Online Machine Learning

Vladyslav Kolbasin
data scientist at GridDynamics,
lecturer at NTU KhPI, dep. KMMM

Definition

- **Online machine learning** (OML) is a method of machine learning in which:
  - data becomes available in a **sequential order**
  - is used to **update** our best **predictor** for future data **at each step**,
    as **opposed to batch learning** techniques which generate the best predictor by learning on the entire training data set at once

Difference to standard approach

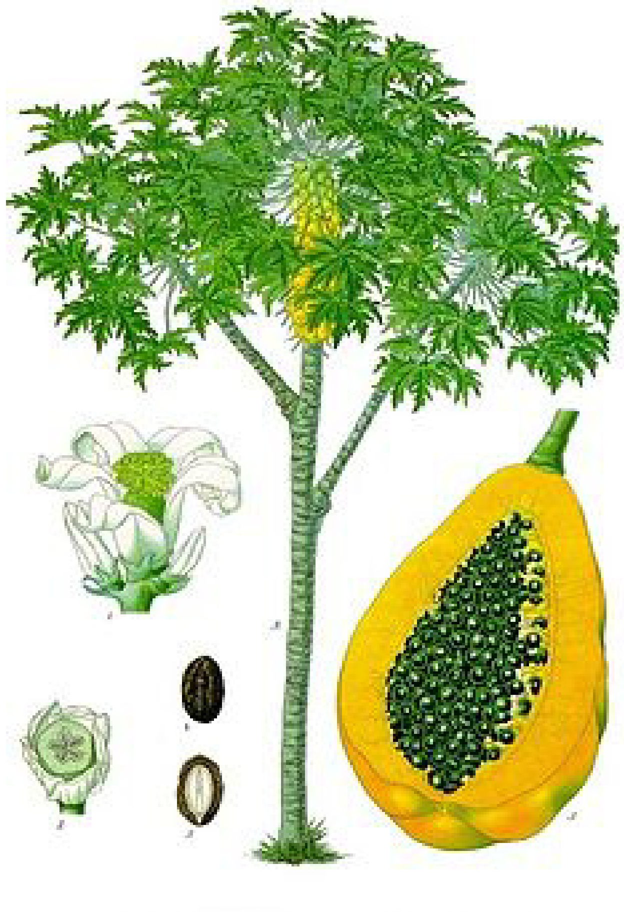
- Online prediction refers to the problem of prediction in the online protocol (sequential prediction problems):
  - Nature outputs some side information
  - Predictor outputs a prediction
  - Nature outputs an observation
  - The cycle is repeated
- Difference to usual supervised learning:
  - Test and Train datasets are the same but the distinguish between train and test is through time

Motivation example

- A problem:
  - What is papaya?
  - How to find good papaya?

Motivation example

- A problem:
  - What is papaya?
  - How to find good papaya?
  - Buy papaya and let's try to predict if it is ok.



Motivation

- It is computationally infeasible to train over the entire dataset
  - So should train by **mini-batches** (or one-by-one)
- It is used in situations where it is necessary for the algorithm to **dynamically adapt to new patterns** in the data
  - Data changes too fast – decrease lag
  - Fast tuning to important data trends

How to solve it?

- Recursive adaptive algorithms (Robbins and Monro - 1951)
- Stochastic approximation (Kushner and Clark, 1978)
- Adaptive filtering (Haykin 2002, 2010)

Model types

- Statistical models
  - Data samples are usually assumed to be i.i.d.
  - Algorithm just has a limited access to the data
- Adversarial models
  - They are looking at the learning problem as a game between two players (the learner vs the data generator)
  - The goal is to minimize losses regardless of the move played by the other player

I. Statistical online models

- Gradient descent
- Kalman filtering
- Kernel model
- SVM
- Folding-in

1. Batch gradient descent

$$w_{t+1} = w_t - \gamma_t \nabla_w \hat{J}_L(w_t) = w_t - \gamma_t \frac{1}{L} \sum_{i=1}^{L} \nabla_w Q(x_i, w_t)$$

- When the learning rate γ_t are small enough, the algorithm converges towards a local minimum of the empirical risk $\hat{J}_L(w)$
- We can speed up convergence by replacing γ_t by positive matrix
- We should **save all points** from training dataset
- Compute **average gradient** for **all points**

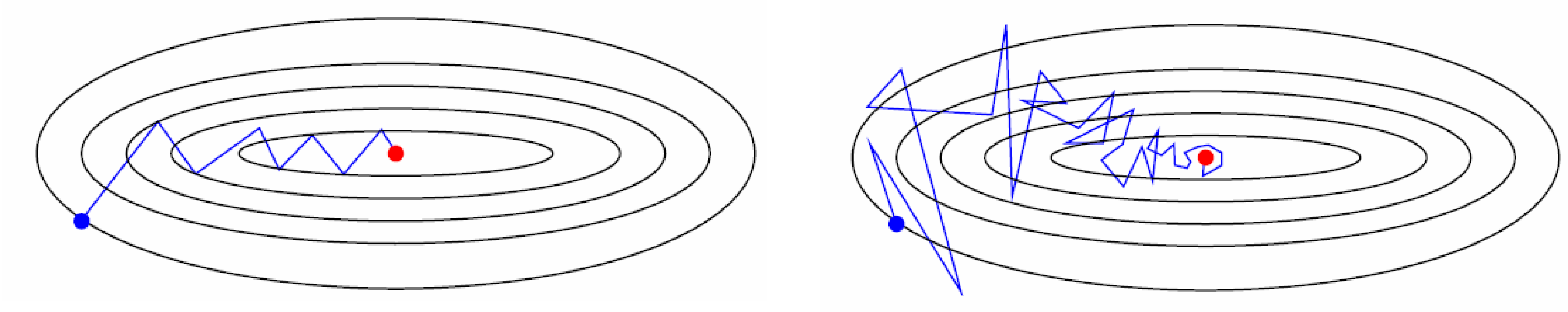
Online gradient descent

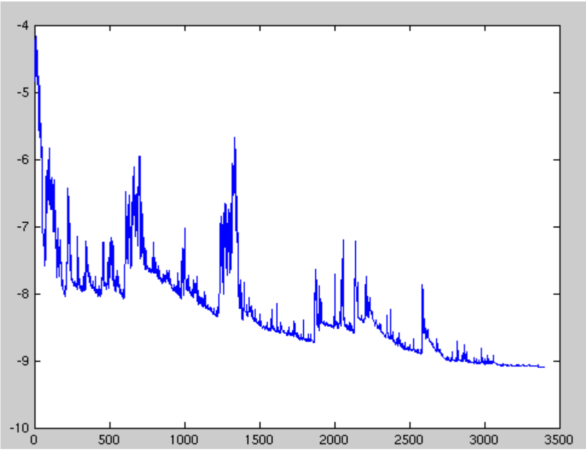
- We don't do averaging

$$w_{t+1} = w_t - \gamma_t \nabla_w Q(x_t, w_t)$$

- Use just one point one at a time
- We **hope** that random selection will not perturbate the average behavior of the algorithm

Online gradient descent





- So we see weird behavior
- Is there a convergence?

Online gradient descent

- The main question: is there a convergence?
- Theory: Yes!
  - When the learning rate decreases with an **appropriate rate**
  - We will get global minimum if **objective function is convex**, otherwise almost surely will get local minimum

Gradient descent optimizations

- There is a couple of GD optimizations:
  - Momentum (Sutton, R. S. 1986)
  - Nesterov accelerated gradient (Nesterov, Y. 1983)
  - AdaGrad (Duchi, J., Hazan, E., & Singer, Y. 2011)
  - Adadelta (extension of AdaGrad)
  - Stochastic average gradient
    (Le Roux, Schmidt, and Bach, 2012)

AdaGrad

- SGD with per-parameter learning rate
  - Large for more sparse parameters

$$g_t = \nabla_w Q(x_t, w_t)$$

$$G_{t,jj} = \sum_{k=1}^{t} g_{k,j}^2$$

$$w_{t+1} = w_t - \gamma \cdot diag(G_t + e)^{-1/2} \circ g_j$$

- Useful for sparse applications
  (for example NLP and image recognition)
  - Used in Google
  - Used for Glove model

2. Kalman filtering

- Recursive least squares filter
- Quasi-Newton algorithm

$$K_t = H_{t-1}^{-1}$$

$$K_{t+1} = K_t - \frac{(K_t x_t)(K_t x_t)^T}{1 + x_t^T K_t x_t}$$

$$w_{t+1} = w_t - K_{t+1}(y_t - w_t^T x_t)^T x_t$$

3. Online Kernel models

- Batch regime:

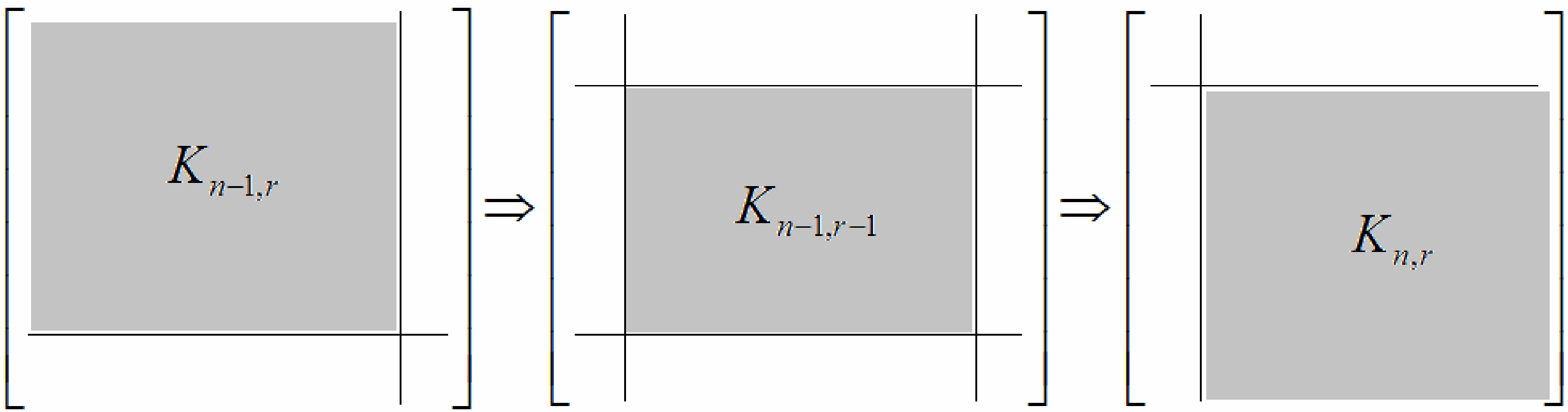
$$\alpha = K(\lambda)^{-1} Y = \left(K + \lambda I\right)^{-1} Y$$

$$K_{i,j} = \kappa(x_i, x_j), \quad i, j = \overline{1, r}$$

Online Kernel models

- The main problem: how to update inverse matrix
- It is possible to do in 2 stages:

$$\mathrm{K}_{n-1,\ r}^{-1}(\lambda) \rightarrow \mathrm{K}_{n-1,\ r-1}^{-1}(\lambda) \rightarrow \mathrm{K}_{n,\ r}^{-1}(\lambda)$$



Online Kernel models

$$\alpha_n = \left(K_{n-1,r}(\lambda)\right)^{-1}\left(\lambda^{-1}\alpha_{n-1} + Y_{n-1,r}\right)$$

$$K^{-1}_{n-1,r-1} = \mathrm{R}_r K^{-1}_{n-1,r} \mathrm{R}_r^{\mathrm{T}} -$$

$$-(\mathrm{e}_1^{\mathrm{T}} K^{-1}_{n-1,r} \mathrm{e}_1)^{-1} \mathrm{R}_r K^{-1}_{n-1,r} \mathrm{e}_1 \mathrm{e}_1^{\mathrm{T}} K^{-1}_{n-1,r} \mathrm{R}_r^{\mathrm{T}}$$

$$K^{-1}_{n,r}(\lambda) = \begin{pmatrix} A & B \\ C & \delta_n^{-1} \end{pmatrix}$$

$$A = K^{-1}_{n-1,r-1} + \delta_n^{-1} \cdot K^{-1}_{n-1,r-1} \cdot \mathrm{k}_{n-1,r-1}(x_n) \cdot$$

$$\cdot \mathrm{k}^{\mathrm{T}}_{n-1,r-1}(x_n) \cdot K^{-1}_{n-1,r-1}$$

Online Kernel models

$$B = -\delta_n^{-1} K_{n-1,r-1}^{-1} \cdot \mathrm{k}_{n-1,r-1}(x_n)$$

$$C = B^{\mathrm{T}} = -\delta_n^{-1} \mathrm{k}_{n-1,r-1}^{\mathrm{T}}(x_n) \cdot K_{n-1,r-1}^{-1}$$

$$\delta_n = \lambda^{-1} + \mathrm{k}_{n,n} - \mathrm{k}_{\mathrm{n\text{-}1,r\text{-}1}}^{\mathrm{T}}(x_n) \cdot \mathrm{K}_{n-1,r-1}^{-1} \cdot \mathrm{k}_{n-1,r-1}(x_n)$$

$$\mathrm{R}_r = \left(0_r \vdots \mathrm{I}_{r-1}\right) \qquad \mathrm{e}_1 = \begin{pmatrix}1 & 0 & \ldots & 0\end{pmatrix}^{\mathrm{T}}$$

$$\mathrm{k}_{n-1,r-1}^{\mathrm{T}}(x_n) = (\kappa(x_n, x_{n-r+1}) \ldots \kappa(x_n, x_{n-1}))$$

$$\mathrm{k}_{n,n} = \kappa(x_n, x_n)$$

Online Kernel models

- The issue is a complexity

- Accumulate new observations in Gram matrix
- Control complexity of Gram matrix:
  - Sparsification
  - Prunning

Online Kernel models

- Approximate Linear Dependency (Engel 2004)

$$\delta_t = \min_a \left\| \sum_{j=1}^{m_{t-1}} a_j \phi(\tilde{x}_j) - \phi(x_t) \right\|^2 \leq v$$

$$\delta_t = \min_a \left\{ a^T \tilde{K}_{t-1} a - 2a^T \tilde{k}_{t-1}(x_t) + k_{tt} \right\}$$

- Novelty criterion (Haykin 2010)

Online Kernel models

- More about kernels and recursive models: “Kernel Adaptive Filtering A Comprehensive Introduction (Adaptive and Learning Systems for Signal Processing, Communications and Control Series)” by Haykin

4. Online SVMs

- The most famous is LASVM algorithms (Léon Bottou 2005-2011)
- There is a package for R: https://cran.r-project.org/web/packages/lasvmR/index.html
- Uses 2 steps: PROCESS & REPROCESS
- In general add and delete support vectors

5. Folding-in

- SVD decomposition for recommender systems

$$SVD(A) = U \times S \times V^T$$

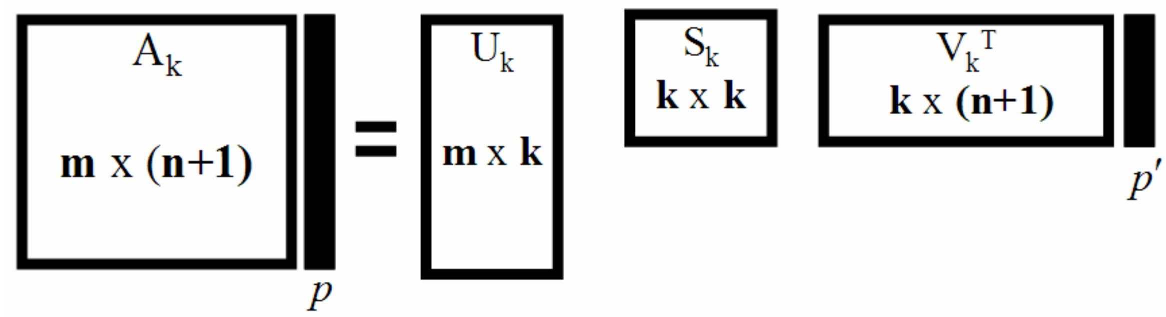
$$A \approx U_k \times S_k \times V_k^T$$

$$P_{i,j} = \bar{r}_i + \left(U_k \sqrt{S_k}^T (i)\right) \cdot \left(\sqrt{S_k}^T V_k (j)\right)$$

- Challenge: building SVD is time consuming
- How to add new product, new customer?

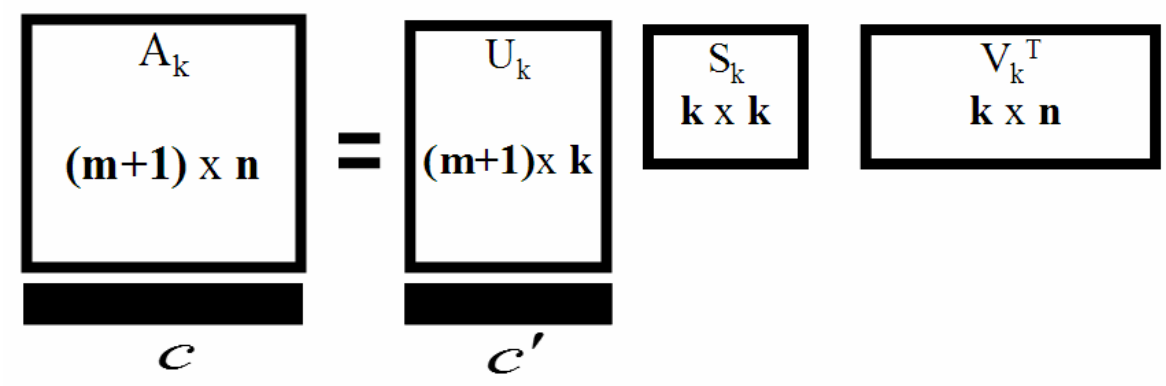
Incremental Singular Value Decomposition

1) New product p (mx1)



$$p' = p^T U_k S_k^{-1}$$

2) New customer c (1xn)



$$c' = c V_k S_k^{-1}$$

II. Adversarial models

- Definition:
  - Player chooses w_t
  - Adversary chooses $l_t(w)$
  - Player suffers loss $l_t(w_t)$
  - Need to minimize cumulative loss

- Some standard algorithms:
  - Follow the leader (FTL)
  - Follow the regularised leader (FTRL)

Adversarial models

- We will not look at this models, but they have several advantages:
  - In contrast to statistical machine learning (stochastic), adversarial algorithms don't make stochastic assumptions about the data they observe, and even handle situations where the data is generated by a malicious adversary
  - So no i.i.d. assumption!

Pros and Cons of Online ML algorithms

- Online algorithms are
  - Often much faster
  - More memory-efficient
  - Can adapt to the best predictor changing over time
  - Hard to maintain in production
  - Hard to evaluate
  - Have problems with convergence

Applications

- Online Machine Learning can be used for:
  - Computer vision
  - Recommender systems
  - Predicting stock market trends
  - Deciding which ads to present on a web page
  - IoT applications
- Put here your application…

Useful Links

- http://sebastianruder.com/optimizing-gradient-descent/index.html
- Kernel Adaptive Filtering A Comprehensive Introduction (Haykin 2010)
- The Kernel Recursive Least Squares Algorithm (Engel 2003)
- Foundations of Machine Learning (M. Mohri, A. Rostamizadeh, and A. Talwalkar 2012)

Thank you!

Questions?

ka_vlad@list.ru