

# An Application of Deep Learning to Algorithmic Trading for Time Series Forecasting Using Deterministic and Metaheuristic Optimization

Author: Khaykin G.A., Advisor: Lukyanchenko P.P.

National Research University  
«Higher School of Economics»

March 23, 2021

# Brief Overview

## 1 Background

- Problem Statement
- Purpose
- Tasks
- Data Description
- Technical Analysis Indicators
- Loss Functions & Performance Criteria

## 2 Models

- ARIMA
- FNN
- LSTM
- GA-LSTM
- PSO-LSTM

# Problem Statement

- Traditional time series forecasting methods as the ARIMA or GARCH models have been widely used to make predictions about the financial market.
- Alas, prediction systems based on statistical methods do not perform well since:
  - Financial markets are regarded as nonlinear and non-parametric dynamic systems.
  - Traditional methods have their own limitations.

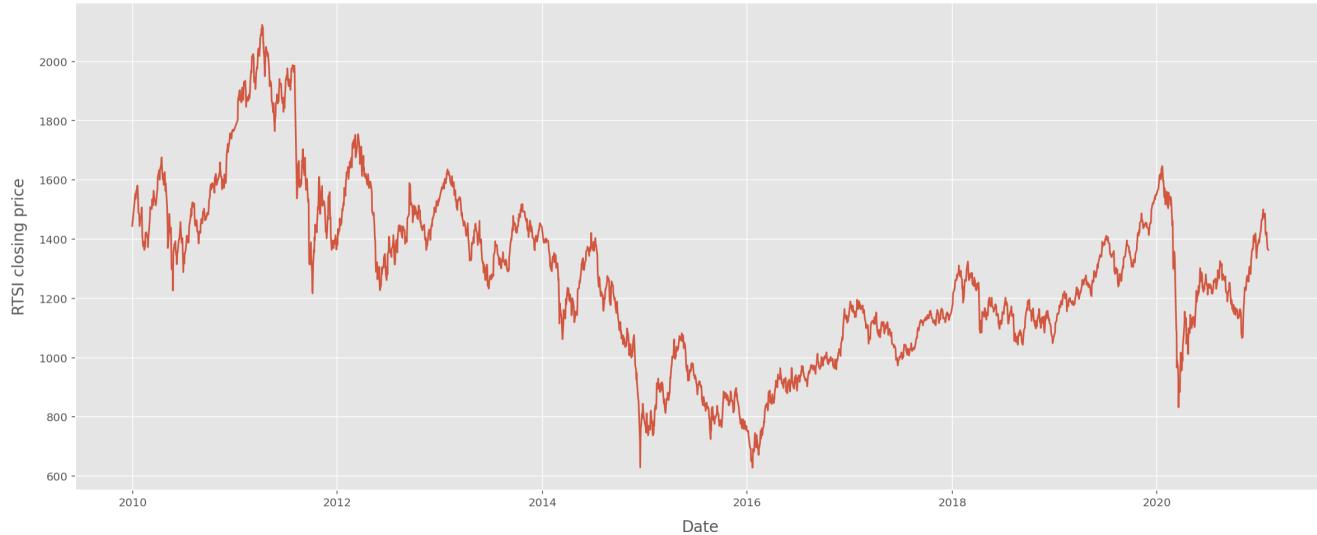
# Purpose

- Introduce methods that can learn complex dimensionality that are essential to improve the prediction performance:
  - LSTM;
  - PSO-LSTM;
  - GA-LSTM.

# Tasks

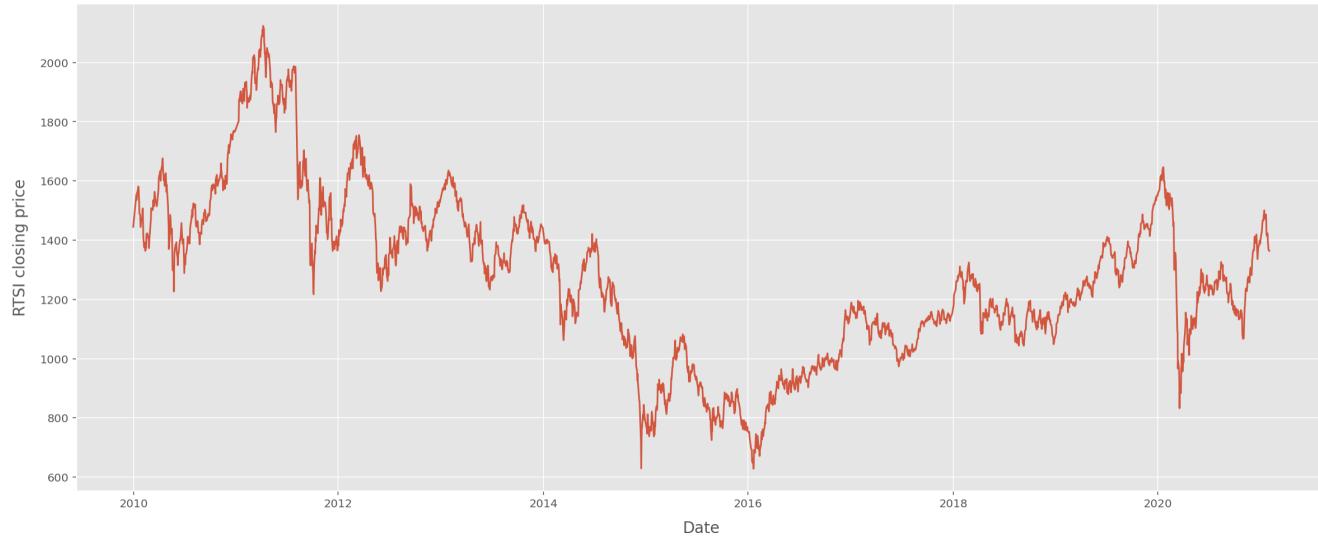
- ① Analysis of the current research.
- ② Data prepossessing, feature engineering.
- ③ ARIMA model as a benchmark.
- ④ LSTM, GA-LSTM, PSO-LSTM as alternatives.
- ⑤ Comparison of the models.

# Data Description



- Research data for trend forecasting of a stock index comes from January 2010 to December 2020. ↗ **RTSI**
- The dataset is divided into training (80%) and testing (20%). From the training dataset, we comprise a validation dataset (20% of training dataset).

# Data Description



- Historical data is obtained from Bloomberg.
- Each sample contains:
  - Daily price information (low price, high price, opening price, closing price, trading volume).
  - Technical indicators

# Technical Analysis Indicators

- $n$ -day  $SMA_i = \frac{y_i + y_{i-1} + y_{i-2} + \dots + y_{i-n}}{n}$
- $n$ -day  $WMA_i = \frac{n \times y_i + (n-1) + \dots + y_{i-n}}{[n \times (n-1)]/2}$
- $ROC_i = \left[ \frac{y_i}{y_{i-n}} - 1 \right] \times 100$
- $RSI_i = 100 - \left[ \frac{100}{1 + RS_i} \right], RS_i = \frac{\sum_{i=1}^{n-1} Up_{i-1}}{\sum_{i=1}^{n-1} Down_{i-1}}$
- $n$ -day Stochastic oscillators:  
$$\%K_t = \frac{y_t - LL_{t-n}}{HH_{t-n} - LL_{t-n}} \times 100$$
$$\%D_t = \frac{\%K_{t-1} + \dots + \%K_{t-n}}{n}$$

# Loss Functions & Performance Criteria

- Loss functions:

$$\textcircled{1} \quad RMSE(\hat{y}, X^\ell) = \sqrt{\frac{1}{\ell} \sum_{i=1}^{\ell} (y_i - \hat{y}(x_i))^2}$$

$$\textcircled{2} \quad SMAPE(\hat{y}, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} \frac{|y_i - \hat{y}(x_i)|}{(|y_i| + |\hat{y}(x_i)|)/2} \times 100$$

- Evaluation metric:

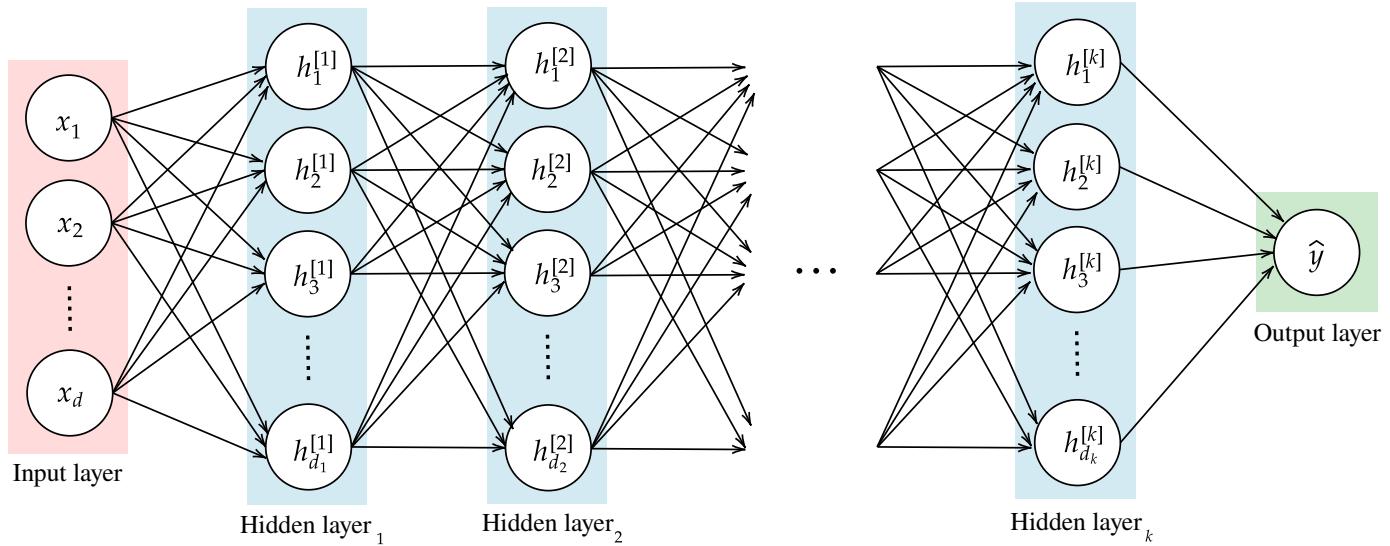
$$R_{adj}^2(\hat{y}, X^\ell) = 1 - \frac{\frac{\sum_{i=1}^{\ell} (y_i - \hat{y}(x_i))^2}{\ell - p}}{\frac{\sum_{i=1}^{\ell} (y_i - \bar{y})^2}{\ell - 1}}, \bar{y} = \frac{1}{\ell} \sum_{i=1}^{\ell} y_i,$$

$p$  — total number of explanatory variables in the model

# ARIMA

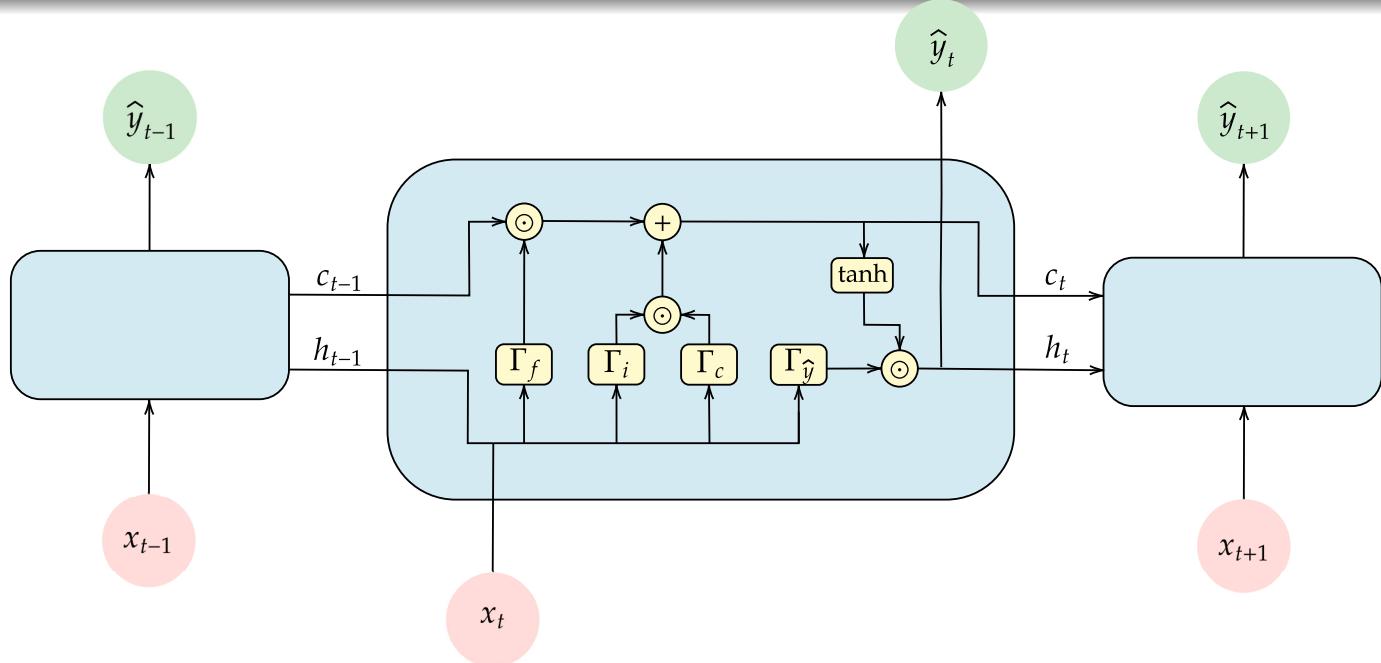


# Feedforward Neural Network



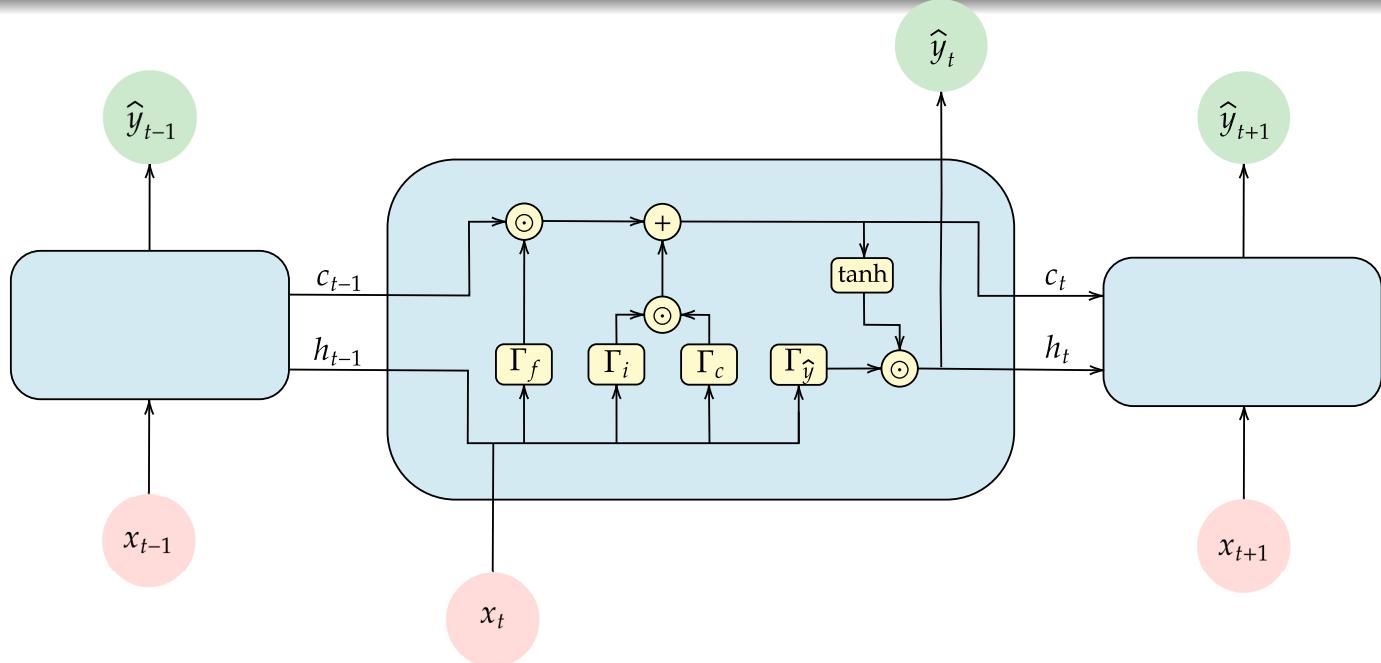
- $h_j^{[1]} = f \left( \langle w_j^{[1]}, x \rangle + w_{0j}^{[1]} \right), j \in (1, \dots, d_1)$
- $h_j^{[i]} = f \left( \langle w_j^{[i]}, h^{[i-1]} \rangle + w_{0j}^{[i]} \right), i \in (2, \dots, k), j \in (1, \dots, d_i)$

## LSTM



- The cell state at time t:  $c_t = \Gamma_i \odot \Gamma_c + \Gamma_f \odot c_{t-1}$
- The hidden state at time t  $h_t = \Gamma_{\hat{y}} \odot \tanh(c_t)$
- Output gate:  $\Gamma_{\hat{y}} = \sigma(W_{xy}x_t + W_{hy}h_{t-1} + b_y)$

# LSTM

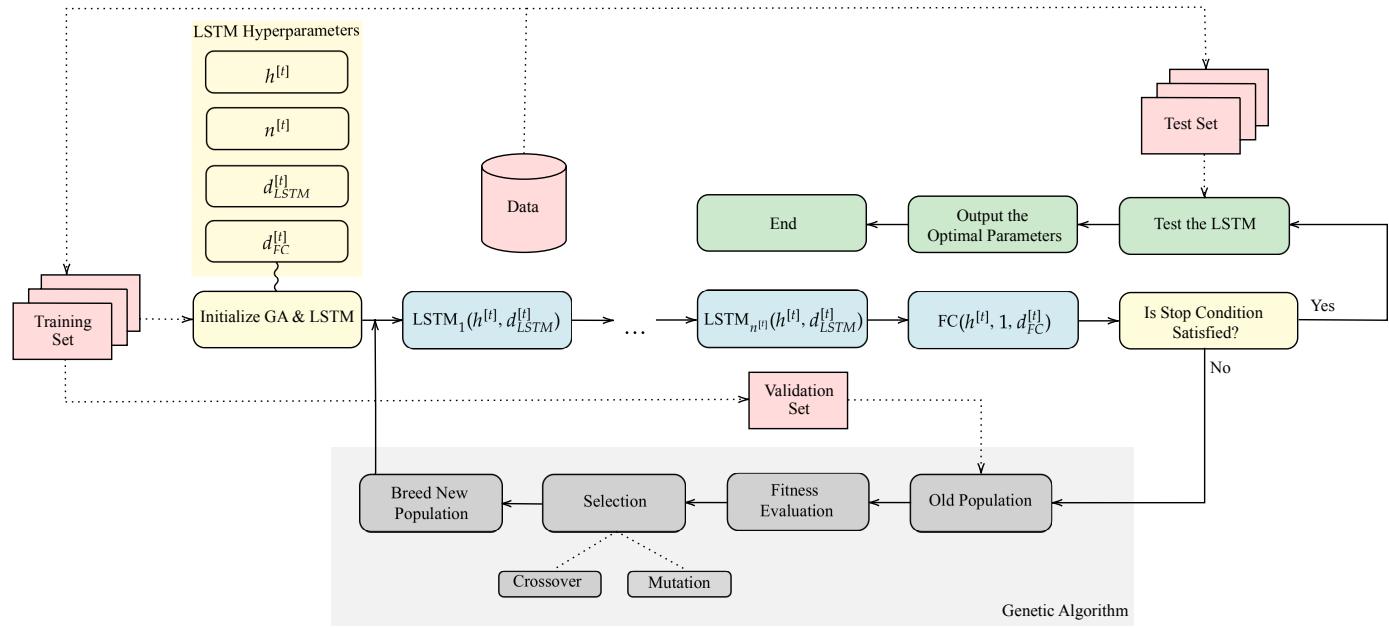


- Input gate:  $\Gamma_i = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$
- Forget gate:  $\Gamma_f = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$
- Cell gate:  $\Gamma_c = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$

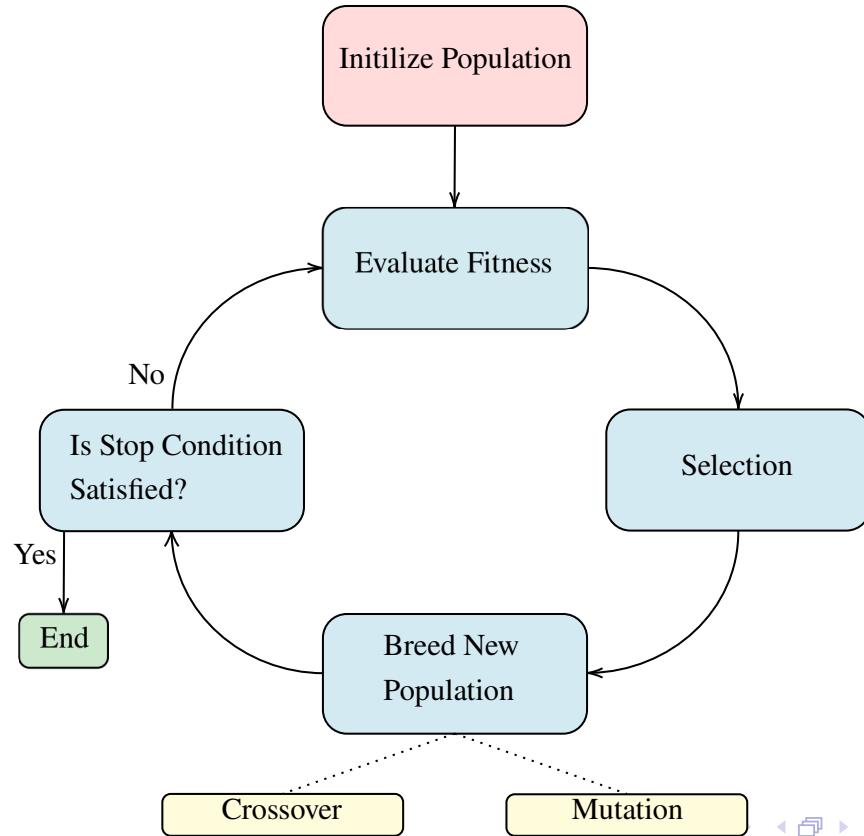
## Background Models

ARIMA  
FNN  
LSTM  
GA-LSTM  
PSO-LSTM

# GA-LSTM



# Genetic Algorithm



## Genetic Algorithm (GA)

- ]} нахождение Р с n особями:

$$\beta_t = \{x_i^{(t)} : (x_{i1}^{(t)}, \dots, x_{id}^{(t)}) \in \mathbb{R}^{d_g}\}_{i=1}^n$$

хромосома      ген

↑ особь    ≈ векторное представление особи

- Алгоритм:

1) Инициализирует первое поколение особей случайных образов:

$$\beta_0 = \{(\mathbf{x}_1^{(0)}, \dots, \mathbf{x}_n^{(0)}) \in \mathbb{R}^{n \times d} : \mathbf{x}_{ij} \sim U(\alpha, \beta)\}, \text{ где}$$

]  $\mathbf{x}_{ij}$  - dropout  $\Rightarrow \alpha=0, \beta=1$

]  $\mathbf{x}_{ij}$  - hidden size  $\vee$  num\_layers  $\Rightarrow \alpha=1, \beta=10$

2) Оценивается приспособленность (fitness) текущ. поколения:

$\Rightarrow$  Времянее знако. членов гр. (RMSE / SMAPE) для каждой хромосомы

$$f_t(\beta_t) = \{Q(x_i^{(t)})\}_{i=1}^n$$

$\Rightarrow$  Члены знако. предваряются σ так, чтобы все члены f\_t лежали на отрезке [0, 1]

$$\sigma(z) = \frac{z - f_t^{\text{worst}}}{f_t^{\text{best}} - f_t^{\text{worst}}} \quad \text{← это есть Softmax}$$

$\Rightarrow$  Получим:

$$\Sigma_t = \{\sigma(Q(x_i^{(t)}))\}_{i=1}^n \quad (0.1, 0.3, \dots, 0.7, 0.2)$$

3) Отдаётся к наилучшему приспособлению и дальнейшего размножения.

$$\text{] Особь } z \text{ более приспособлена чем } z_j \Leftrightarrow \sum_{t=m}^t \sigma(Q(x_m^{(t)})) > \sum_{t=j}^t \sigma(Q(x_j^{(t)}))$$

Размножение особей в текущ. поколении по знако. залогам их приспособленности

с порогом убывания:

$$\sum^{(1)} > \dots > \sum^{(k)} > \dots > \sum^{(n)}$$

4) Сортируются к конц. приносящими особей со всеми остаточными от особей

Так как разбросанные хромосомы не попадают.

При сортировании будем иметь в виду операцию создания новых хромосом, у кот. гены генов будут от принос. особей.

$$x_{i;j}^{(t)} = \begin{cases} x_{m;j}^{(t)} \in \text{вер. } P(x_{i;j}^{(t)} = x_{m;j}^{(t)}) \\ x_{v;j}^{(t)} \in \text{вер. } P(x_{i;j}^{(t)} = x_{v;j}^{(t)}) \end{cases}, \text{ где}$$

$$P(x_{i;j}^{(t)} = x_{m;j}^{(t)}) = \frac{\sum_{t_m}}{\sum_{t_m} + \sum_{t_v}}$$

$$P(x_{i;j}^{(t)} = x_{v;j}^{(t)}) = \frac{\sum_{t_v}}{\sum_{t_m} + \sum_{t_v}}$$

Генет. алгоритм приносящих генов (выделяют сильнейшие)

5) Всех особей в текущ. популяции, кроме лучших подвергают мутации.

Оператор мутации меняет хромосом. кот-го генов в хромосоме особи на другие (однако, добавлено движение к исходным).

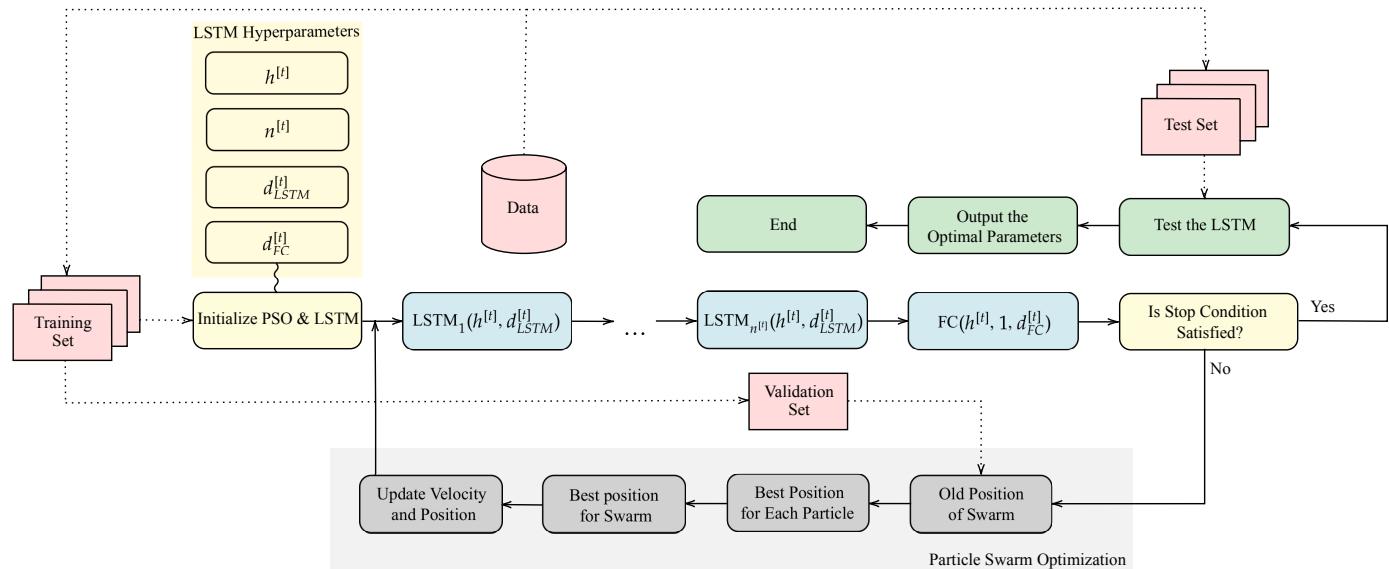
$$P_t^{\text{mutated}} = \{x_i^{(t)}\} \cup \{x_i^{(t)} : x_{i;j}^{(t)} = \Psi(x_{i;j}^{(t)})\}, \text{ где}$$

$$\Psi(x_{i;j}^{(t)}) = \begin{cases} x_{i;j}^{(t)} + \epsilon_j, \text{ если } w_{i;j} - \text{dropout} \\ x_{i;j} + U(0,2), \text{ если } w_{i;j} - \text{num\_layers} \vee \text{hidden\_size} \end{cases}$$

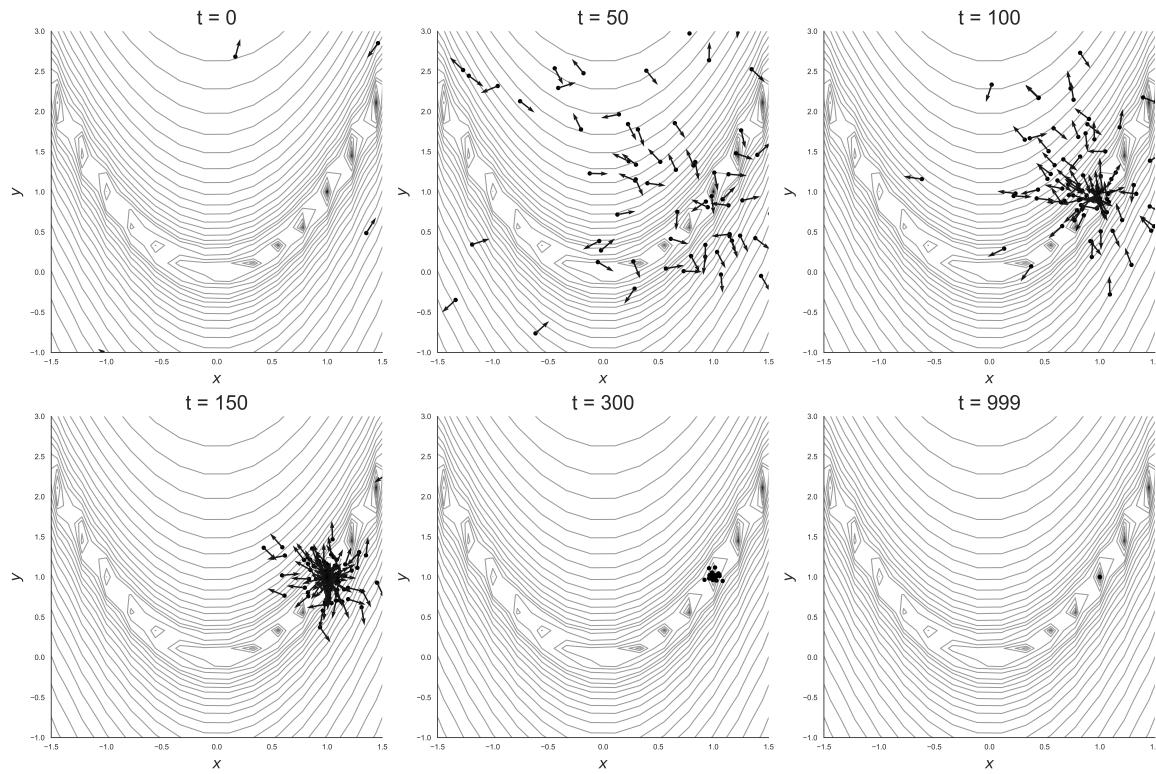
(дополнительное равномерное)

6) Переходим к шагу ②, пока не выполнено критерий останова

# PSO-LSTM



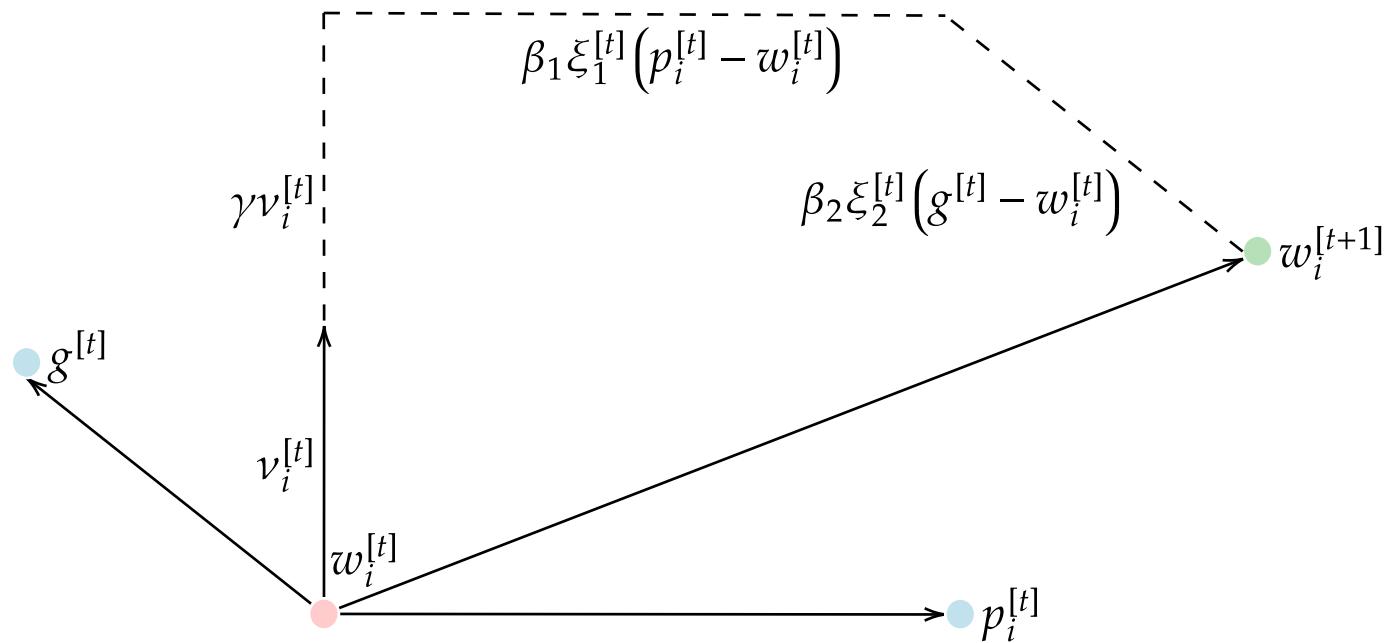
# Particle swarm optimization



## Particle Swarm Optimization

- ІІІ) Ініціалізація:  $W_t = \{w_i^{t+1} : (w_{i1}^{t+1}, \dots, w_{id}^{t+1}) \in \mathbb{R}^{d \times n}\}_{i=1}^n$   
Вектор кофеси та:  $V_t = \{v_i^{t+1} : (v_{i1}^{t+1}, \dots, v_{id}^{t+1}) \in \mathbb{R}^{d \times n}\}_{i=1}^n$
- Алогоритм:
  - 1) Рандомно вибираємо  $X_0$  та  $V_0$
  - 2) Створюємо вектори позицій для кожного гравця та:  
 $P_t = \{p_i^{t+1}\}_{i=1}^n$ , де позиція вибирається зі залежності:  
 $p_i^{t+1} = \begin{cases} w_i^{t+1}, & Q(w_i^{t+1}) > Q(w_i^{t+1}) \\ w_i^{t+1}, & Q(w_i^{t+1}) < Q(w_i^{t+1}) \end{cases}$
  - 3) Вибираємо позицію мінімуму функції відповідно до:  
 $g^{t+1} = \underset{i \in \{1, \dots, n\}}{\operatorname{argmin}} Q(p_i^{t+1})$
  - 4) Оновлюємо кофеси інших гравців:  
 $v_i^{t+1} = \gamma v_i^{t+1} + \beta_1 \zeta_1 (p_i^{t+1} - w_i^{t+1}) + \beta_2 \zeta_2 (g^{t+1} - w_i^{t+1}), \quad i = \overline{1, n}, \quad \text{де}$   
 $\gamma, \beta_1, \beta_2 \in \mathbb{R}$  - конст.,  $\zeta_1 \sim U(0, 1)$
  - 5) Отримуємо нове розподілення кофес:
- $w_{t+1} = w_t + V_{t+1}$  ← На цю стадію чаргування, якщо не діємо.  
Все діємо
- 6) Переходимо в шаг 2, поки критерій останкова не виконано

# Particle swarm optimization



# Thank you!