Recurrent neural networks

Lecture 12

Changho Suh

September 30, 2021

LSTM

Outline

Will study LSTM in details.

1. Discuss a brief history and key aspects of LSTM.

2. Discuss the key idea of LSTM.

3. Study how it works.

LSTM cell (1997)

Invented by:

Sepp Hochreiter Jürgen Schmidhuber

Performs much better by simply replacing a basic cell.

Offers faster training and detects dependencies in data.

Idea of LSTM cell

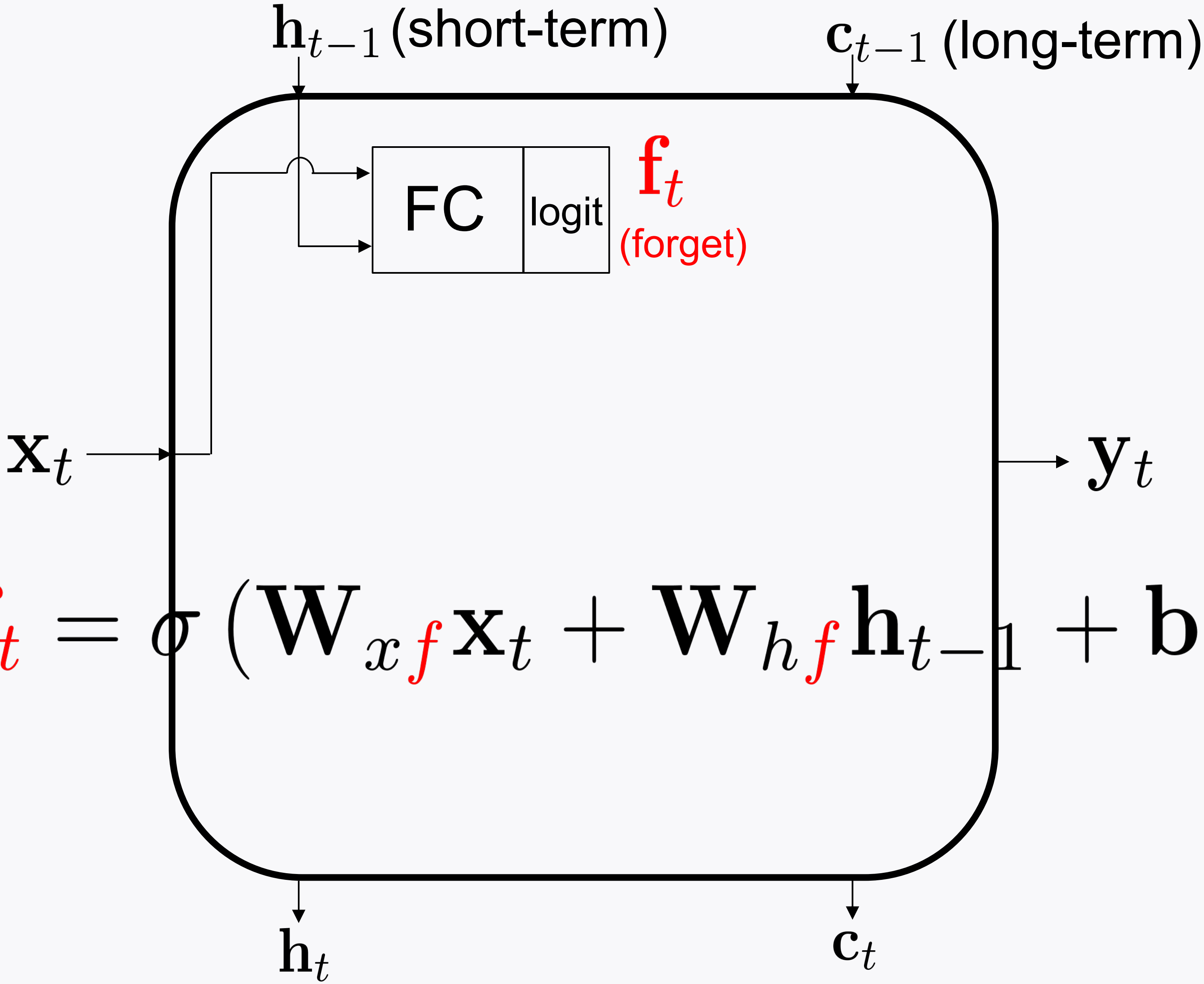
Split a state into two:

1. Short-term state $\mathbf{h}_t$
2. Long-term state $\mathbf{c}_t$

Design a cell so that the network can learn:

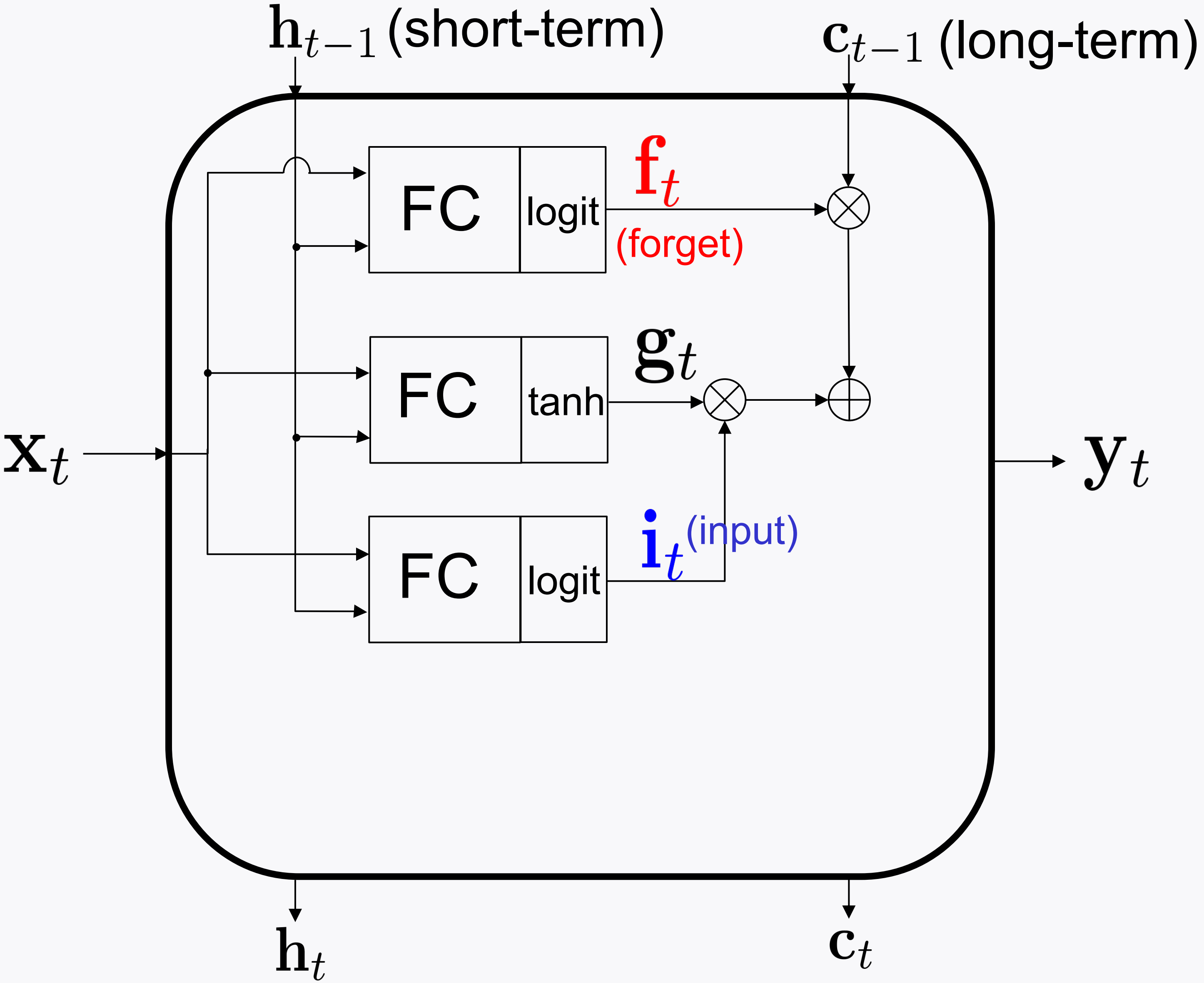
1. What to throw away (forget);
2. What to remember (input);
3. What to read (output).

Structure of LSTM cell

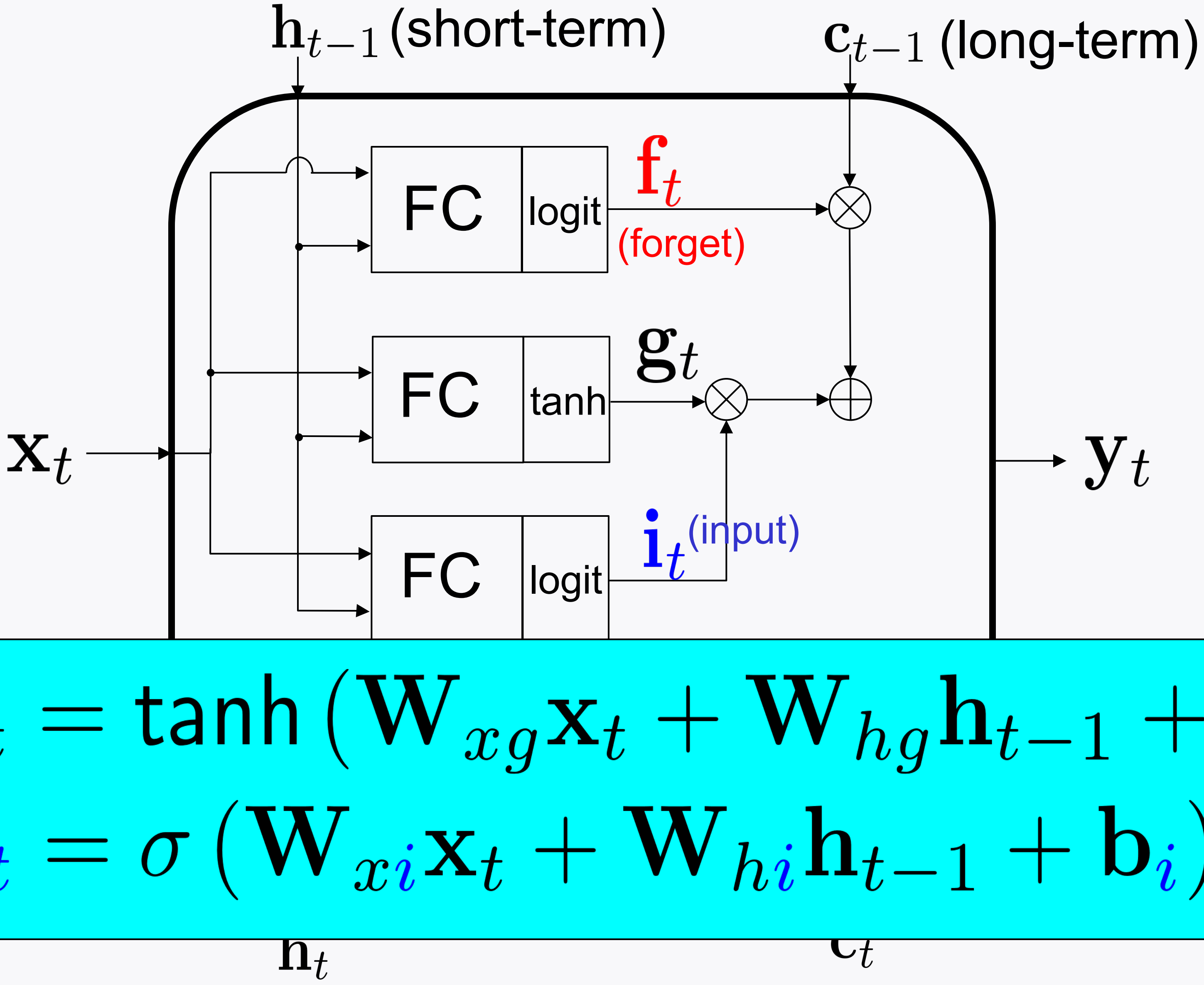


$$\mathbf{f}_t = \sigma\left(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f\right)$$

Structure of LSTM cell



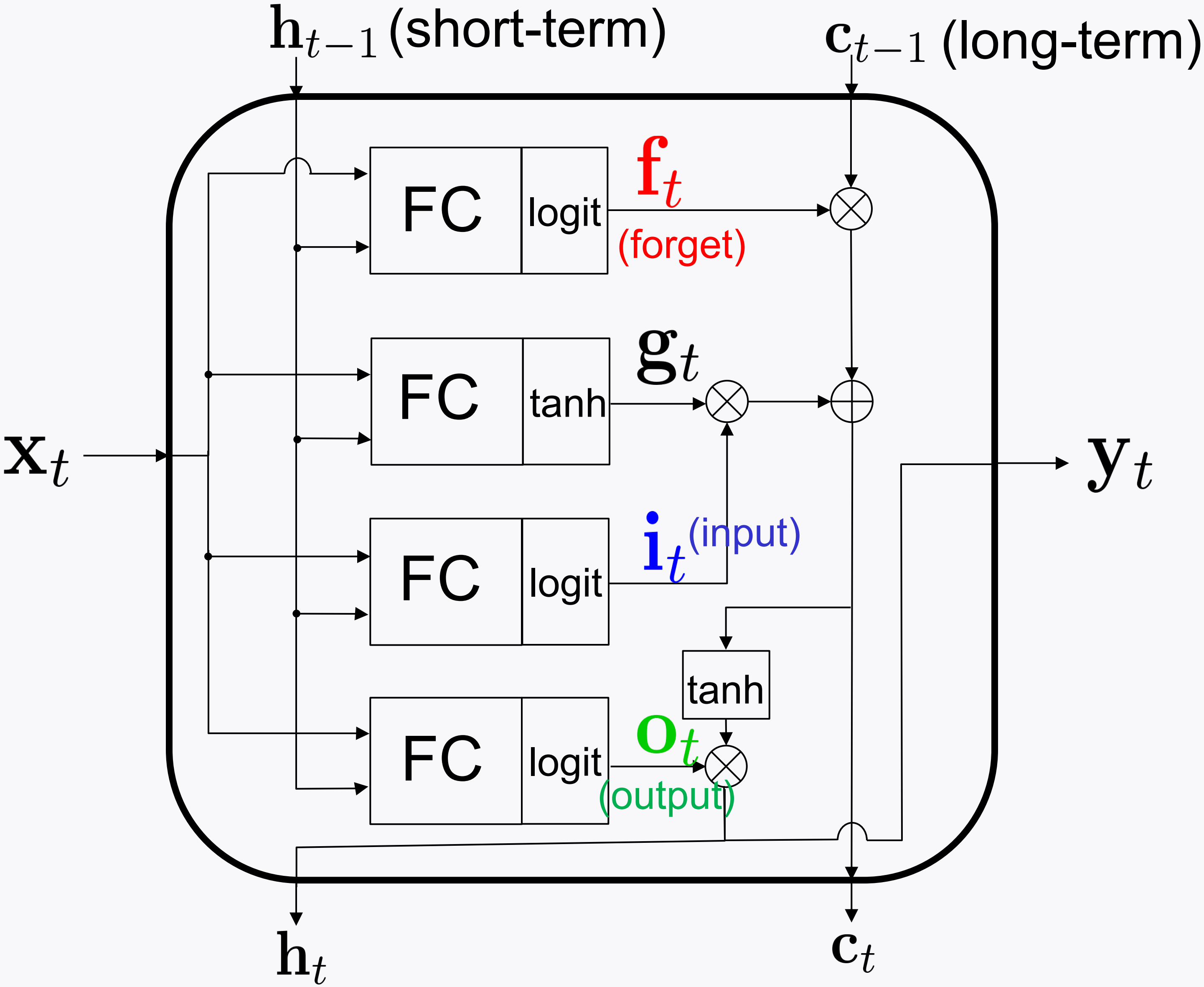
Structure of LSTM cell



$$\mathbf{g}_t = \tanh\left(\mathbf{W}_{xg}\mathbf{x}_t + \mathbf{W}_{hg}\mathbf{h}_{t-1} + \mathbf{b}_g\right)$$

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i\right)$$

Structure of LSTM cell



Mathematical expression

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f\right)$$

$$\mathbf{g}_t = \tanh\left(\mathbf{W}_{xg}\mathbf{x}_t + \mathbf{W}_{hg}\mathbf{h}_{t-1} + \mathbf{b}_g\right)$$

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i\right)$$

$$\mathbf{c}_t = \mathbf{f}_t \otimes \mathbf{c}_{t-1} + \mathbf{i}_t \otimes \mathbf{g}_t$$

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o\right)$$

$$\mathbf{y}_t = \mathbf{h}_t = \mathbf{o}_t \otimes \tanh(\mathbf{c}_t)$$

A simplified version of LSTM

A simplified version was developed in 2014:

Both states are merged into one.

Yet it performs just as well.

Kyunghyun Cho

Applications

Turns out: LSTM and/or its variants work well in many applications:

Machine translation

Text generation

Grammar correction

Any natural language processing (NLP) applications

So far ...

Studied several models:

Least squares

Logistic regression

DNN

CNN

RNN

Questions

1. What if still unsatisfactory performances?

   A better approach for the small-data regime?

2. What about interpretability of DNNs?

Day 5 lectures

Will explore a technique that may enable a better performance for the small-data regime, as well as offer model interpretability:

Random forests (RFs)

The **most powerful** ML algorithm in industry

Outline of Day 5 lectures

Specifically we will study:

1. **Decision trees (DTs)**:

   Fundamental components of RFs

2. **Ensemble learning**:

A generic technique that includes RFs as a special case.

3. **RFs** in depth