# Convolutional Neural Networks, image recognition and financial time series forecasting

ECML-PKDD 2019 Workshop MIDAS

Sept. 16, 2019 - Wurzburg, Germany

**Argimiro Arratia** and Eduardo Sepúlveda

Universitat Politècnica de Catalunya, SPAIN

# Motivation

- Convolutional Neural Networks (CNN) are best known as good image classifiers
- CNN has recently been used for financial forecasting
- Our goal: to show that by converting financial information into images and feeding these financial-image representation to the CNN, it results in an improvement in classification.

# Methods and Data I

## Recurrence Plots [Eckman et al 1987]

Formalized as a matrix $\mathcal{R} = (R(i,j))$ where each entry

$$R(i,j) = \Theta(\varepsilon - ||\vec{x}(i) - \vec{x}(j)||), \quad \vec{x}(\cdot) \in \mathbf{R}^m, i,j = 1 \ldots N \tag{1}$$

where $N$ is the number of states, $\vec{x}(i)$ is the subsequence observed at time $i$, $||\cdot||$ is a norm, $\varepsilon$ is a threshold for closeness and $\Theta$ is the Heaviside function ($\Theta(z) = 0$, if $z < 0$, or 1 otherwise).

Thus, if the $m$-dimensional trajectory of the time series at time $i$ is close (w.r.to $||\cdot||$) to the subsequence observed at time $j$, there will be a 1 (e.g. yellow square) at entry $(i,j)$ of $\mathcal{R}$; otherwise, the value is 0 (a dark purple square).

# Methods and Data II

In our experiments we use the pairwise Euclidean distance for the RP norm $||\cdot||$.

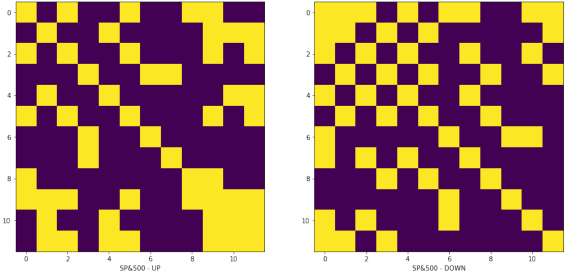


Figure: RP for S&P500

# Convolutional Neural Networks (CNN) I

CNN are made up of hidden nodes (or neurons), distributed through various layers, with learnable weights and biases. Each neuron receives several inputs and computes a weighted sum over them, and then passes the result through an activation function which gives an output.

Popular choices for activation functions are:

- logistic sigmoid: $\sigma(x) = 1/(1 + \exp(-x))$,
- hyperbolic tangent: $\tanh(x) = 2/(1 + \exp(-2x)) - 1$,
- rectified linear units: ReLu, $R(x) = \max(0, x)$.

# Convolutional Neural Networks (CNN) II

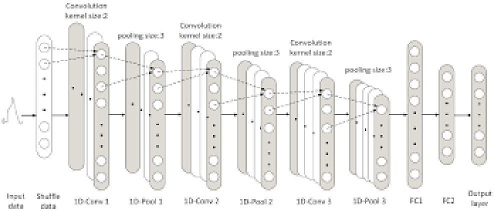


Figure: Basic functioning of 1D-CNN

# Convolutional Neural Networks (CNN) III

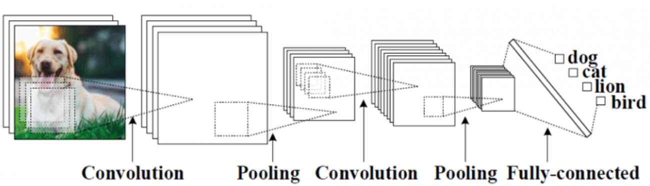


Figure: Basic functioning of 2D-CNN

The network's parameters are tuned by minimizing some loss function.

As opposed to regular neural networks, nodes in a CNN are not fully connected but only connected to a local region in the inputs. This local connectivity is attained by using convolutions instead of weighted sums.

# Convolutional Neural Networks (CNN) IV

The dimension of the convolution operator accommodates to the dimension of the data.

For one-dimensional input, e.g. time series $x = \{x(t) : t = 1, \dots, N\}$, one-dimensional (1D) convolutions with kernels $w_h$, for $h = 1, \dots, M$ with $M < N$, are formally defined as

$$s(t, h) = (w_h \star x)(t) = \sum_{n=-\infty}^{\infty} x(n) w_h(t - n) \tag{2}$$

For two-dimensional input, e.g. images $(I(i, j))_{1 \le i, j \le N}$, the appropriate convolutions must be two-dimensional (2D) on kernel matrices $(K_h(i, j))_{1 \le i, j \le M}$:

$$s(i, j, h) = (K_h \star I)(i, j) = \sum_m \sum_n I(i - m, j - n) K_h(m, n) \tag{3}$$

# Convolutional Neural Networks (CNN) V

We build two types of CNN by sequential modeling using the Keras library in Python.

Conv1D: Consists of one convolutional layer made of one-dimensional convolutions with kernels of size 2 and 64 nodes. Inputs are numeric arrays.
The activation function we use is the **ReLU**, which works quite well in practice and is less computationally expensive than **tanh** and **sigmoid** because it involves simpler mathematical operations.
The Pooling layer will be produced with Max pooling.

# Convolutional Neural Networks (CNN) VI

Conv2D + RP: For this model the inputs (arrays of numbers) are first pre-processed with the recurrence plot (RP) method to produce corresponding images (matrices of 0 and 1). Then follows a convolutional layer made of two-dimensional convolutions with kernels of size $2 \times 2$ and 64 nodes, a ReLu activation function, and a Max pooling layer.

# Datasets I

S&P 500 index. Target: the direction of monthly price of S&P 500 with reference to the risk-free interest rate (i.e. the sign of S&P 500 equity premium (GSPCep)).

$$GSPCep(t) = \log\left(\frac{P(t) + D12(t)}{P(t-1)}\right) - \log(r(t) + 1)$$

where $P(t)$ and $D12(t)$ are, respectively, the monthly price and the 12-month moving sums of dividends paid on the S&P 500 index at time $t$, and $r(t)$ is the interest rate of the three months U.S. Treasury bill. Features: past history of the equity premium and its variance, up to three lags.

# Datasets II

US Banks. Target: categorical taking value 1 to indicate a bankrupt entity, or 0 otherwise.
Features: arrays with 106 exploratory variables pertaining to financial indicators drawn from quarterly financial reports of U.S. banks. There are 5152 of these arrays, each representing the financial situation of a bank (in 1992-2017). Each bank has at least 8 quarter periods of financial data reporting. This financial information have been retrieved from the Federal Deposit Insurance Corporation (FDIC).

# Experiments and results I

Experiments consist in comparing the CNN's performance with two different processing of its data inputs.

1. consists on feeding the 1-D numerical data directly as input to our CNN.
2. consists on first pre-processing the numerical data into images using the Recurrence Plot technique.

We compare the performance of the models with respect to: accuracy in classification, loss in training, AUC, Matthews Correlation and 10-fold cross-validation.

We do experiments in two different financial scenarios:
1) to predict direction of price of the S&P 500 index;
2) to predict the possibility of bankruptcy in a set of U.S. banks.

# Experiments and results II

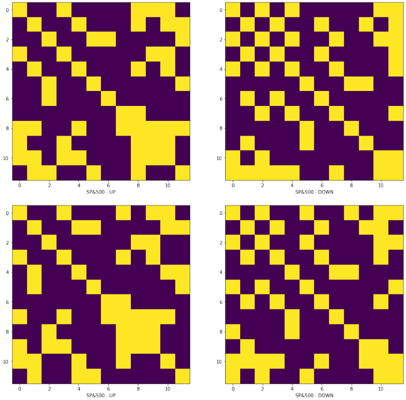


Figure: RP images of S&P 500 data inputs with target 0 (price down) or 1 (price up)

# Experiments and results III

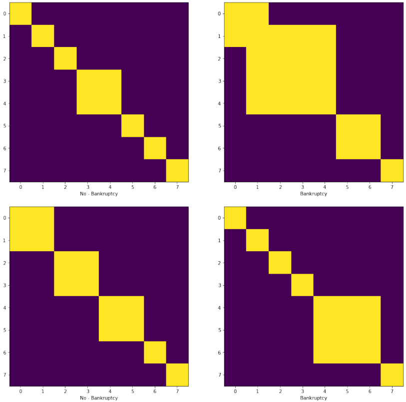


Figure: RP images of U.S. bank data with target 1 (bankrupt) or 0 (non-bankrupt)

# Experiments and results IV

We repeat each experiment 100 times and report below average results for each CNN model

Table: Performance of CNN without/with RP for S&P500 price prediction

| | Acc | Loss | 10-fold CV | AUC | Matthews cor. |
|---|---|---|---|---|---|
| Conv1D | 0.52 | 5.75 | 61.1% ($\pm$ 4.16%) | 0.65 | -0.102 |
| Conv2D + RP | 0.63 | 2.69 | 63.22% ($\pm$ 0.93%) | 0.66 | -0.0026 |

Table: Performance of CNN without/with RP for Bankruptcy Detection

| | Acc | Loss | 10-fold CV | AUC | Matthews cor. |
|---|---|---|---|---|---|
| Conv1D | 0.82 | 3.27 | 58.91% ($\pm$ 17.98%) | 0.72 | 0.42 |
| Conv2D + RP | 0.94 | 1.01 | 93.75% ($\pm$ 0.07%) | 0.83 | 0.67 |

# Experiments and results V

The improvement in accuracy, AUC, Matthews correlation and decrease in loss are significant when pre-processing data with RP before feeding it to the 2D Convolutional Neural Network.

# Conclusions

- We have shown that by making a previous processing of the input by recurrence plot transformation of our numerical data to images, we obtain better classification results as measured by five different metrics of classification performance.
- We have used a standard norm (Euclidean norm) for the RPs. Other norms are possible, in particular ones more suitable for capturing similarities among financial time series, like, for example, a correlation based metric. It would be interesting to see the difference in performance of Conv2D+RP model under different norms underlying the definition of the RP method.

# Convolutional Neural Networks, image recognition and financial time series forecasting

ECML-PKDD 2019 Workshop MIDAS

Sept. 16, 2019 - Wurzburg, Germany

**Argimiro Arratia** and Eduardo Sepúlveda

Universitat Politècnica de Catalunya, SPAIN

**Questions?**