input vector to the LSTM unit
- $f_t \in (0,1)^h$: forget gate's activation vector
- $i_t \in (0,1)^h$: input/update gate's activation vector
- $o_t \in (0,1)^h$: output gate's activation vector
- $h_t \in (-1,1)^h$: hidden state vector also known as output vector of the LSTM unit
- $\tilde{c}_t \in (-1,1)^h$: cell input activation vector
- $c_t \in \mathbb{R}^h$: cell state vector
- $W \in \mathbb{R}^{h \times d}$, $U \in \mathbb{R}^{h \times h}$ and $b \in \mathbb{R}^h$: weight matrices and bias vector parameters which need to be learned during training

where the superscripts d and h refer to the number of input features and number of hidden units, respectively.

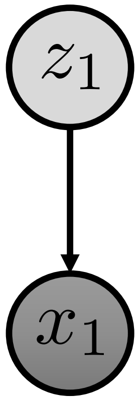
Source: https://en.wikipedia.org/wiki/Long_short-term_memory

LSTM equations



Source: https://en.wikipedia.org/wiki/Long_short-term_memory

Latent factor models



$$\begin{tikzcd} z_1 \arrow[d] \\ x_1 \end{tikzcd}$$

- Unsupervised models of (often high-dimensional data)
- Z: unobserved latent variation (often lower dimensional) than X (observed data)
- You may have encountered many variations of latent factor models:
  - Linear models:
    - Probabilistic PCA
    - Factor analysis
  - Non-linear models
    - Variational autoencoders

State space models

$$\begin{tikzcd}
z_1 \arrow[r] \arrow[d] & z_2 \arrow[r] \arrow[d] & z_3 \arrow[d] \\
x_1 & x_2 & x_3
\end{tikzcd}$$

There are many different varieties of state space models.

Each one makes different assumptions on how the probabilities behave and are transformed.

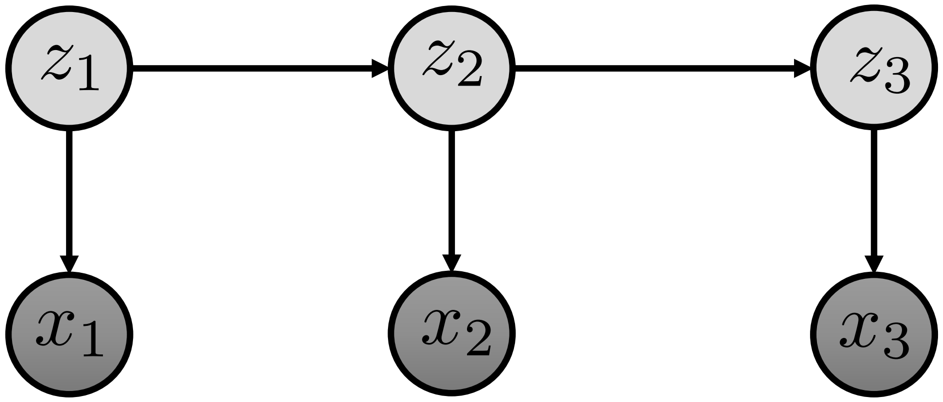
$$p(x_1, x_2, x_3) = \int_{z_1, z_2, z_3} p(x_1, x_2, x_3, z_1, z_2, z_3)$$

$$= \int_{z_1, z_2, z_3} p(z_1) p(z_2 | z_1) p(z_3 | z_2) \prod_{k=1}^{3} p(x_k | z_k)$$

Hidden Markov Model

Z are discrete random variables (often categorical)

Edges denote transition matrices


$$\begin{tikzcd} z_1 \arrow[r] \arrow[d] & z_2 \arrow[r] \arrow[d] & z_3 \arrow[d] \\ x_1 & x_2 & x_3 \end{tikzcd}$$

Linear Gaussian State Space Model

Z are continuous valued random variables (Gaussian)

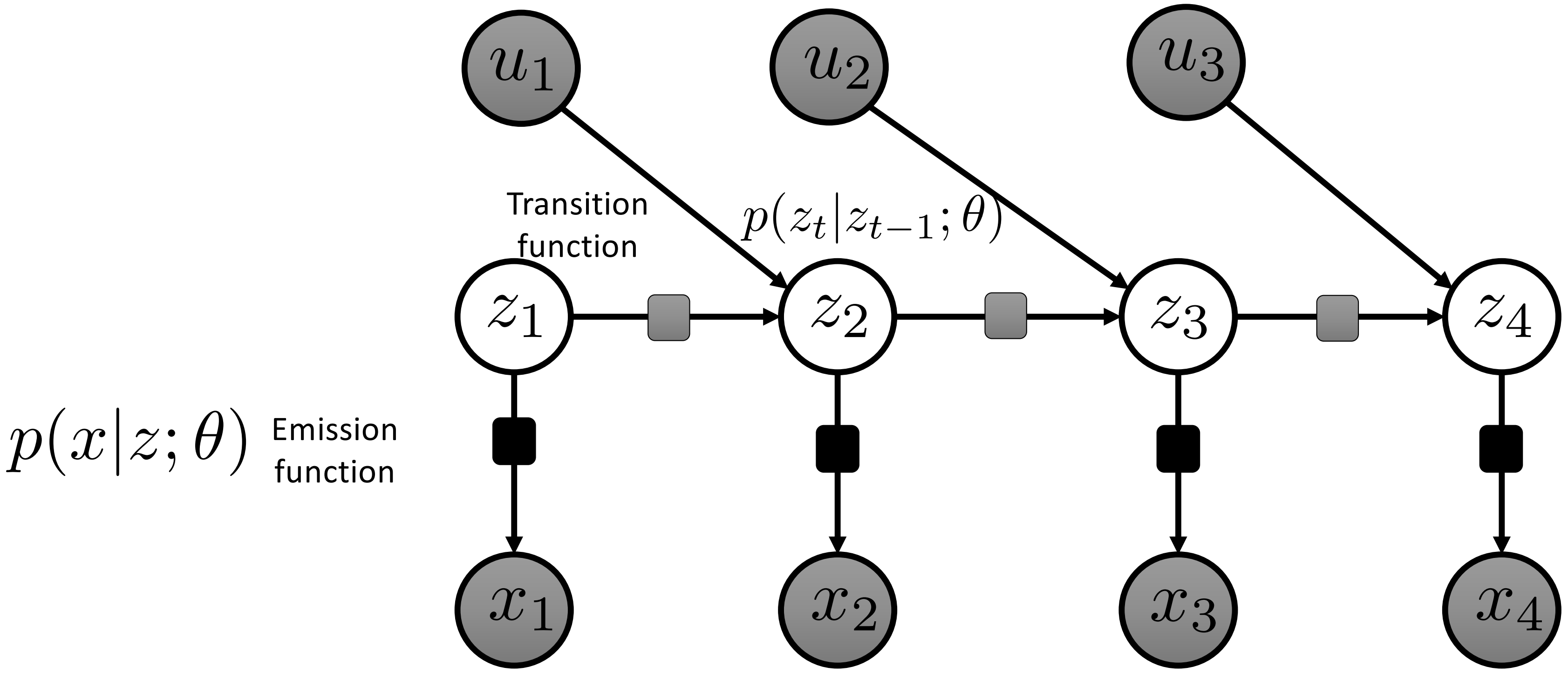
$$\begin{tikzcd}
z_1 \arrow[r] \arrow[d] & z_2 \arrow[r] \arrow[d] & z_3 \arrow[d] \\
x_1 & x_2 & x_3
\end{tikzcd}$$

$$z_t = \mathcal{N}(\mu_t, \sigma)$$

$$\mu_t = W z_{t-1} + b$$

$$\sigma = C$$

Deep Markov Models



Structured Inference Networks for Nonlinear State Space Models, **RGK,** US, DS, AAAI 2017

Learning time-series models

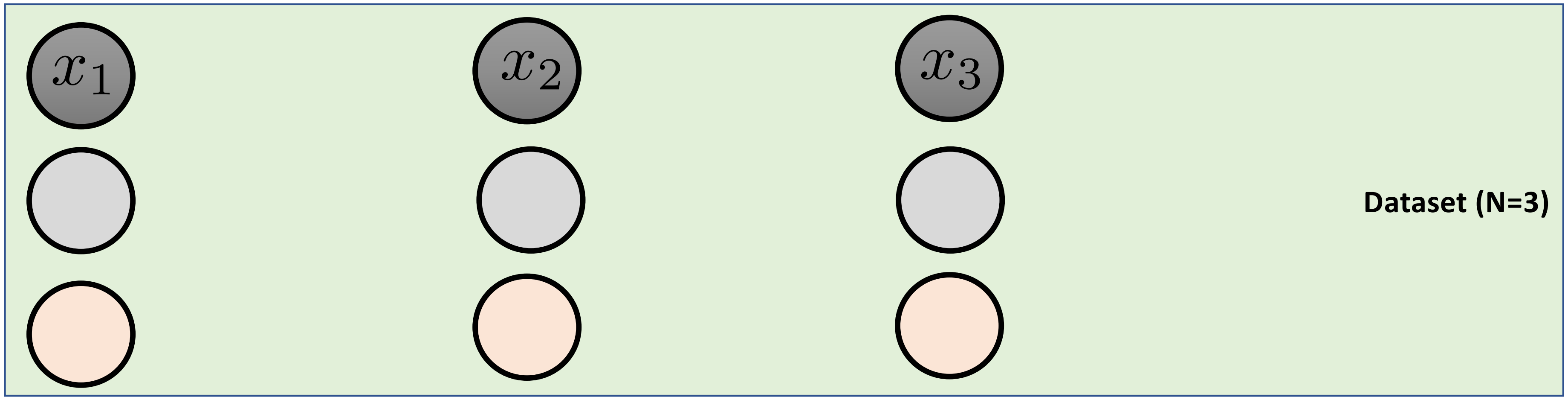
Univariate time-series

- Regression
- ARIMA
- Nonlinear regression via conv. nets

Multivariate time series

- K-order Markov models
- Recurrent neural networks
- State space models

Learning via maximum likelihood estimation



- Model parameters are learned via **maximum likelihood estimation**

$$\mathcal{L}(x_1, \ldots, x_T; \theta) = \log p(x_1, \ldots, x_T; \theta)$$

Score function (high is good, low is bad)

$$\theta = \arg\max_\theta \sum_{i=1}^{N} \mathcal{L}(x_1^i, \ldots, x_T^i; \theta)$$

Solve this optimization problem to **learn** the model. Often formulated as a minimization of the negative of the log-likelihood function

Recipes for learning via maximum likelihood estimation

- Usually:
  - Write down the log likelihood as a function of the model parameters
  - Use stochastic gradient ascent to maximize log likelihood of observed data to learn parameters
- For latent variable models:
  - If the posterior distribution is tractable, often can write the log-likelihood in closed form or obtain an unbiased estimate via Monte-Carlo sampling
  - Else: approximate inference
    - Variational inference
    - Markov Chain Monte Carlo

Evaluation of time-series models

- Mean-squared error
  - Forecasting on training data
  - Forecasting on held-out data
- Held-out log likelihood
- Introspection of model parameters

Supervision with time-series models

- If we care about doing classification with time-series we can adapt approaches we have seen to modify them for time-series data
- Case study:

$$y = f(x_1, \ldots, x_T; \theta)$$

Supervised learning case study: Classification of time-series

AliveCore ECG device

ECG = electrocardiogram

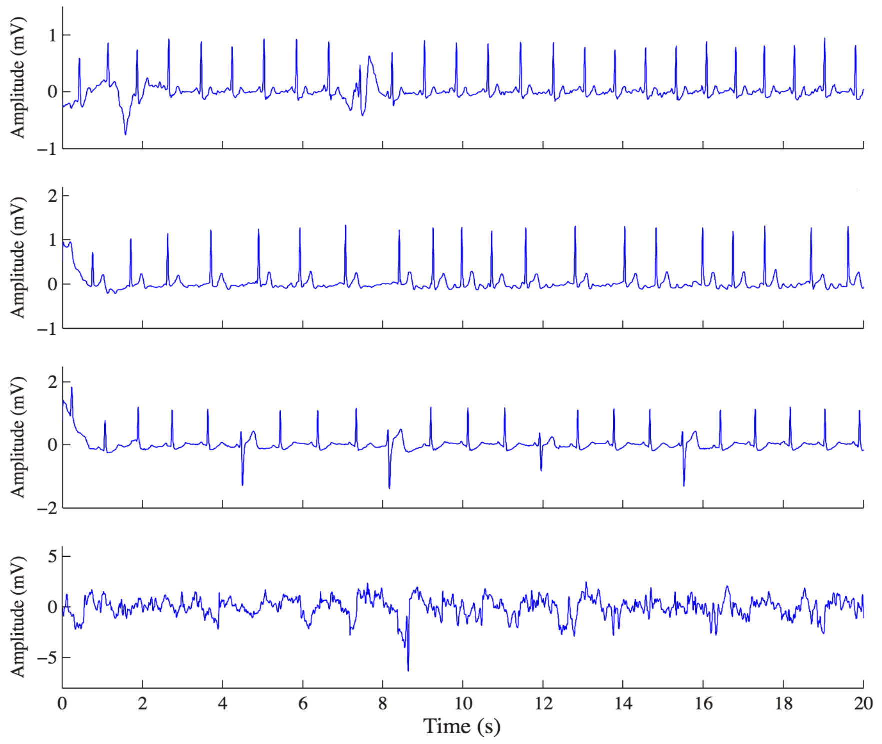
Classification of heart rhythm

Normal rhythm

AF rhythm

Other rhythm

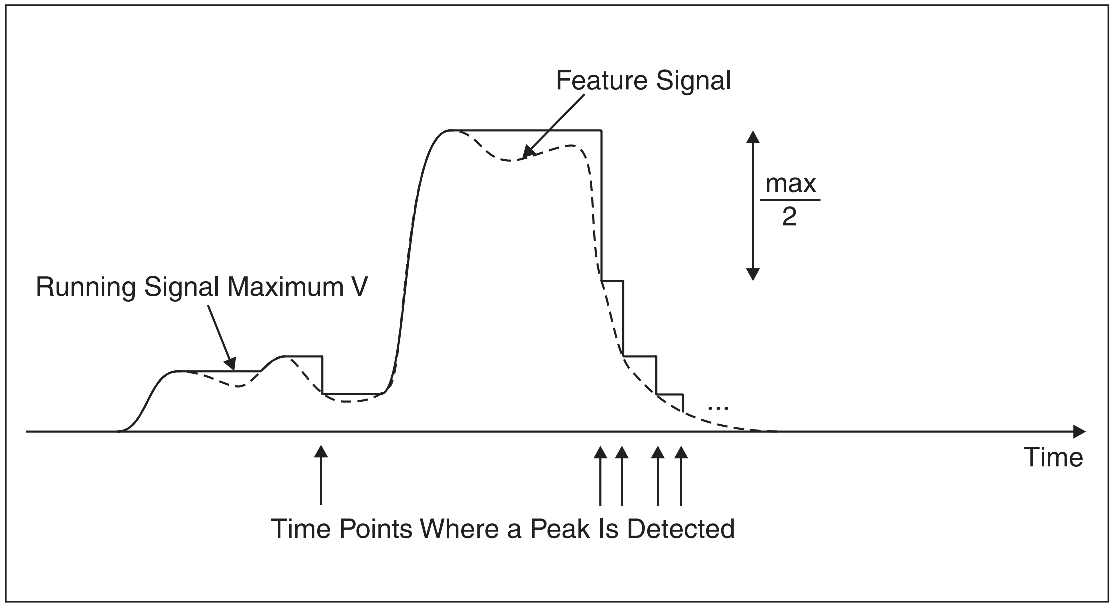
Noisy recording



[Clifford, Liu, Moody, Mark. PhysioNet Computing in Cardiology Challenge 2017]

Why should we care about this problem?
Atrial fibrillation (A-fib) is an irregular and often very rapid heart rhythm (arrhythmia) that can lead to blood clots in the hea

Traditional approaches to this problem



3. Peak detector proposed in [41].

[Kohler, Hennig, Orglmeister. The Principles of Software QRS Detection, IEEE Engineering in Medicine & Biology, 2002]

Proceedings Computers in Cardiology (1991)

Detection of Atrial Fibrillation Using Artificial Neural Networks

SG Artis, RG Mark, GB Moody

Harvard-MIT
Division of Health Sciences and Technology, Cambridge, MA

Abstract

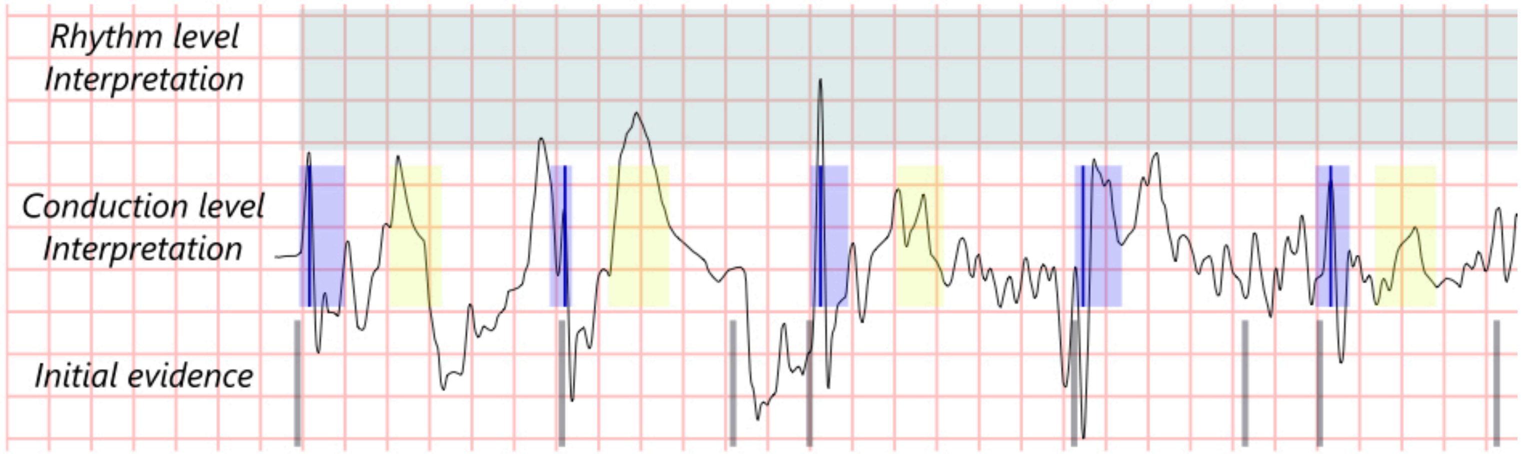
Artificial neural networks (ANNs) were used as pattern detectors to detect atrial fibrillation (AF) in the MIT-BIH Arrhythmia Database. ECG data was represented using generalized interval transition matrices, as in Markov model AF detectors[1]. A training file was developed, using these transition matrices, for a backpropagation ANN. This file consisted of approximately 15 minutes each of AF and non-AF data. The ANN was succesfully trained using this data. Three standard databases were used to test network performance. Postprocessing of the ANN output yielded an AF sensitivity of 92.86% *and an AF positive predictive accuracy of* 92.34%.

1 Introduction

on R-R interval sequences using a variety of statistical methods [1] but there is room for improvement in these techniques.

Pattern classifiers exist in many forms, and artificial neural networks (ANNs) represent an important subset of these classifiers. ANNs are attractive for solving pattern recognition problems because few assumptions about the underlying data need to be made. The task of the operator of an ANN is to separate the data into subsets. The network will be able classify these subsets according to type as long as they are distinct. Neural network training requires appropriate training data, pre-processing and post-processing algorithms, an appropriate network topology, and a training algorithm, as well as evaluation databases. This document will present the design and evaluation of a technique which detects AF in the presence of other cardiac arrhythmias using a backpropagation artificial neural network.

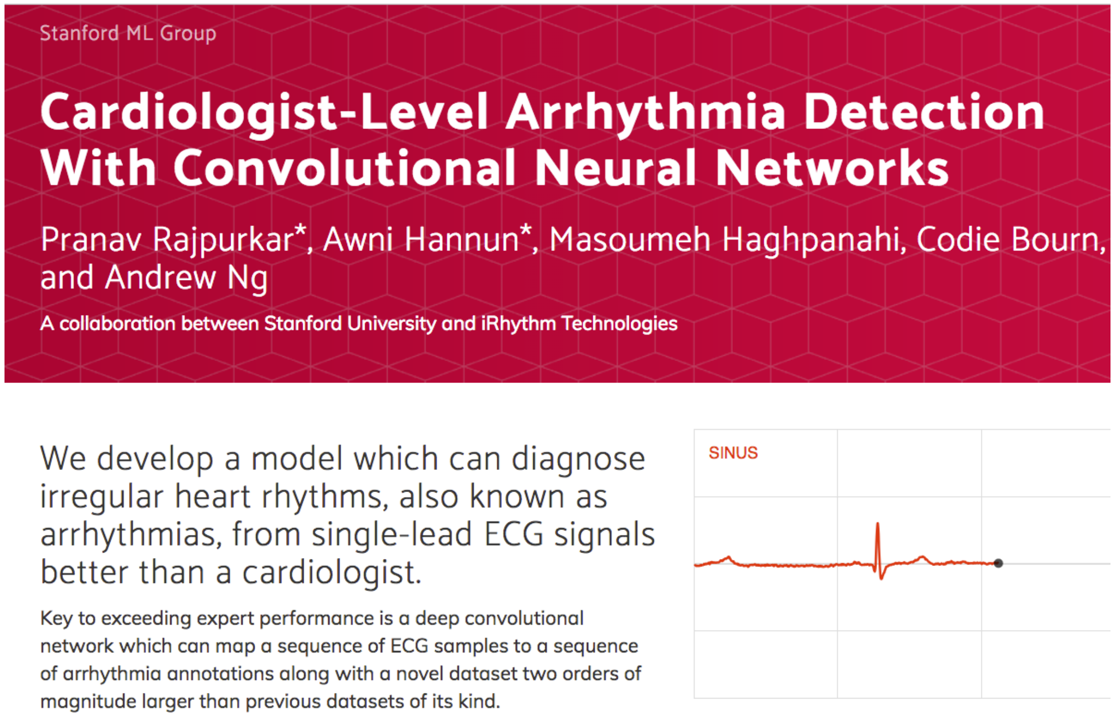
Physionet 2017 challenge [~8K ECGs]



[Teijeiro, Garcia, Castro, Felix. arXiv:1802.05998, 2018]

Best approach uses a combination of expert-derived and ML features

Diagnosing arrhythmia



Stanford ML Group

Cardiologist-Level Arrhythmia Detection With Convolutional Neural Networks

Pranav Rajpurkar*, Awni Hannun*, Masoumeh Haghpanahi, Codie Bourn, and Andrew Ng

A collaboration between Stanford University and iRhythm Technologies

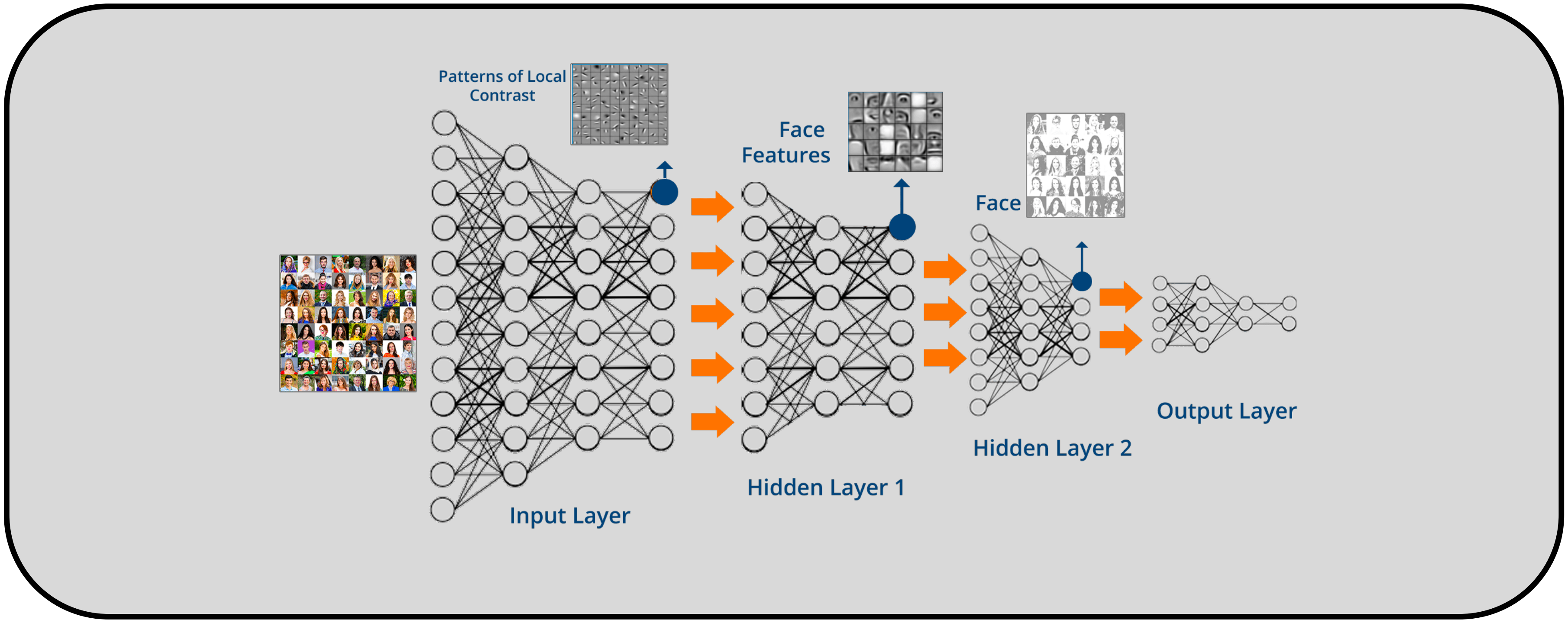
We develop a model which can diagnose irregular heart rhythms, also known as arrhythmias, from single-lead ECG signals better than a cardiologist.

Key to exceeding expert performance is a deep convolutional network which can map a sequence of ECG samples to a sequence of arrhythmia annotations along with a novel dataset two orders of magnitude larger than previous datasets of its kind.

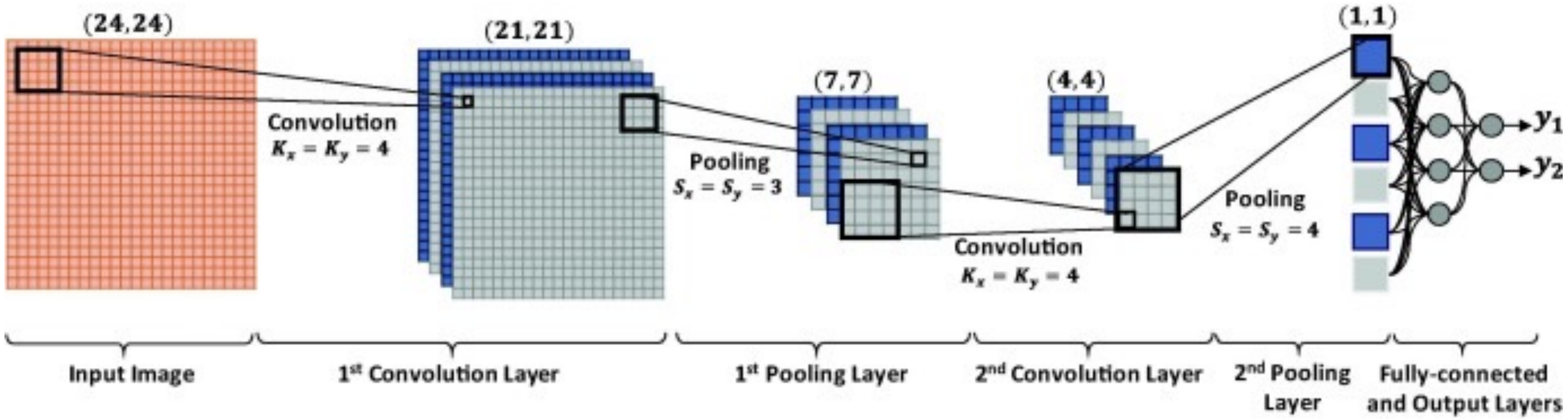
Use a Zio sensor to extract single-lead ECG signals

[Rajpurkar et al., arXiv:1707.01836, 2017; Nature Medicine '19]

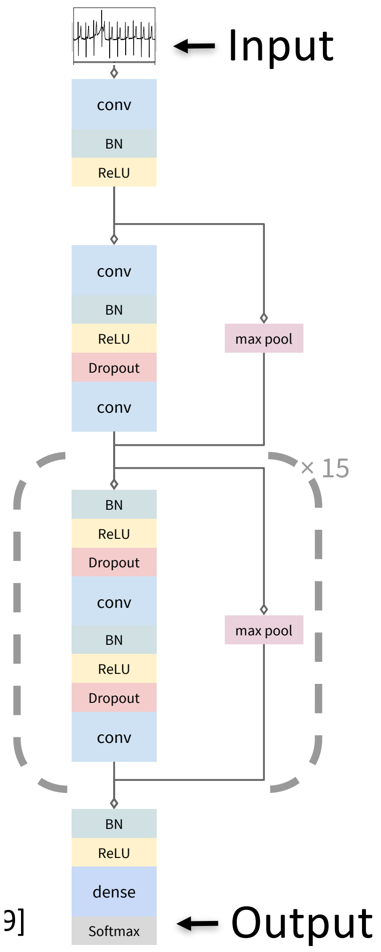
Neural networks in a picture



Convolutions for 2D data



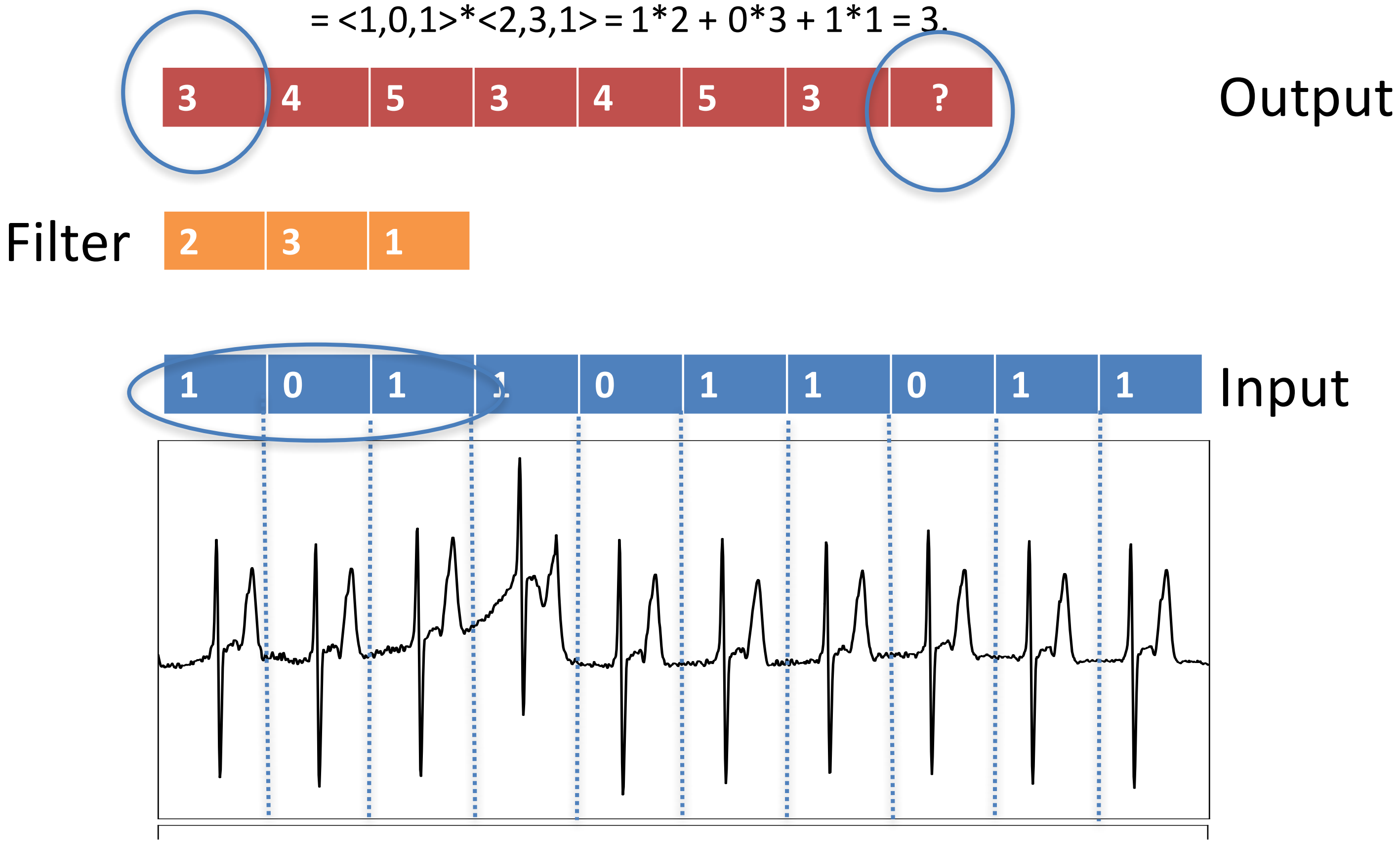
Using stacks of 1D convolutions to make predictions

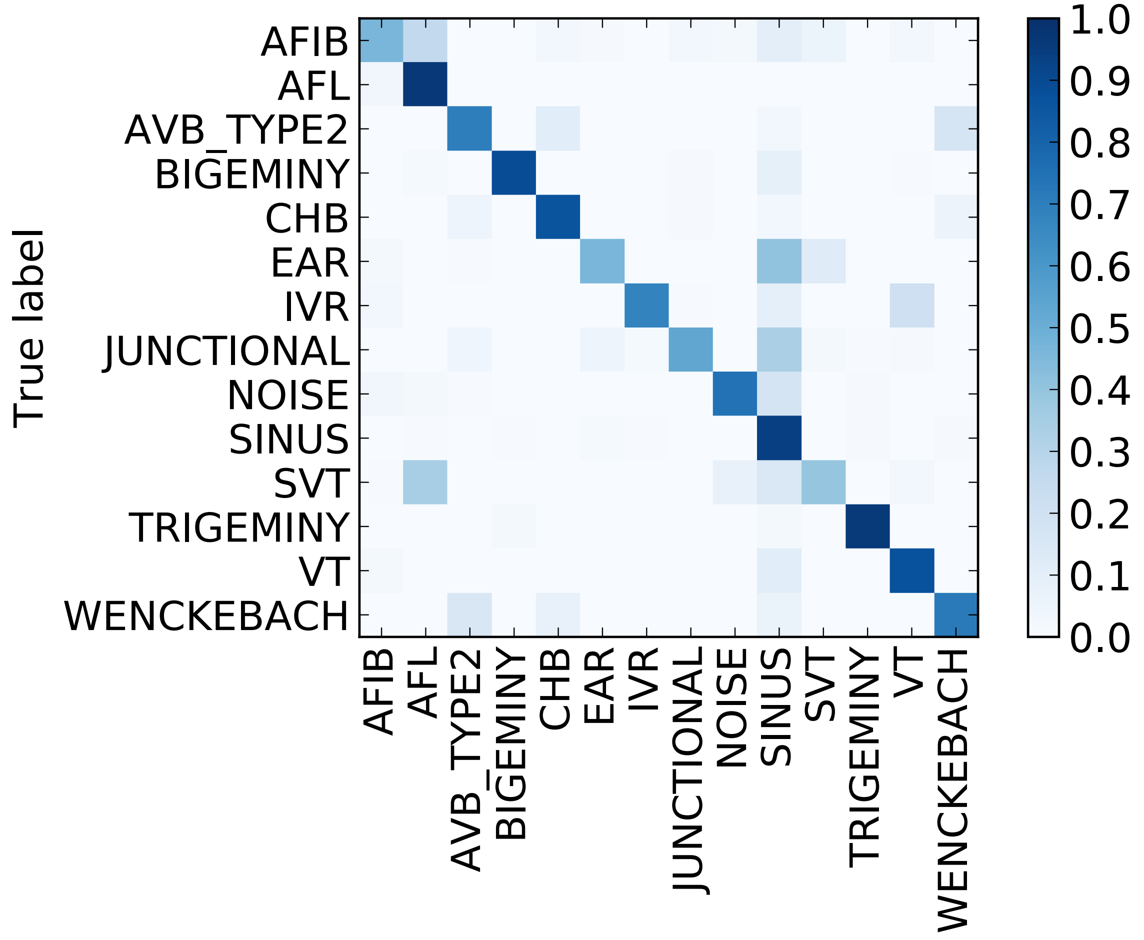


9]

Example of 1D convolution

= <1,0,1>*<2,3,1> = 1*2 + 0*3 + 1*1 = 3.





True label
Predicted label
AFIB
AFL
AVB_TYPE2
BIGEMINY
CHB
EAR
IVR
JUNCTIONAL
NOISE
SINUS
SVT
TRIGEMINY
VT
WENCKEBACH
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0