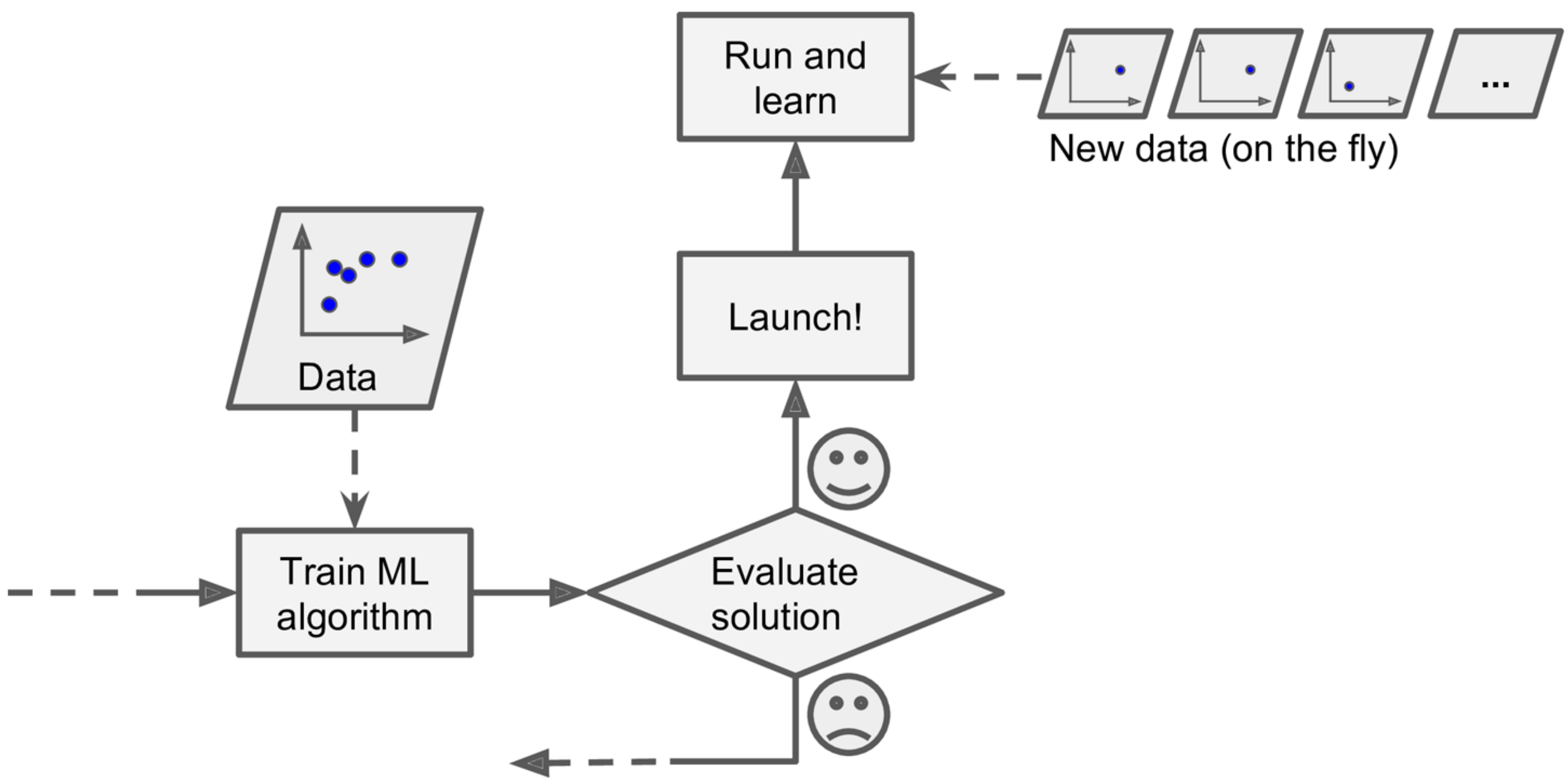
CSI 436/536
Introduction to Machine Learning

Online learning and Recursive LLSE

Professor Siwei Lyu
Computer Science
University at Albany, State University of New York

Learning paradigms

- Batch vs. online learning
  - batch learning: training data given at a batch
  - online learning: training data given continuously



Recursive LLSE

- So far we have assumed we get all the data (x_i, y_i) at the same time — this is known as the **batch learning**
- Many practical cases (e.g., predicting stock prices, user preferences, click through data, etc) we are not able to access all the training data because
  - The total dataset is too large to load into the memory at the same time
  - The data points are coming in a streaming manner, and we cannot have the whole dataset
- We need to consider a faster algorithm
  - This is known as the recursive LLSE algorithm

Matrix inversion lemma

- Woodsbury identity: when A and D are invertible
  $$(A + BDC^T)^{-1} = A^{-1} - A^{-1}C(D^{-1} + CA^{-1}B^T)^{-1}B^TA^{-1}$$
  - Proof: multiply the matrix on both sides
- important special case
  - B=C=z, a vector, D=I
    $$(A + zz^T)^{-1} = A^{-1} - (A^{-1}zz^TA^{-1})/(1 + z^TA^{-1}z)$$
  - B=-C=z, a vector, D=I
    $$(A - zz^T)^{-1} = A^{-1} + (A^{-1}zz^TA^{-1})/(1 - z^TA^{-1}z)$$
  - caching A^{-1} and computing the inversion recursively, typical inversion will take $O(n^3)$, while this special case it is $O(n)$

Correlation matrix

- Data matrix

$$X = \begin{pmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_N \\ | & | & & | \end{pmatrix}$$

- Correlation matrix $XX^T = x_1x_1^T + x_2x_2^T + \cdots + x_Nx_N^T$
- Inverse of correlation matrix when adding x

$$(XX^T + xx^T)^{-1} = (XX^T)^{-1} - \frac{(XX^T)^{-1}xx^T(XX^T)^{-1}}{1 + x^T(XX^T)^{-1}x}$$

- Inverse of correlation matrix when removing x

$$(XX^T - xx^T)^{-1} = (XX^T)^{-1} + \frac{(XX^T)^{-1}xx^T(XX^T)^{-1}}{1 - x^T(XX^T)^{-1}x}$$

Review of over-complete LLSE

- LLSE objective $\min_w \|y - X^T w\|^2$
  - Solution (over-complete) $w = (XX^T)^{-1}Xy$
  - LS Error at optimum $y^T y - (Xy)^T(XX^T)^{-1}Xy$
    - To prove: the prediction $(I_N - X^T(XX^T)^{-1}X)y$ matrix $A = (I_N - X^T(XX^T)^{-1}X)y$ is idempotent, i.e., AA = A, so the LS error

$$y^T(I_N - X^T(XX^T)^{-1}X)^T(I_N - X^T(XX^T)^{-1}X)y$$

becomes $y^T y - (Xy)^T(XX^T)^{-1}Xy$

Recursive LLSE

- Solving the same problem $\min_p \|y - Xp\|^2$
- Batch LLSE focus on solving the normal equation

$$X^T X p = X^T y, p = (X^T X)^{-1} X^T y$$

- For the online setting, when a new datapoint (x_t^T, y_t) is received, the normal equation becomes

$$\begin{pmatrix} X \\ x_t^T \end{pmatrix}^T \begin{pmatrix} X \\ x_t^T \end{pmatrix} \hat{p} = \begin{pmatrix} X \\ x_t^T \end{pmatrix}^T \begin{pmatrix} y \\ y_t \end{pmatrix} \Rightarrow (XX^T + x_t x_t^T)\hat{p} = X^T y + y_t x_t$$

  - So we can get a solution to the updated parameter as $\hat{p} = (XX^T + x_t x_t^T)^{-1}(X^T y + y_t x_t)$
  - problem: we have to inverse a matrix every time

Recursive LLSE

- Recursive normal equation

$$\hat{w} = (XX^T + x_t x_t^T)^{-1}(X^T y + y_t x_t)$$

- Using matrix inversion lemma

$$(XX^T)^{-1}X^T y - \frac{(XX^T)^{-1}x_t x_t^T (XX^T)^{-1}X^T y}{1 + x_t^T (XX^T)^{-1}x_t} + y_t (XX^T)^{-1}x_t - \frac{y_t (XX^T)^{-1}x_t x_t^T (XX^T)^{-1}x_t}{1 + x_t^T (XX^T)^{-1}x_t}$$

- Simplifying the two terms we get

$$\hat{w} = \left(I - \frac{(XX^T)^{-1}x_t x_t^T}{1 + x_t^T (XX^T)^{-1}x_t} \right) w + \frac{y_t (XX^T)^{-1}x_t}{1 + x_t^T (XX^T)^{-1}x_t}$$

  or further combining terms $$\hat{w} = w + \frac{y_t - x_t^T p}{1 + x_t^T (XX^T)^{-1}x_t}(XX^T)^{-1}x_t$$

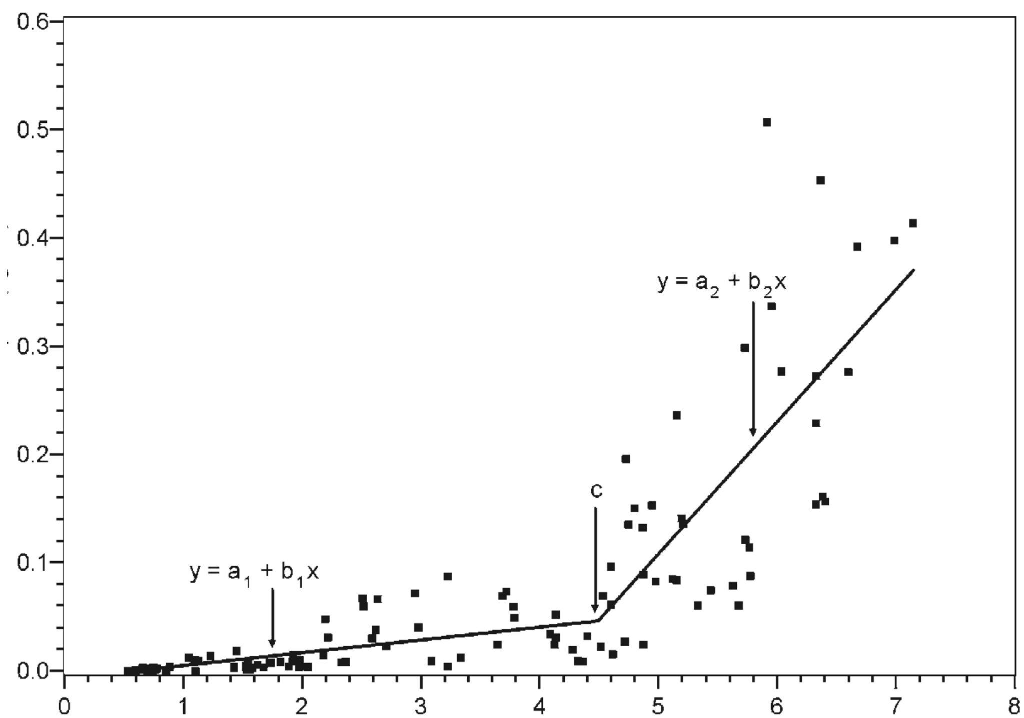
  - Term on top of the ratio is the prediction error for the newly come data point, if it is zero, no update
  - XX^T and w are precomputed

Recursive LLSE

- Loss function update
  - This will be useful for the derivation of the segmented LLSE in the following
  - Original LLSE loss $e = y^T y - (Xy)^T (XX^T)^{-1} Xy$
  - Recursive loss
    $\hat{e} = y^T y + y_t^2 - (Xy + y_t x_t)^T (XX^T + x_t x_t^T)^{-1} (Xy + y_t x_t)$
    use matrix inversion lemma to update without actually inverting the new correlation matrix

Multi-line LLSE

- Fitting multiple lines (or polynomials) to 1D data
- Assuming $x_1 < x_2 < \cdots < x_N$ as inputs and $y_1, y_2, \cdots, y_N$ as the corresponding targets
- Goal: fit multiple lines to fit the dataset
  - Each extra line introduce a penalty of p



Segmented LLSE

- Also known as the Bellman algorithm
- Widely used in geography, computer graphics, and image processing
- Basic idea:
  - Fit as many data point as possible using one line
  - Each time a new line is introduced add penalty
  - The algorithm is incremental using dynamic programming
  - Balance between data fitting and model complexity

Dynamic programming

- Define function OPT(i) as the cost at step i, as
  $$OPT(i) = \min_{j=1:i-1}\{e_{ji} + p + OPT(j-1)\}$$
  - $e_{j,i}$ is the LLSE error fitting a line for points (x_j,y_j), $(x_{j+1},y_{j+1}),\ldots,(x_i,y_i)$, which can be easily computed from $e_{j,i-1}$ using the recursive LLSE algorithm
  - p is the penalty of adding one extra line
  - OPT(j-1) is the cost of step j-1
  - $OPT(0) = 0$, and $e_{1,i} + p + OPT(0) = e_{1,i} + p$ corresponds to fitting all data using one line
- Interpretation: the minimal cost at step i is obtained by trying each previous optimal solution, and adding one extra line into the system

Algorithm

- $OPT(0) = 0$
- <u>For</u> $i = 1$ to N:
  - <u>For</u> $j = 1$ to $i-1$:
    - $e_{j,i} \leftarrow$LLSE error fitting a line for points (x_j,y_j), $(x_{j+1},y_{j+1}),\ldots,(x_i,y_i)$ from $e_{j,i-1}$ using the recursive LLSE algorithm
  - $OPT(i) = \min_{j=1:i-1}\{e_{ji} + p + OPT(j-1)\}$
  - Back-tracking the line segment parameters
- <u>Return</u> $OPT(N)$

Summary

- Difference between online and batch learning algorithms
- Scenarios when online learning is needed
  - Online learning is related with stochastic gradient descent method