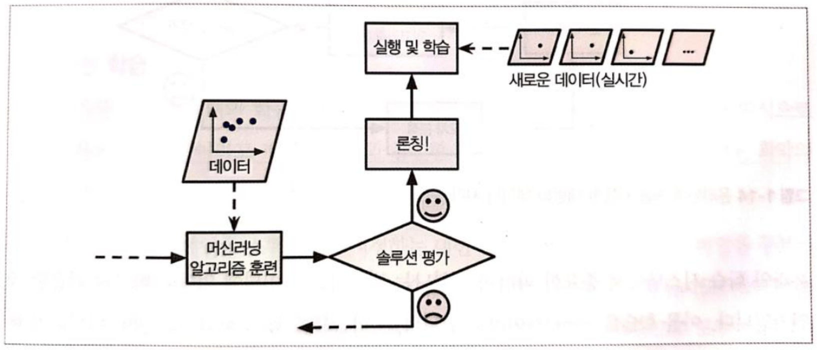
On-line Learning

What is on-line learning?

- On-line learning vs. batch learning
  - Data available in a sequential order
  - Not by learning on the entire training data set at once
- Learning method
  - Dynamically adapt to new patterns



출처: Aurelien Geron, “Hands-On Machine Learning with Scikit-Learn, Keras, and Ternsorflow

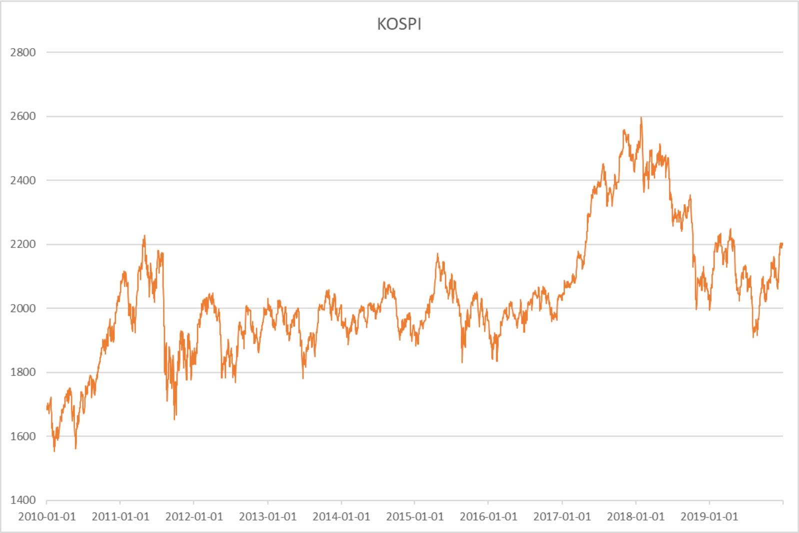
Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Introduction

- Time series prediction models require new learning as data changes over time.

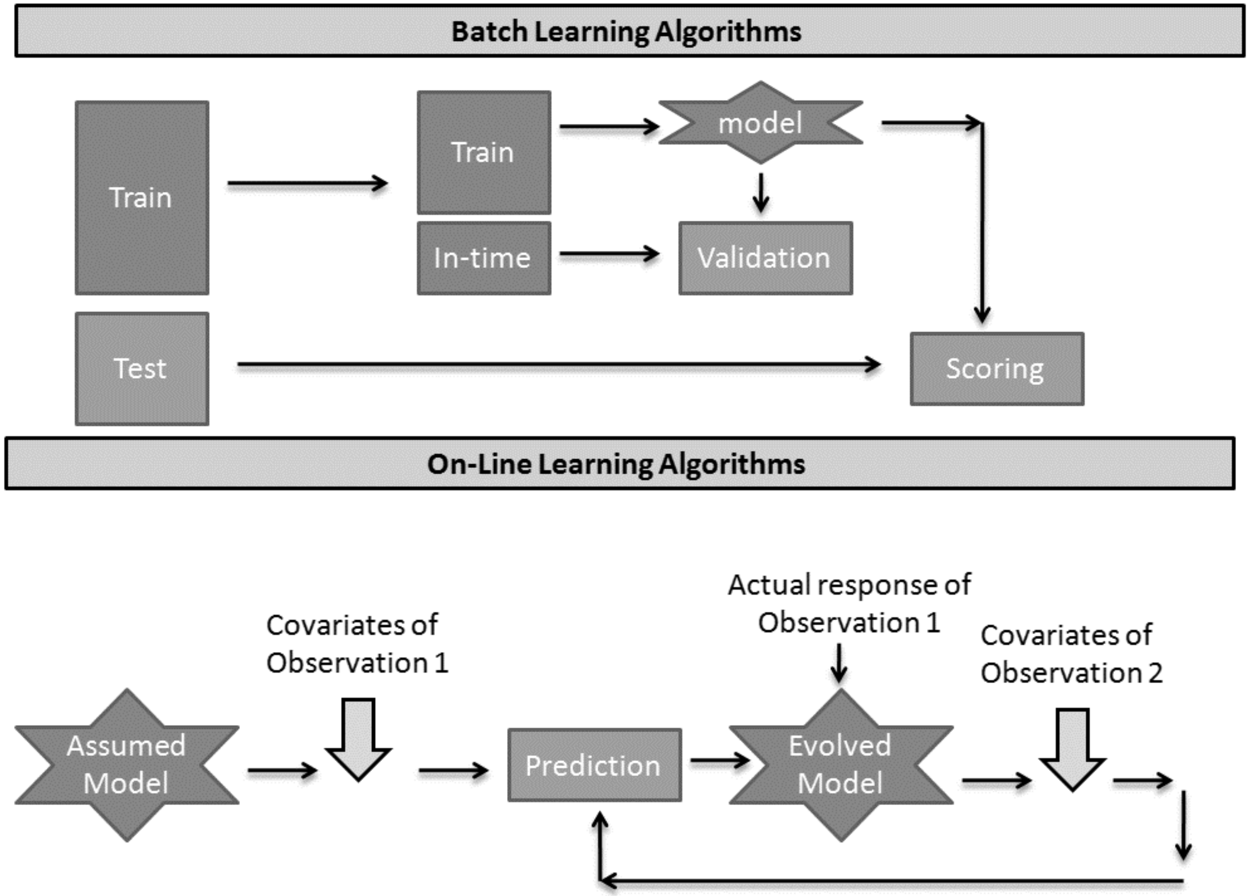
- For typical batch learning methods, learning is carried out using all the data so far for new learning.

- Batch learning methods require a long time and a large amount of storage space for learning, as they require all the data so far.



Introduction

- Batch learning vs Online learning



Logistic Regression example

```python
def calc_prob(b, x):
    dot = np.array(b).dot(np.array(x))
    if dot >= 7.6:
        prob = 1.0
    else:
        prob = np.exp(dot) / (1 + np.exp(dot))
    return prob

def calc_err(p, y):
    err = np.log2(np.power(p, y) * np.power((1-p),(1-y)))
    if err <= 1e-10:
        err = 1e+10
    return abs(err)

def relDiff(a, b):
    diff = abs(a - b) / (abs(a) + abs(b))
    return diff

def update(b, x, lr, y, p):
    for i in range(len(b)):
        b[i] +=  lr * x[i] * (y-p)
    return b

def OLR(x, y, threshold, epsilon, lr_i, delta, instances, B, W, fn, theta):
    instances.append((x, y)) # add x, y to the set of observed instances.
    H = sum(W) # w_i returns the weight value associated with parameter i
    rand_idx = int(np.random.choice(range(len(W)), 1, p = [i / H for i in W]))
    b = B[rand_idx].copy() # sample a random b_r, r is it's index in B
    loss_i = 0 # initial loss value
    loss_est = 1e+10 # infinite
    t = 0 # initial trial value
    prob = calc_prob(b, x) # calculate probability of potential

    if calc_err(prob, y) >= threshold: # The prediction makes mistake
        lr = lr_i / (1 + np.exp(1)/delta)
        b_est = b.copy() # initial a new parameter
        W[rand_idx] *= fn # update weight value of b_r
        err = 0
        if len(B) < theta:
            while relDiff(loss_est, loss_i) > epsilon:
                for (x_i, y_i) in instances:
                    p = calc_prob(b_est, x_i)
                    b_est = update(b, x_i, lr, y_i, p)
                    err += calc_err(p, y_i)
                loss_est = loss_i
                loss_i = -err
                t += 1
            B.append(b_est)
            W.append(1)

    return prob, B, W, instances
```

Algorithm 3 OLR$(x, y, \epsilon, \varepsilon, \eta_0, \delta, \Omega, \mathcal{B}, W, \omega, \theta)$

Input:
- x, y : Input vector x and its true label y.
- ϵ : Threshold value $\epsilon \in \mathbb{R}, \epsilon > 0$
- ε : Minimum relative error improvement $\varepsilon \in \mathbb{R}, \varepsilon > 0$
- η_0 : Initial learning rate $\eta_0 \in \mathbb{R}, \eta_0 > 0$
- δ : Annealing rate $\delta \in \mathbb{R}, \delta > 0$
- Ω : Set of observed instances
- $\mathcal{B}$: Set of parameter vectors, $\mathcal{B} = \{\beta_1, \beta_2, \dots, \beta_{|\mathcal{B}|}\}$
- W : Set of associated weight of $\mathcal{B}$, $W = \{w_1, w_2, \dots, w_{|\mathcal{B}|}\}$
- ω : A reduction function $\omega = e^{-\alpha}, \alpha > 0$
- θ : Maximum size of $\mathcal{B}$, $\theta \in \mathbb{R}, \theta > 0$

Output: $p, \mathcal{B}, W, \Omega$

$\Omega \longleftarrow (x, y)$ /* add x, y to the set of observed instances */
$H \longleftarrow \sum_{i=1}^{|\mathcal{B}|} w_i$ /* w_i returns the weight value associated with parameter i */
$\beta_r \longleftarrow \text{sampling}\left(\frac{w_1}{H}, \frac{w_2}{H}, \dots, \frac{w_{|\mathcal{B}|}}{H}\right)$ /* sample a random β_r, r is it's index in $\mathcal{B}$ */
$\ell \longleftarrow 0$ /* initiate loss value */
$\hat{\ell} \longleftarrow \infty$
$t \longleftarrow 0$ /* initiate trial value */
$p \longleftarrow \frac{e^{(\beta_r^T \cdot x_r)}}{1+e^{(\beta_r^T \cdot x_r)}}$ /* Calculate probability of potential */
IF $abs\left(\log_2\left(p^y(1-p)^{(1-y)}\right)\right) \geq \epsilon$ **THEN** /* The prediction makes mistake */
  $\eta_e \longleftarrow \frac{\eta_0}{1+e/\delta}$
  $\hat{\beta} \longleftarrow \beta_r$ /* initiate a new parameter */
  $w_r \longleftarrow w_r \times \omega$ /* update weight value of β_r */
  IF $|\mathcal{B}| < \theta$ **THEN**
    WHILE relDiff$(\hat{\ell}, \ell) > \varepsilon$ /* Define relDiff$(a, b) = \frac{abs(a-b)}{abs(a)+abs(b)}$ */
      FOR $i \longleftarrow 1$ **TO** $|\Omega|$ **DO**
        $p_i = \frac{e^{\hat{\beta} \cdot x_i}}{1+e^{(\hat{\beta} \cdot x_i)}}$
        $\hat{\beta} \longleftarrow \hat{\beta} + \eta_e x_i(\mathrm{I}(y_i = 1) - p_i)$
        /* The indicator function $\mathrm{I}(a = 1)$ returns 1 if $a = 1$, otherwise 0 */
      ENDFOR
      $\hat{\ell} \longleftarrow \ell$
      $\ell \longleftarrow -\sum_{i \leq |\Omega|} \log\left(p_i^{y_i}(1-p_i)^{(1-y_i)}\right)$
      $t \longleftarrow t + 1$
    ENDWHILE
    $\mathcal{B} \longleftarrow \hat{\beta}$
    $W \longleftarrow 1$
  ENDIF
ENDIF
RETURN $p, \mathcal{B}, W, \Omega$

Logistic Regression example

- Data – Telecom-CostomerChunk.csv (7032 rows, 16 columns)

```python
# experiments
x = data.iloc[:, :-1]
y = data.iloc[:, -1]
B = [np.zeros_like(x.iloc[0].to_numpy())]
W = [1]
lr_i = 0.01
instances = []
threshold = 2
epsilon = 0.1
delta = 2
fn = np.exp(-0.5)
theta = 800

probs = list()
for i in range(x.index.size):
    p, B, W, instances = OLR(x.iloc[i].to_numpy(), y.iloc[i], threshold, epsilon,
                             lr_i, delta, instances, B, W, fn, theta)
    probs.append(p)
```

0	0
0	0
0	-0.10253
0	0
0	0
0	0
0	-0.10253
0	0
0	0
0	0
0	0
0	0
0	0
0	-0.0118343
0	-0.000130736

parameter vector updated (1 step)

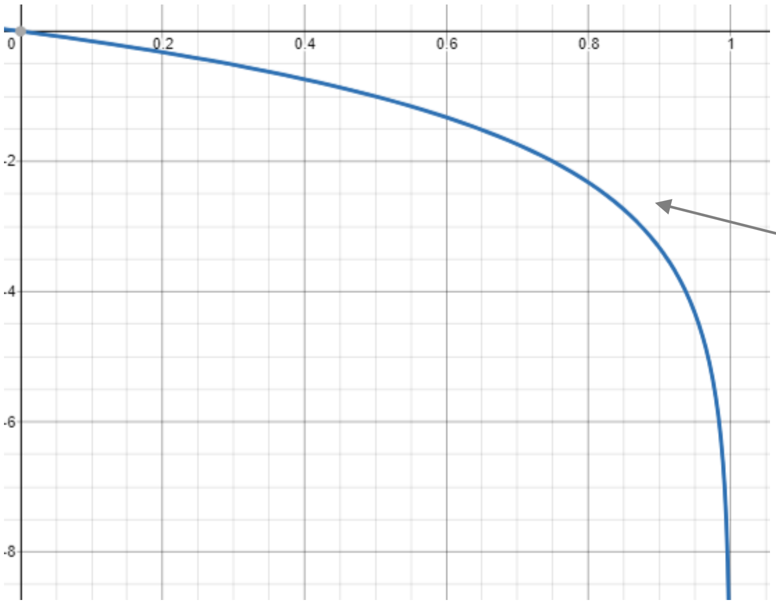
```
[0.6065306597126334, 1]
```

parameter weight vector updated (1 step)

Logistic Regression example

- Compare to TLR (Traditional Logistic Regression)
  - OLR acc : 0.7436
  - TLR acc : 0.7996

- The greater theta, the better the acc
  - theta 200 : 0.7463
  - theta 500 : 0.7571
  - theta 800 : 0.7651



$$abs(log_2(p^y(1-p)^{(1-y)}))$$

when y = 0, the err value according to p

threshold = 2 means when y = 0 and p >= 0.7
(y = 1 , p <= 0.25)

LSTM example

Algorithm OLL

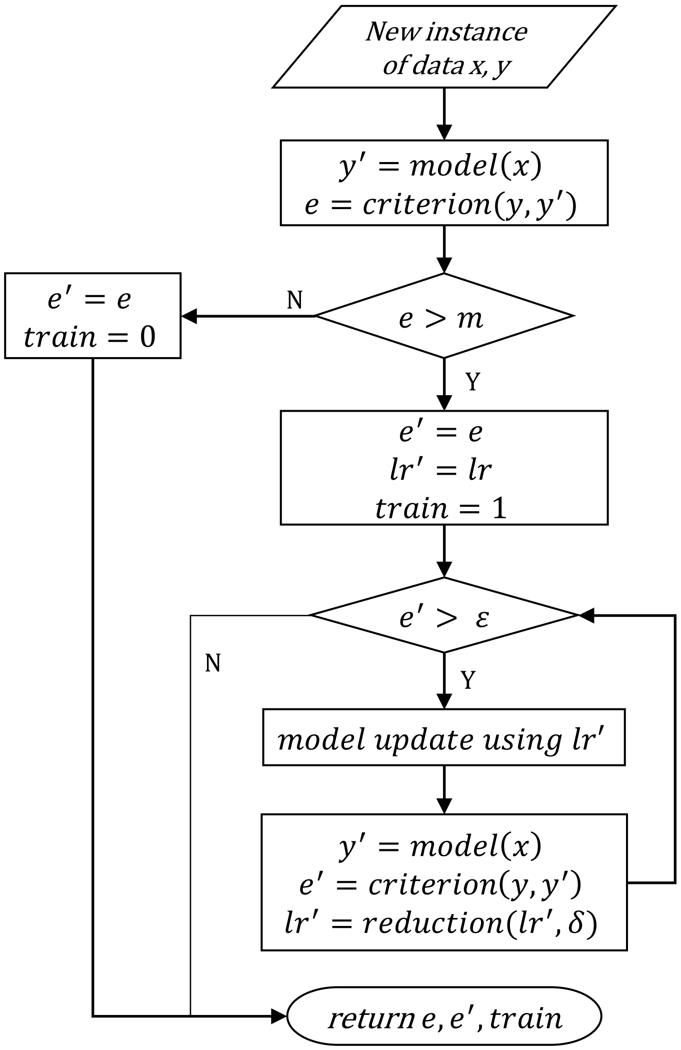
INPUT :

- x, y : Input vector x and true value y
- model : pretrained LSTM model
- m : margin of error
- η_0 : initial learning rate
- δ : learning rate reduction rate
- ε : minimum error improvement
- $R(x, d)$: learning rate reduction function

OUTPUT : *e, e', train*

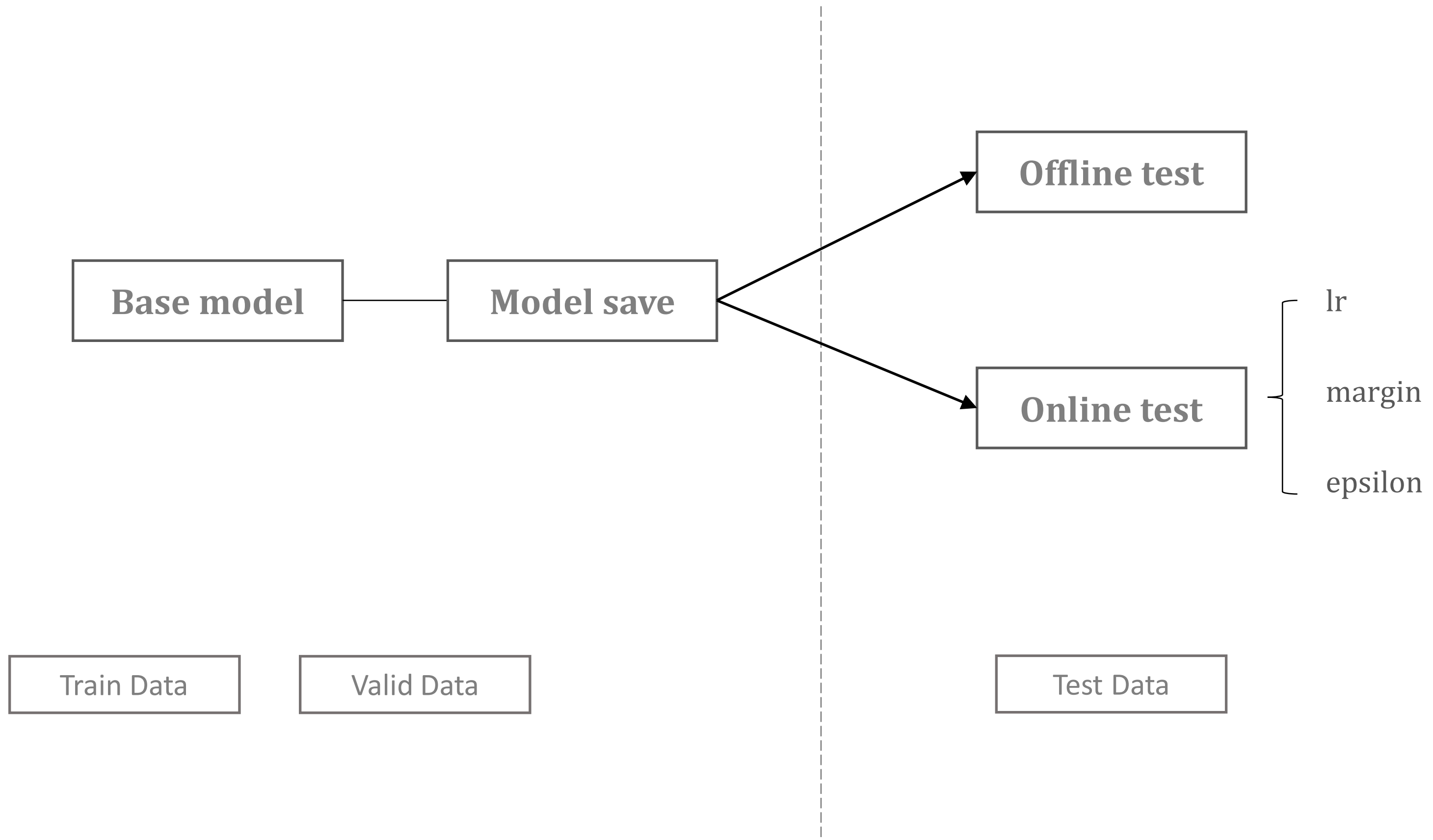
y' = model(x)
e = mae(y, y')
IF *e > m* **THEN**
    $e' = e$
    $\eta' = \eta$
    train = 1
    WHILE $e' > \varepsilon$
        model update using η'
        y' = model(x)
        e' = mae(y, y')
        $\eta' = R(\eta, \delta)$
ELSE
    $e' = e$
    train = 0

RETURN *e, e', train*



Experiment Result

- Experiment Overview



Proposed method

- Learning Rate Reduction

$$lr' : new\ learning\ rate, \qquad t : traing\ count,$$
$$\delta : \ learning\ rate\ reduction\ rate$$

$$lr' = lr\ - \delta t$$

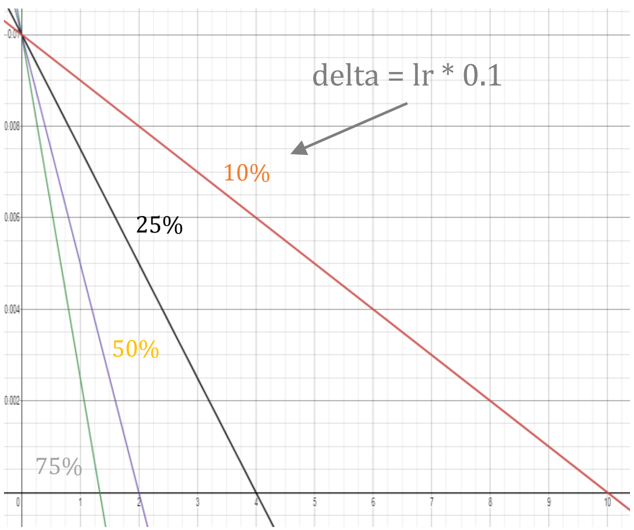
linear Reduction

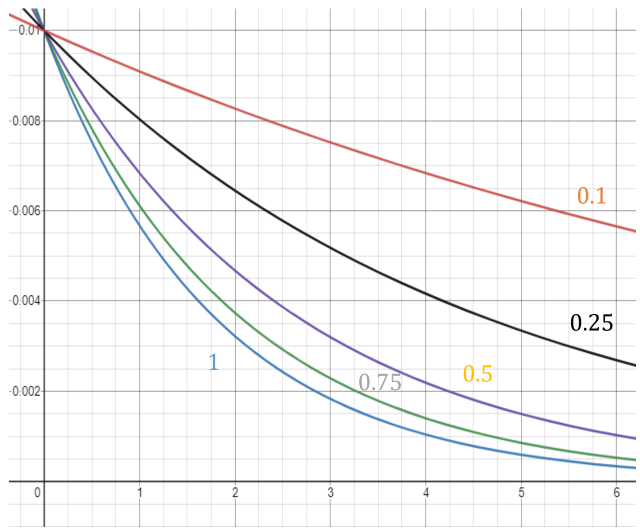
$$lr' = \frac{lr}{(1+\tanh(\delta))^{t}}$$

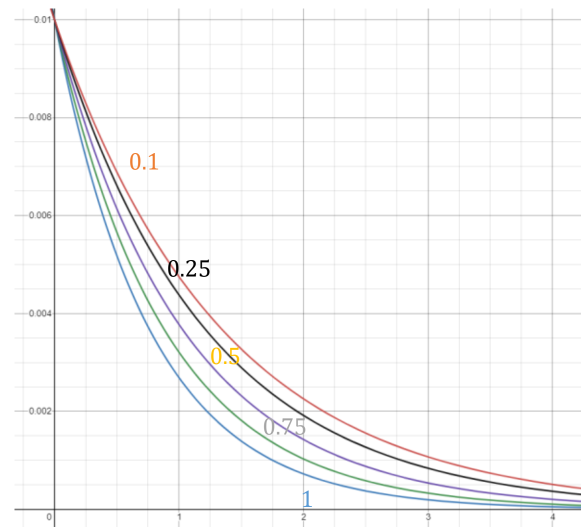
tanh Reduction

$$lr' = \frac{lr}{(1+\exp(\delta))^{t}}$$

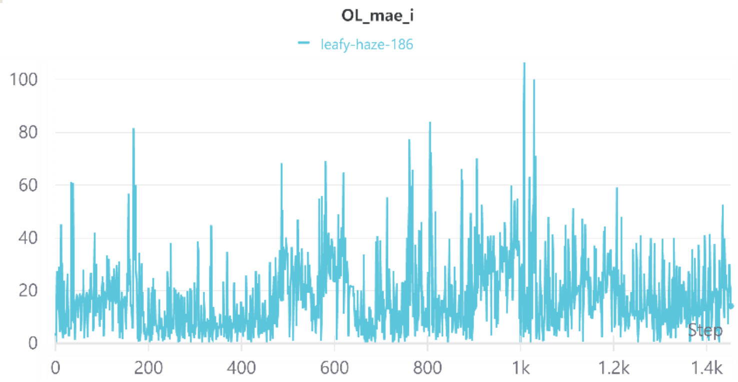
exp Reduction

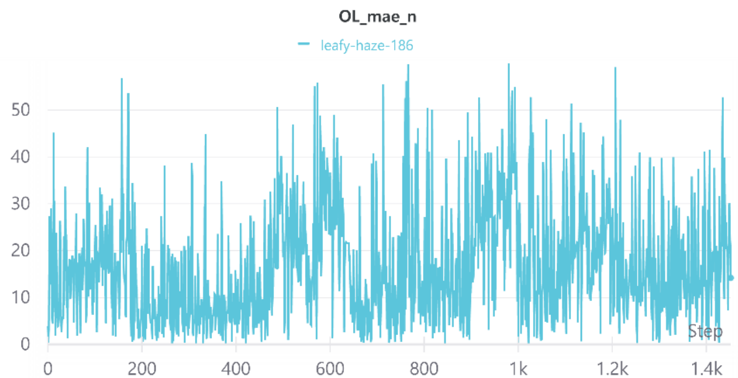


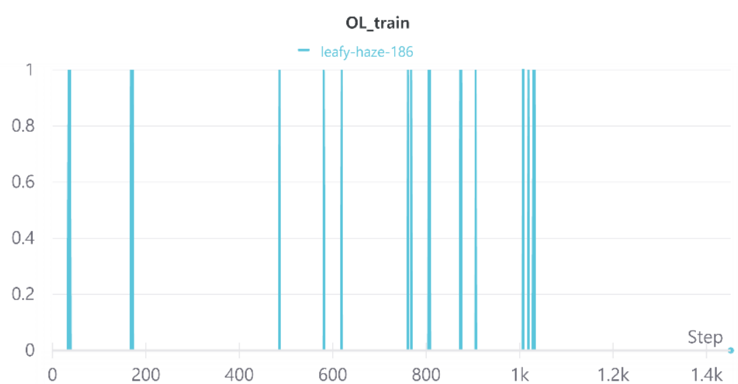




Proposed method







Evaluation Index

MAE_i : Mean error of values predicted by the initial model

MAE_n: Mean error of values predicted by the new(evolved) model

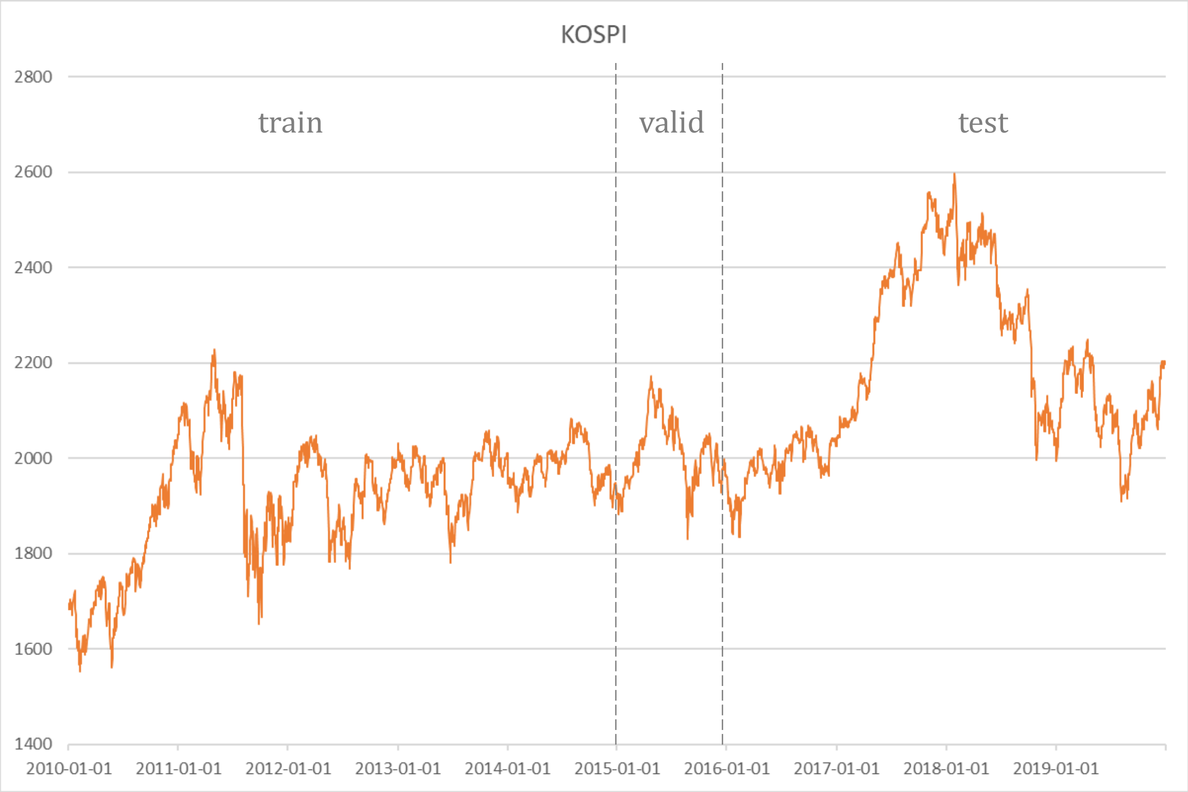
Train count : Number of times over margin

Experiment Result

- Experiment Data
  - KOSPI
  - CSI
  - Gold (London Gold Exchange Market price ($))
  - 10 years, 3652 days

	mean	std	min	max
KOSPI	2036.37	192.63	1552.79	2598.19
CSI	3140.42	629.56	2086.97	5353.75
Gold	20.79	13.67	6.17	55.77

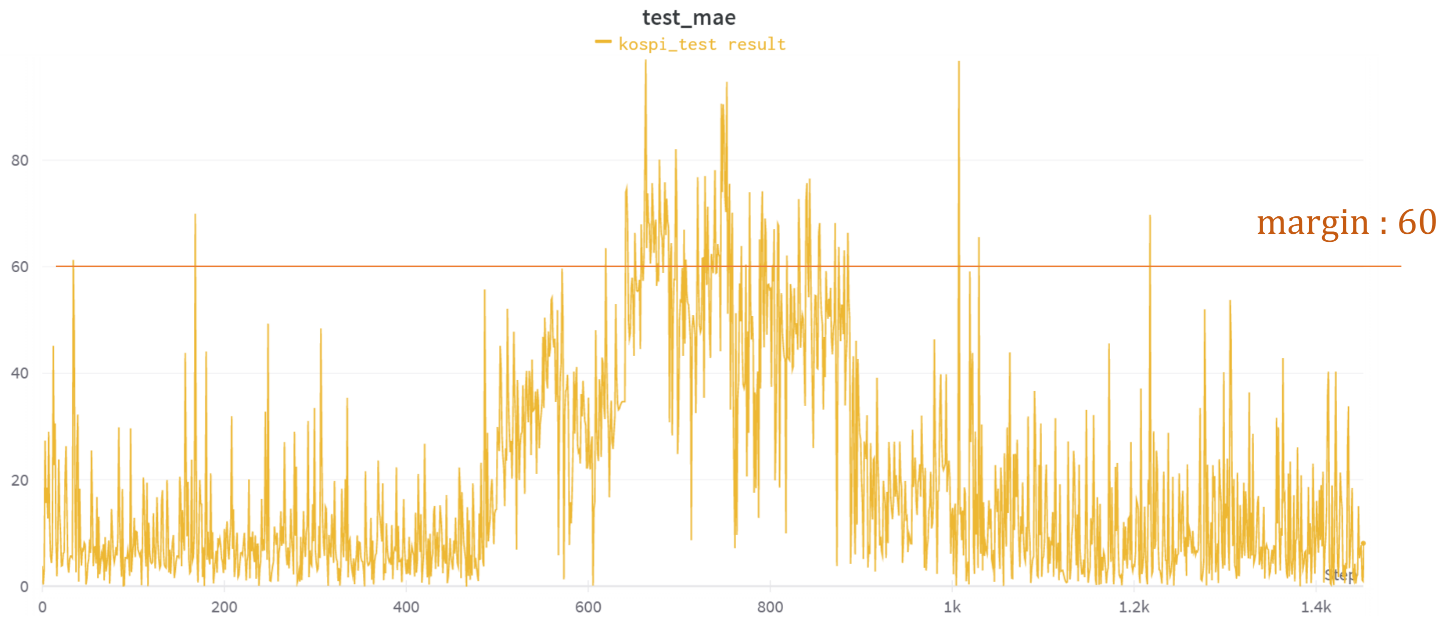
- Train : Valid : Test = 0.5 : 0.1 : 0.4 (1826, 365, 1461)
- window len : 7



Experiment Result

- Offline Learning Test Result

AVG_mae	Margin exceeded
20.388	92



Experiment Result

- Online Learning Test Result : no reduction – initial lr
  - Margin : 60
  - Epsilon : 25

		1e-2	2.5e-3	1e-3	2.5e-4	1e-4
KOSPI	MAE_i	93.072	62.595	29.702	19.943	22.072
	MAE_n	18.348	20.281	18.821	19.052	21.306
	Train	927	742	210	21	19

Experiment Result

- Online Learning Test Result : linear reduction – initial lr
  - Margin : 60
  - Epsilon : 25

delta = lr * 0.75

delta		75%	50%	25%	10%
1e-2	MAE_i	37.766	105.796	57.031	40.185
	MAE_n	22.557	18.881	20.747	22.806
	Train	155	796	307	163
2.5e-3	MAE_i	27.224	40.399	30.233	25.333
	MAE_n	21.040	22.442	20.507	22.353
	Train	89	234	117	65
1e-3	MAE_i	23.071	23.703	21.064	15.695
	MAE_n	20.825	19.712	19.590	15.058
	Train	47	75	28	15
2.5e-4	MAE_i	22.770	20.718	21.210	18.137
	MAE_n	21.810	19.979	20.276	17.100
	Train	23	18	22	26
1e-4	MAE_i	17.461	16.543	20.927	15.608
	MAE_n	16.739	15.888	20.336	15.054
	Train	19	16	17	16

Experiment Result

- Online Learning Test Result : tanh reduction – initial lr
  - Margin : 60
  - Epsilon : 25

delta		1	0.75	0.5	0.25	0.1
1e-2	MAE_i	23.434	27.492	32.419	39.192	33.171
	MAE_n	20.904	23.863	25.316	23.648	23.703
	Train	50	69	119	200	154
2.5e-3	MAE_i	21.685	20.452	23.153	23.904	21.917
	MAE_n	20.248	17.887	21.044	20.716	17.983
	Train	35	56	46	65	76
1e-3	MAE_i	20.342	20.066	20.843	18.479	20.490
	MAE_n	19.633	19.364	20.008	17.836	19.233
	Train	16	15	18	14	31
2.5e-4	MAE_i	16.269	17.967	19.096	20.064	22.930
	MAE_n	15.390	17.346	18.338	18.964	21.840
	Train	24	18	19	29	28
1e-4	MAE_i	19.604	20.838	14.568	17.080	15.961
	MAE_n	19.029	20.117	13.945	16.646	15.406
	Train	15	19	18	13	15

Experiment Result

- Online Learning Test Result : exp reduction – initial lr
  - Margin : 60
  - Epsilon : 25

delta		1	0.75	0.5	0.25	0.1
1e-2	MAE_i	24.907	24.841	21.888	22.055	23.441
	MAE_n	23.350	23.122	20.695	20.549	22.023
	Train	45	49	35	40	33
2.5e-3	MAE_i	16.951	18.563	18.532	22.209	17.973
	MAE_n	15.951	17.379	17.506	20.942	17.035
	Train	27	33	29	28	22
1e-3	MAE_i	23.257	18.714	17.358	17.245	16.736
	MAE_n	21.837	17.587	16.368	16.348	15.720
	Train	35	29	24	23	24
2.5e-4	MAE_i	14.747	16.644	17.752	20.310	16.398
	MAE_n	14.104	15.877	16.880	19.630	15.544
	Train	17	20	23	18	23
1e-4	MAE_i	15.804	15.419	15.464	16.033	16.327
	MAE_n	15.319	14.863	14.938	15.546	15.816
	Train	13	15	15	14	14

Experiment Result

- Online Learning Test Result : Effect of epsilon
  - initial lr : 1e-4
  - Margin : 60
  - linear delta : 10%
  - tanh delta : 0.5
  - exp delta : 0.75

	MAE_i	MAE_n	Train
no_reduction	221.991	22.029	22
linear	22.040	21.221	18
tanh	20.485	19.711	17
exp	21.056	20.299	17

epsilon : 14.3 (mean of train data variance)

	MAE_i	MAE_n	Train
no_reduction	19.649	18.827	17
linear	18.411	17.435	21
tanh	21.652	20.779	19
exp	20.024	18.981	22

epsilon : 10

	MAE_i	MAE_n	Train
no_reduction	21.965	20.945	22
linear	20.140	19.292	17
tanh	21.515	20.348	24
exp	18.398	17.415	20

epsilon : 5

	MAE_i	MAE_n	Train
no_reduction	23.483	21.515	42
linear	20.416	19.811	11
tanh	23.288	21.679	34
exp	22.54	21.416	24

epsilon : 3

Experiment Result

- Online Learning Test Result : Effect of margin
  - initial lr : 1e-4
  - Epsilon : 10
  - linear delta : 10%
  - tanh delta : 0.5
  - exp delta : 0.75

margin	margin exceeded
60	92
50	168
40	266
30	374

Offline test result

	MAE_i	MAE_n	Train
no_reduction	15.471	14.057	35
linear	13.894	12.704	31
tanh	15.886	14.547	35
exp	17.209	15.947	33

margin : 50

	MAE_i	MAE_n	Train
no_reduction	15.238	12.694	82
linear	16.223	13.755	78
tanh	17.225	14.898	78
exp	15.607	13.505	67

margin : 40

	MAE_i	MAE_n	Train
no_reduction	14.833	10.873	158
linear	15.746	11.659	167
tanh	13.931	10.836	130
exp	14.710	11.268	143

margin : 30

Conclusion & Future work

- This work is a methodological approach to online learning.
- As with typical batch learning, learning rates are very important hyperparameters.
- It is directly proportional to margin and MAE_i and inversely proportional to margin and train count.
- Various combinations are possible, so experimenting with the appropriate set values depending on the data can yield much better results.
- plan to reinforce the experiment using other reduction functions and learning points.

On-line Learning pseudo-code

Algorithm 3 OLR$(x, y, \epsilon, \varepsilon, \eta_0, \delta, \Omega, \mathcal{B}, W, \omega, \theta)$

Input:

- x, y : Input vector x and its true label y.
- ϵ : Threshold value $\epsilon \in \mathbb{R}, \epsilon > 0$
- ε : Minimum relative error improvement $\varepsilon \in \mathbb{R}, \varepsilon > 0$
- η_0 : Initial learning rate $\eta_0 \in \mathbb{R}, \eta_0 > 0$
- δ : Annealing rate $\delta \in \mathbb{R}, \delta > 0$
- Ω : Set of observed instances
- $\mathcal{B}$: Set of parameter vectors, $\mathcal{B} = \{\beta_1, \beta_2, \dots, \beta_{|\mathcal{B}|}\}$
- W : Set of associated weight of $\mathcal{B}$, $W = \{w_1, w_2, \dots, w_{|\mathcal{B}|}\}$
- ω : A reduction function $\omega = e^{-\alpha}, \alpha > 0$
- θ : Maximum size of $\mathcal{B}$, $\theta \in \mathbb{R}, \theta > 0$

Output: $p, \mathcal{B}, W, \Omega$

$\Omega \longleftarrow (x, y)$ /* add x, y to the set of observed instances */
$H \longleftarrow \sum_{i=1}^{|\mathcal{B}|} w_i$ /* w_i returns the weight value associated with parameter i */
$\beta_r \longleftarrow \text{sampling}\left(\frac{w_1}{H}, \frac{w_2}{H}, \dots, \frac{w_{|\mathcal{B}|}}{H}\right)$ /* sample a random β_r, r is it's index in $\mathcal{B}$ */
$\ell \longleftarrow 0$ /* initiate loss value */
$\hat{\ell} \longleftarrow \infty$
$t \longleftarrow 0$ /* initiate trial value */
$p \longleftarrow \frac{e^{(\beta_r^T \cdot x_r)}}{1+e^{(\beta_r^T \cdot x_r)}}$ /* Calculate probability of potential */
IF $abs\left(\log_2\left(p^y (1-p)^{(1-y)}\right)\right) \geq \epsilon$ **THEN** /* The prediction makes mistake */
  $\eta_e \longleftarrow \frac{\eta_0}{1+e/\delta}$
  $\hat{\beta} \longleftarrow \beta_r$ /* initiate a new parameter */
  $w_r \longleftarrow w_r \times \omega$ /* update weight value of β_r */
  IF $|\mathcal{B}| < \theta$ **THEN**
   WHILE relDiff$(\hat{\ell}, \ell) > \varepsilon$ /* Define relDiff$(a, b) = \frac{abs(a-b)}{abs(a)+abs(b)}$ */
    FOR $i \longleftarrow 1$ **TO** $|\Omega|$ **DO**
     $p_i = \frac{e^{\hat{\beta} \cdot x_i}}{1+e^{(\hat{\beta} \cdot x_i)}}$
     $\hat{\beta} \longleftarrow \hat{\beta} + \eta_e x_i (\mathrm{I}(y_i = 1) - p_i)$
   /* The indicator function I(a = 1) returns 1 if a = 1, otherwise 0 */
    ENDFOR
    $\hat{\ell} \longleftarrow \ell$
    $\ell \longleftarrow -\sum_{i \leq |\Omega|} \log\left(p_i^{y_i} (1-p_i)^{(1-y_i)}\right)$
    $t \longleftarrow t + 1$
   ENDWHILE
   $\mathcal{B} \longleftarrow \hat{\beta}$
   $W \longleftarrow 1$
  ENDIF
ENDIF
RETURN $p, \mathcal{B}, W, \Omega$

Source: Dinh Thanh Cong, “Mining potential iddeas for new product development in crowdsourcing application”, Master thesis of PNU

Prof. Hyerim Bae (hrbae@pusan.ac.kr)