Machine Learning Techniques for Battery State Prediction and Time Series Forecasting

Michael Bosello

Università di Bologna – Department of Computer Science and Engineering, Cesena, Italy

Outline

Machine Learning I

- A ML algorithm is an algorithm that is able to learn from data

  "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E"

  [Mitchell, 1997]

- It is essentially a form of applied statistics
- Enables us to tackle tasks that are too difficult to solve with fixed programs
  - Use it only when no closed-form solution is available/applicable

Machine Learning II

Experience E

- Set of examples
- An example is a collection of features
- An example is represented as a vector

Performance P

- Quantitative measure like accuracy

Task T

We are interested in:

- Classification
- Regression
- Time series forecasting

$\rightarrow$ Supervised Learning
  - each example is associated with a target

Machine Learning III

Algorithm Components

- Model
  - parametric mathematical model
- Optimization algorithm
  - that will improve the weights in a way that reduces error
- Cost function
  - e.g. MSE, negative log-likelihood
- Dataset
  - training examples

We can replace any of these components mostly independently

Machine Learning IV

Generalization

- It is the ability to perform well on previously unobserved inputs
- It is what separates machine learning from optimization
- How can we affect performance on the test set when we can observe only the training set?
  - We typically make a set of assumptions known as the i.i.d. (independent and identically distributed)
  - The examples are independent of each other
  - The training set and test set are drawn from the same probability distribution

Applaying ML

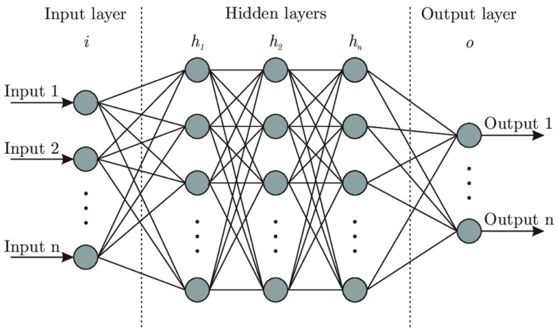
We have to:

- choose the features
- perform data pre-processing
- choose the model/algorithm

hyperparameters

- Algorithm's settings
- Hyperparameter tuning: "more an art than a science"

Artificial Neural Networks



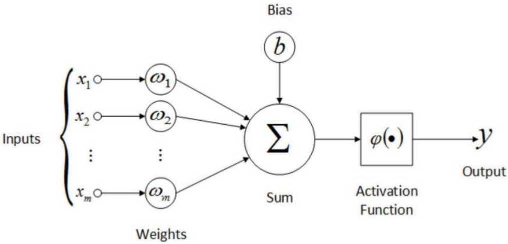
The goal of a network is to **approximate** some function f^* ; $y = f^*(x)$

- The network defines a mapping $y = f(x; \theta)$
- It learns the value of the parameters θ that result in the best approximation

A directed *acyclic* graph describes how functions are connected together

- Acyclic $\leftrightarrow$ **Feedforward**
- Functions connected in a chain (vector-to-vector function)
- Sets of neurons organized in layers (vector-to-scalar function)

Artificial Neural Networks II



- The Activation function should be nonlinear
- Example: Rectified linear function $g(z) = max0, z$

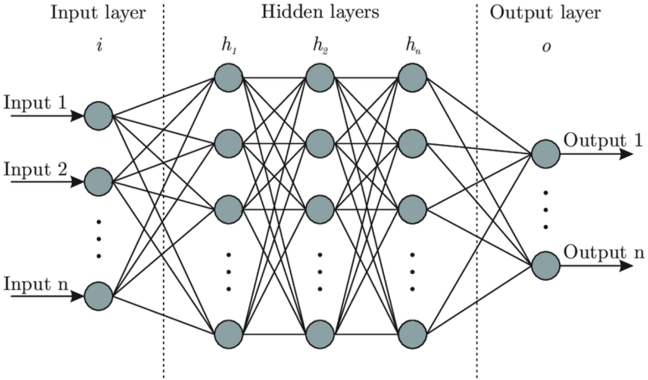
Training

- **SGD** Improve f by moving in small steps with opposite sign of the derivative
- **Backpropagation** A method to compute gradient efficiently

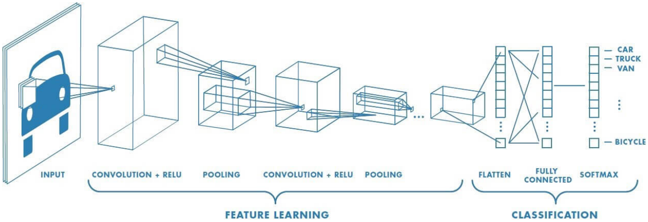
Nonlinearity

- Manually engineering a nonlinear mapping is very difficult
- Deep learning learns the mapping thanks to the hidden layers

Fully Connected Network

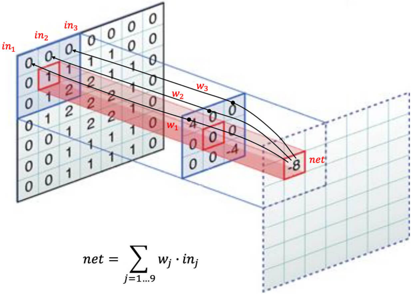


Convolutional Neural Network

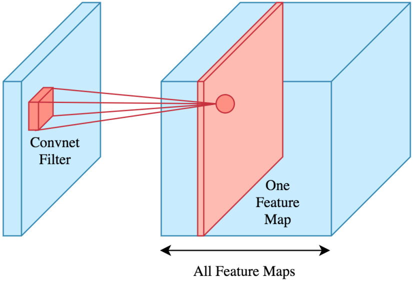


- Assumptions on data allow reducing the number of parameters
- CNNs perform better on structured and spatially related data
- Neurons arranged in N-dimensional volumes
- Layers of a ConvNet transforms one volume of activations to another

Convolution Layer



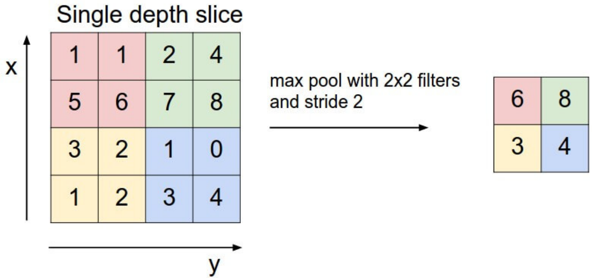
$$net = \sum_{j=1...9} w_j \cdot in_j$$



- The Conv layer's parameters consist of a set of learnable filters
- Every filter is small spatially but extends through the full depth of the input volume
- we slide (convolve) each filter across the width and height of the input volume
- The activation map gives the responses of a filter at every spatial position
- Filters' responses are stacked along the depth producing the output volume

- neurons are connected only to a local region in the input
- neurons in a filter share parameters

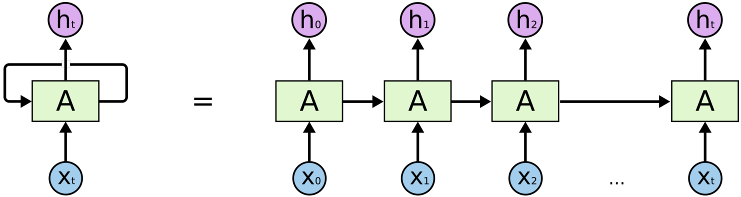
Pool Layer



It performs a downsampling operation along the spatial dimensions

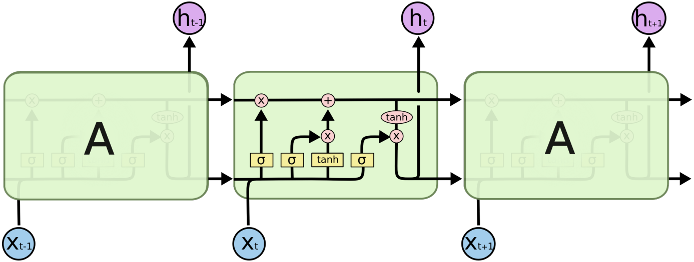
- Reduce spatial size/parameters
- Controls overfitting

Recurrent NN



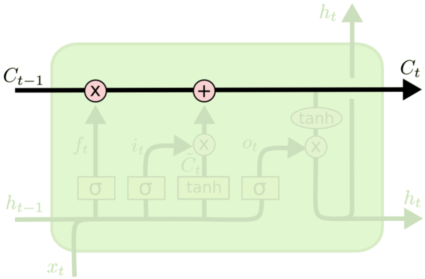
- RNNs allow information to persist
- Good for sequences
- They can retain only recent information

Long Short-Term Memory Networks I



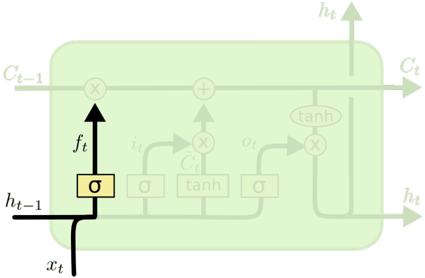
- A special kind of RNN
- Capable of learning long-term dependencies

Long Short-Term Memory Networks II



- Cell state layer
- It's very easy for information to just flow along it unchanged (long term)
- Removing or adding information to the cell state is carefully regulated by structures called gates
- Weights of gates regulate memorization

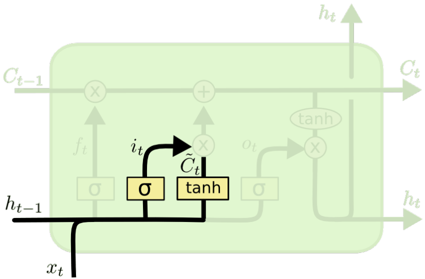
Long Short-Term Memory Networks III



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

- Forget gate layer
- Sigmoid output in $[0, 1]$
- For each element in the cell state

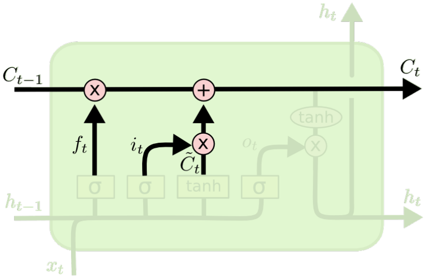
Long Short-Term Memory Networks IV



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

- Input gate layer
- Tanh layer creates a vector of new candidate values

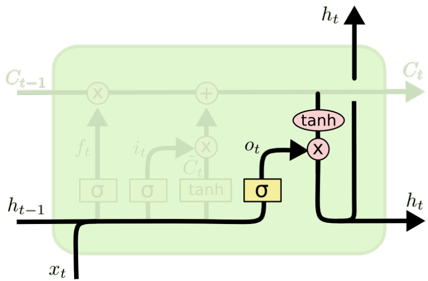
Long Short-Term Memory Networks V



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- Update

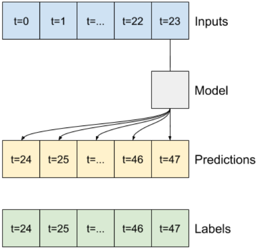
Long Short-Term Memory Networks VI



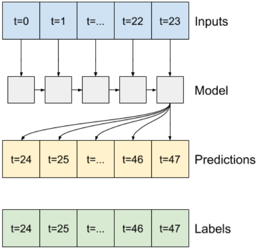
$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

- Output
- Only one part of the cell is given

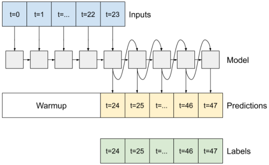
Time Series



Feedforward



Recurrent NN



Autoregressive

- Data windowing
  - Deciding the width (number of time steps) of the input and output
  - Deciding the time offset between them

Why ML?

Why NNs for SoC/SoH?

- Less computationally expensive than other algorithms
- No model engineering
- Generalization
- Nonlinear

Why NNs for TS forecasting?

Growing evidence suggests that machine learning approaches offer a superior modeling methodology to tackle time-series

Other techniques

Meta-learning

- A model is trained on a set of (correlated) tasks
- Two gradient descent phases
- The model generalizes to other tasks without re-training

Bayesian Network (Belief Network)

- Directed Acyclic Graph
- Nodes are random variables
- Structure/parameters could be learned from data
- **Dynamic Bayesian network** is the extension to time series
- **Decision graphs** handle decision making under uncertainty
- Hidden Markov Model
  - Special case
  - A process Y depends on X which is unobservable

ARIMA (time series model)

Appendix: References

References Index I

- Introduction to ML and ANNs [Goodfellow et al., 2016]
- CNNs
  - https://cs231n.github.io/convolutional-networks/
  - [Krizhevsky et al., 2012]
- LSTM
  - https://colah.github.io/posts/2015-08-Understanding-LSTMs/
  - [Hochreiter and Schmidhuber, 1997]
- Model Agnostic Meta-Learning [Finn et al., 2017]
- Model Bayesian Networks [Russell and Norvig, 2009] Chapters:
  - 14 – Probabilistic Reasoning
  - 15 – Probabilistic Reasoning over Time
  - 20 – Learning Probabilistic Models

References Index II

- Review of SoC/SoH estimation using ML [Vidal et al., 2020]
- Time series forecasting
  - Review of ANNs methodologies for TS [Tealab, 2018]
  - TS forecasting using meta-learning [Oreshkin et al., 2020]
  - TS forecasting using LSTM [Hua et al., 2019] [Guo et al., 2016]
  - TS classification using CNN [Zhao et al., 2017]

References I

Finn, C., Abbeel, P., and Levine, S. (2017).
Model-agnostic meta-learning for fast adaptation of deep networks.

Goodfellow, I., Bengio, Y., and Courville, A. (2016).
Deep Learning.
MIT Press.
http://www.deeplearningbook.org.

Guo, T., Xu, Z., Yao, X., Chen, H., Aberer, K., and Funaya, K. (2016).
Robust online time series prediction with recurrent neural networks.
In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 816–825.

Hochreiter, S. and Schmidhuber, J. (1997).
Long short-term memory.
Neural computation, 9:1735–80.

Hua, Y., Zhao, Z., Li, R., Chen, X., Liu, Z., and Zhang, H. (2019).
Deep learning with long short-term memory for time series prediction.
IEEE Communications Magazine, 57(6):114–119.

References II

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012).
Imagenet classification with deep convolutional neural networks.
In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Mitchell, T. (1997).
Machine Learning.
McGraw-Hill International Editions. McGraw-Hill.

Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. (2020).
Meta-learning framework with applications to zero-shot time-series forecasting.

Russell, S. and Norvig, P. (2009).
Artificial Intelligence: A Modern Approach.
Prentice Hall Press, USA, 3rd edition.

Tealab, A. (2018).
Time series forecasting using artificial neural networks methodologies: A systematic review.
Future Computing and Informatics Journal, 3(2):334 – 340.

References III

Vidal, C., Malysz, P., Kollmeyer, P., and Emadi, A. (2020).
Machine learning applied to electrified vehicle battery state of charge and state of health estimation: State-of-the-art.
IEEE Access, 8:52796–52814.

Zhao, B., Lu, H., Chen, S., Liu, J., and Wu, D. (2017).
Convolutional neural networks for time series classification.
Journal of Systems Engineering and Electronics, 28(1):162–169.

Machine Learning Techniques for Battery State Prediction and Time Series Forecasting

Michael Bosello

Università di Bologna – Department of Computer Science and Engineering, Cesena, Italy