FTRFL 논문 발표

ray.min

목차

CTR_prediction

Factorization Machine

FTRL-Proximal

FTRFL algorithm

kakao

CTR_prediction

Article 의 CTR을 예측하여 각자에게 prior 값을 차등하게 주자는 목적.

kakao

CTR_prediction

Article 의 CTR을 예측하여 각자에게 prior 값을 차등하게 주자는 목적.

- 현재는 thompson.py에 moment_method 가 구현되어 있음
- sample 들을 통해 beta distribution 의 alpha, beta 파라미터에 대한 추정을 하여 prior 를 주는 방법 (http://www.real-statistics.com/distribution-fitting/method-of-moments/method-of-moments-beta-distribution/)
- 클릭수가 min_click 을 넘긴 arm 들의 수가 min_arms 이상이면, 각 arm들의 평균을 sample 로 사용해 파라미터를 추정함

kakao

CTR_prediction

Article 의 CTR을 예측하여 각자에게 prior 값을 차등하게 주자는 목적.

- 현재는 thompson.py에 moment_method 가 구현되어 있음
- sample 들을 통해 beta distribution 의 alpha, beta 파라미터에 대한 추정을 하여 prior 를 주는 방법 (http://www.real-statistics.com/distribution-fitting/method-of-moments/method-of-moments-beta-distribution/)
- 클릭수가 min_click 을 넘긴 arm 들의 수가 min_arms 이상이면, 각 arm들의 평균을 sample 로 사용해 파라미터를 추정함
- arm 각각의 CTR 을 예측하는 것이 아니라, 한 bandit의 모든 arm이 같은 prior 를 받는 다른 의미의 CTR_prediction 임
- alpha, beta 값이 너무 크게 들어오는 경우가 많았음

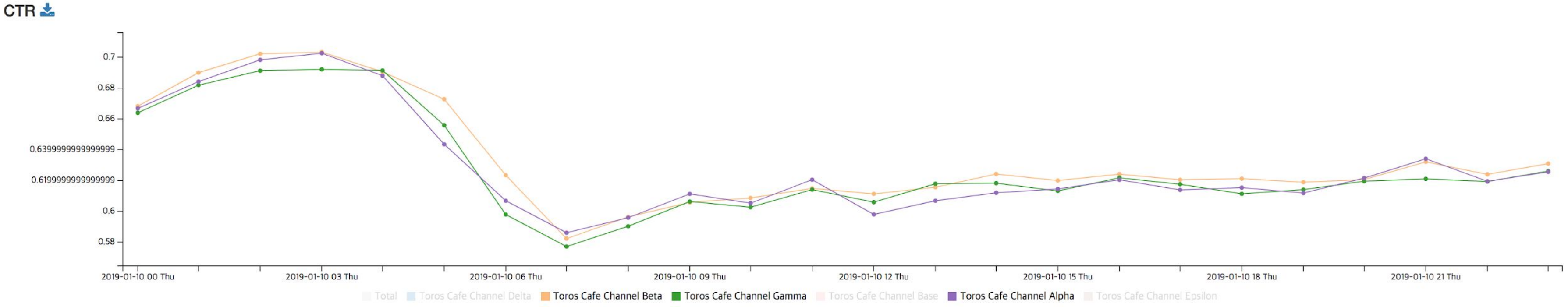
kakao

CTR_prediction

```
[root@aurochs-cafe-channel similar.channel (dev)]$ tail -f Log/reproduce.??.log | grep "mu_pool"
[INFO    ] 2019-01-10 16:42:57 [thompson.py] [update_arms:275] mu_pool: prior_alpha(425.825) prior_beta(9965.751)
[INFO    ] 2019-01-10 16:42:59 [thompson.py] [update_arms:275] mu_pool: prior_alpha(219.638) prior_beta(2774.627)
[INFO    ] 2019-01-10 16:43:13 [thompson.py] [update_arms:275] mu_pool: prior_alpha(424.651) prior_beta(9939.253)
[INFO    ] 2019-01-10 16:43:14 [thompson.py] [update_arms:275] mu_pool: prior_alpha(219.969) prior_beta(2777.425)
[INFO    ] 2019-01-10 16:43:20 [thompson.py] [update_arms:275] mu_pool: prior_alpha(1.933) prior_beta(33.842)
[INFO    ] 2019-01-10 16:43:28 [thompson.py] [update_arms:275] mu_pool: prior_alpha(1.931) prior_beta(33.822)
[INFO    ] 2019-01-10 16:48:20 [thompson.py] [update_arms:275] mu_pool: prior_alpha(430.934) prior_beta(10086.850)
[INFO    ] 2019-01-10 16:48:21 [thompson.py] [update_arms:275] mu_pool: prior_alpha(220.706) prior_beta(2786.814)
[INFO    ] 2019-01-10 16:48:36 [thompson.py] [update_arms:275] mu_pool: prior_alpha(1.931) prior_beta(33.822)
[INFO    ] 2019-01-10 16:53:41 [thompson.py] [update_arms:275] mu_pool: prior_alpha(432.029) prior_beta(10116.911)
[INFO    ] 2019-01-10 16:53:42 [thompson.py] [update_arms:275] mu_pool: prior_alpha(227.900) prior_beta(2877.468)
[INFO    ] 2019-01-10 16:53:56 [thompson.py] [update_arms:275] mu_pool: prior_alpha(1.931) prior_beta(33.822)
[INFO    ] 2019-01-10 16:58:49 [thompson.py] [update_arms:275] mu_pool: prior_alpha(435.336) prior_beta(10196.976)
[INFO    ] 2019-01-10 16:58:50 [thompson.py] [update_arms:275] mu_pool: prior_alpha(229.897) prior_beta(2900.624)
[INFO    ] 2019-01-10 16:59:01 [thompson.py] [update_arms:275] mu_pool: prior_alpha(1.931) prior_beta(33.824)
[INFO    ] 2019-01-10 17:04:17 [thompson.py] [update_arms:275] mu_pool: prior_alpha(441.280) prior_beta(10336.625)
[INFO    ] 2019-01-10 17:04:18 [thompson.py] [update_arms:275] mu_pool: prior_alpha(233.487) prior_beta(2949.856)
[INFO    ] 2019-01-10 17:04:36 [thompson.py] [update_arms:275] mu_pool: prior_alpha(1.932) prior_beta(33.834)
```

kakao

CTR_prediction



CTR_prediction

가장 대중적인 방법은 Logistic regression + SGD optimization

- 또다른 optimaztion 방법으로 Newton and Quasi-Newton methods, Coordinate Descent 등이 있음
- 그 중 SGD 가 제일 좋은 효과를 보이지만, 학습된 모델이 sparse 하지 못해 실서비스에선 큰 비용을 필요로함
- 모델의 sparsity와 좋은 성능을 위해 FOBOS, RDA, FTRL 등의 optimization algorithm이 선호됨

CTR_prediction

가장 대중적인 방법은 Logistic regression + SGD optimization

- 또다른 optimaztion 방법으로 Newton and Quasi-Newton methods, Coordinate Descent 등이 있음
- 그 중 SGD 가 제일 좋은 효과를 보이지만, 학습된 모델이 sparse 하지 못해 실서비스에선 큰 비용을 필요로함
- 모델의 sparsity와 좋은 성능을 위해 FOBOS, RDA, FTRL 등의 optimization algorithm이 선호됨

CTR_prediction

가장 대중적인 방법은 Logistic regression + SGD optimization

- logistic regression 방법의 단점은 feature 간의 non-linear information 을 잘잡지 못한다는 단점이 있음
- 해결 방법으로 LR model 의 input으로 conjunction feature를 만들어서 넣어주는 방법이 있음
- 하지만 quadratic number of new feature 를 만들어야 해서 좋지 못함

kakao

CTR_prediction

가장 대중적인 방법은 Logistic regression + SGD optimization

- logistic regression 방법의 단점은 feature 간의 non-linear information 을 잘잡지 못한다는 단점이 있음
- 해결 방법으로 LR model 의 input으로 conjunction feature를 만들어서 넣어주는 방법이 있음
- 하지만 quadratic number of new feature 를 만들어야 해서 좋지 못함
- Factorization Machine (FM) 모델 사용

CTR_prediction

가장 대중적인 방법은 Logistic regression + SGD optimization

- logistic regression 방법의 단점은 feature 간의 non-linear information 을 잘잡지 못한다는 단점이 있음
- 해결 방법으로 LR model 의 input으로 conjunction feature를 만들어서 넣어주는 방법이 있음
- 하지만 quadratic number of new feature 를 만들어야 해서 좋지 못함
- Factorization Machine (FM) 모델 사용
- FM의 optimization 방법으로 FTRL-Proximal 이 가장 좋은 성능과 sparsity 두마리 토끼를 잡을 수 있음

kakao

CTR_prediction

가장 대중적인 방법은 Logistic regression + SGD optimization

- logistic regression 방법의 단점은 feature 간의 non-linear information 을 잘잡지 못한다는 단점이 있음
- 해결 방법으로 LR model 의 input으로 conjunction feature를 만들어서 넣어주는 방법이 있음
- 하지만 quadratic number of new feature 를 만들어야 해서 좋지 못함
- Factorization Machine (FM) 모델 사용
- FM의 optimization 방법으로 FTRL-Proximal 이 가장 좋은 성능과 sparsity 두마리 토끼를 잡을 수 있음
- FTRFL (FM + FTRL-Proximal) (논문: Factorization Machines with Follw-The-Regularized-Leader for CTR prediction in Display Advertising)

kakao

Factorization Machine (FM) (논문: Factorization Machine)

- Supervised learning
- SVM 의 장점과 factorization model 의 장점을 섞은 모델임

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i \, x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j$$

- SVM 과 마찬가지로 real-valued feature vector 에 적용가능한 general predictor 임

Factorization Machine (FM) (논문: Factorization Machine)

- Supervised learning
- SVM 의 장점과 factorization model 의 장점을 섞은 모델임

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i \, x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j$$

- SVM 과 마찬가지로 real-valued feature vector 에 적용가능한 general predictor 임
- 하지만 SVM 보다 이런저런 점에서 좋다!

Factorization Machine (FM) vs SVM

- SVM 은 sparse data 에서 non-linear kernel 의 parameter 학습이 어려움
- linear time 안에 계산이 가능하므로, SVM의 support vector 같은 training data의 저장이 따로 필요없음
- SVM과 달리 dual form이 아닌 primal form에서 바로 optimization이 가능함
- SVD++, PITF, FPMC 와 같은 specialized models 들과 달리 general 한 predictor임.
- right input data (feature vector)를 잘사용함으로써 위의 모델들을 흉내낼 수 있음

Factorization Machine (FM) vs SVM

- SVM 은 sparse data 에서 non-linear kernel 의 parameter 학습이 어려움

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i \, x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j \qquad w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^n, \quad \mathbf{V} \in \mathbb{R}^{n \times k}$$

And $\langle \cdot, \cdot \rangle$ is the dot product of two vectors of size k:

$$\langle \mathbf{v}_i, \mathbf{v}_j \rangle := \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f} \qquad \hat{w}_{i,j} := \langle \mathbf{v}_i, \mathbf{v}_j \rangle$$

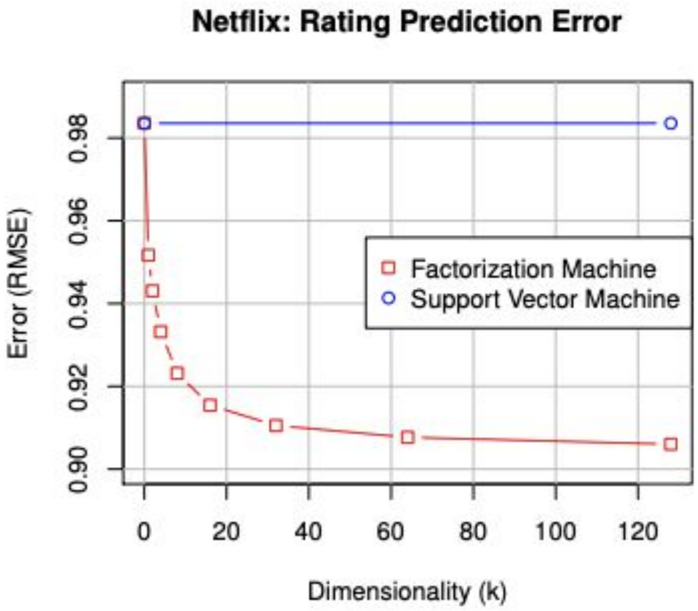
Interaction을 factorization 함으로써 interaction parameter 들의 independence를 없애기 때문에 가능

Factorization Machine (FM) vs SVM

- linear time 안에 계산이 가능하므로 : O(kn)

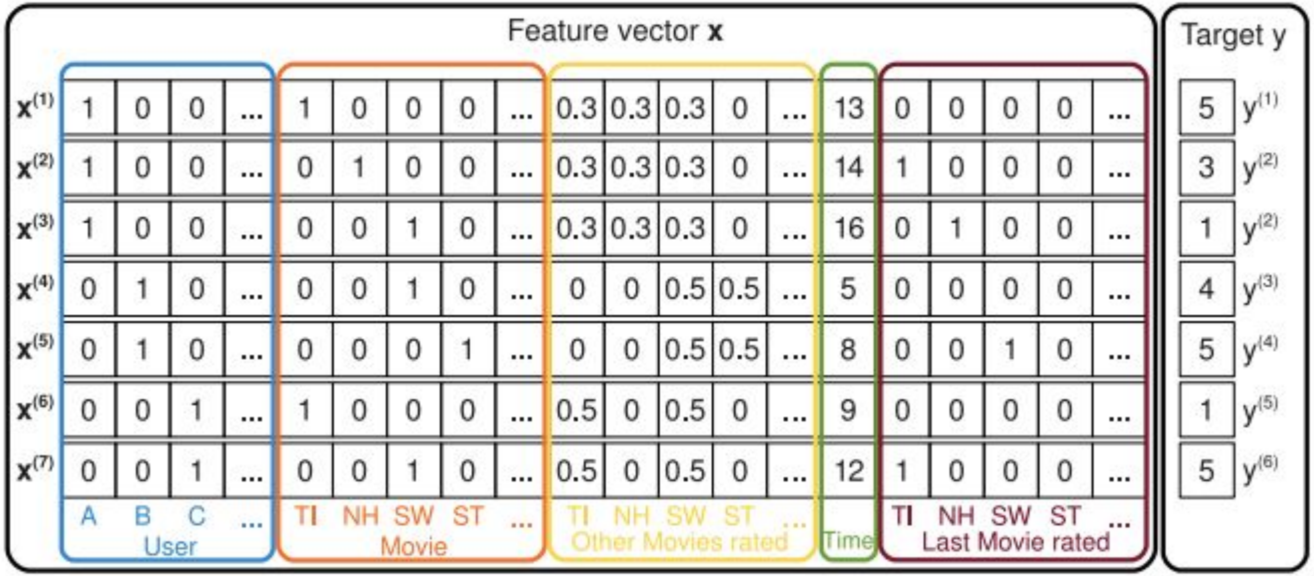
$$
\begin{aligned}
&\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j \\
=&\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j - \frac{1}{2} \sum_{i=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_i \rangle \, x_i \, x_i \\
=&\frac{1}{2} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{f=1}^{k} v_{i,f} \, v_{j,f} \, x_i \, x_j - \sum_{i=1}^{n} \sum_{f=1}^{k} v_{i,f} \, v_{i,f} \, x_i \, x_i \right) \\
=&\frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{n} v_{i,f} \, x_i \right) \left(\sum_{j=1}^{n} v_{j,f} \, x_j \right) - \sum_{i=1}^{n} v_{i,f}^2 \, x_i^2 \right) \\
=&\frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{n} v_{i,f} \, x_i \right)^2 - \sum_{i=1}^{n} v_{i,f}^2 \, x_i^2 \right)
\end{aligned}
$$

Factorization Machine (FM) vs SVM



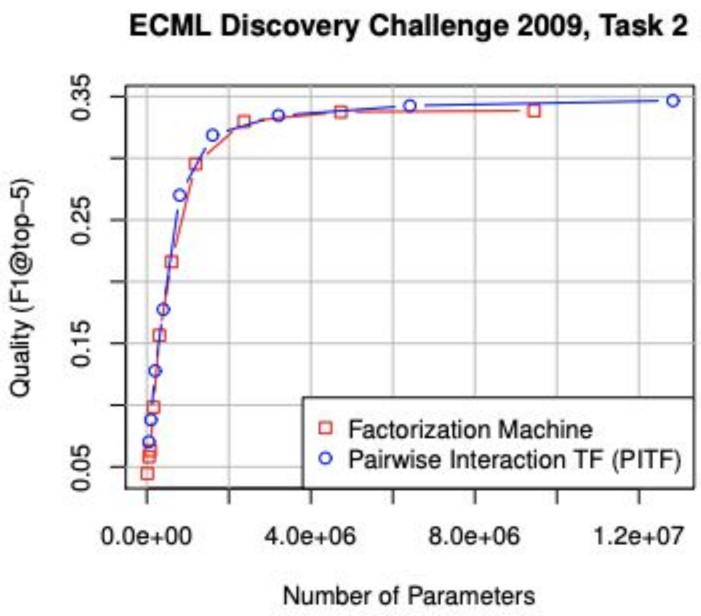
kakao

Factorization Machine (FM) vs Specialized factorization model



kakao

Factorization Machine (FM) vs Specialized factorization model



FTRL-Proximal (논문: Ad Click Prediction: a View from the Trenches)

- Online learning alogrithm
- 우수한 sparsity 와 convergence 를 보임
- OGD의 accuracy 와 RDA의 sparsity를 동시에 챙기기 위한 algorithm

FTRL-Proximal (논문: Ad Click Prediction: a View from the Trenches)

- Online learning alogrithm
- 우수한 sparsity 와 convergence 를 보임
- OGD의 accuracy 와 RDA의 sparsity를 동시에 챙기기 위한 algorithm

Algorithm 1 Per-Coordinate FTRL-Proximal with L_1 and L_2 Regularization for Logistic Regression

 # With per-coordinate learning rates of Eq. (2).
Input: parameters α, β, λ_1, λ_2
$(\forall i \in \{1, \ldots, d\})$, initialize $z_i = 0$ and $n_i = 0$
for $t = 1$ **to** T **do**
  Receive feature vector $\mathbf{x}_t$ and let $I = \{i \mid x_i \neq 0\}$
  For $i \in I$ compute

$$w_{t,i} = \begin{cases} 0 & \text{if } |z_i| \leq \lambda_1 \\ -\left(\frac{\beta + \sqrt{n_i}}{\alpha} + \lambda_2\right)^{-1} (z_i - \text{sgn}(z_i)\lambda_1) & \text{otherwise.} \end{cases}$$

  Predict $p_t = \sigma(\mathbf{x}_t \cdot \mathbf{w})$ using the $w_{t,i}$ computed above
  Observe label $y_t \in \{0, 1\}$
  for all $i \in I$ **do**
    $g_i = (p_t - y_t)x_i$ *#gradient of loss w.r.t.* w_i
    $\sigma_i = \frac{1}{\alpha}\left(\sqrt{n_i + g_i^2} - \sqrt{n_i}\right)$ *#equals* $\frac{1}{\eta_{t,i}} - \frac{1}{\eta_{t-1,i}}$
    $z_i \leftarrow z_i + g_i - \sigma_i w_{t,i}$
    $n_i \leftarrow n_i + g_i^2$
  end for
end for

FTRL-Proximal vs OGD (update 과정)

- OGD

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{g}_t, \qquad \eta_t = \frac{1}{\sqrt{t}}$$

- FTRL-Proximal

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \left(\mathbf{g}_{1:t} \cdot \mathbf{w} + \frac{1}{2} \sum_{s=1}^{t} \sigma_s \|\mathbf{w} - \mathbf{w}_s\|_2^2 + \lambda_1 \|\mathbf{w}\|_1 \right), \qquad \sigma_{1:t} = \frac{1}{\eta_t}$$

	Num. Non-Zero's	AucLoss Detriment
FTRL-Proximal	baseline	baseline
RDA	+3%	0.6%
FOBOS	+38%	0.0%
OGD-Count	+216%	0.0%

kakao

FTRL-Proximal vs OGD (update 과정)

- OGD

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{g}_t, \qquad \eta_t = \frac{1}{\sqrt{t}}$$

- FTRL-Proximal

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \left(\mathbf{g}_{1:t} \cdot \mathbf{w} + \frac{1}{2} \sum_{s=1}^{t} \sigma_s \|\mathbf{w} - \mathbf{w}_s\|_2^2 + \lambda_1 \|\mathbf{w}\|_1 \right), \qquad \sigma_{1:t} = \frac{1}{\eta_t}$$

$$\left(\mathbf{g}_{1:t} - \sum_{s=1}^{t} \sigma_s \mathbf{w}_s \right) \cdot \mathbf{w} + \frac{1}{\eta_t} \|\mathbf{w}\|_2^2 + \lambda_1 \|\mathbf{w}\|_1 + (\text{const}).$$

FTRL-Proximal vs OGD (update 과정)

- OGD

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{g}_t, \qquad \eta_t = \frac{1}{\sqrt{t}}$$

- FTRL-Proximal

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \left(\mathbf{g}_{1:t} \cdot \mathbf{w} + \frac{1}{2} \sum_{s=1}^{t} \sigma_s \|\mathbf{w} - \mathbf{w}_s\|_2^2 + \lambda_1 \|\mathbf{w}\|_1 \right), \qquad \sigma_{1:t} = \frac{1}{\eta_t}$$

$$\left(\mathbf{g}_{1:t} - \sum_{s=1}^{t} \sigma_s \mathbf{w}_s \right) \cdot \mathbf{w} + \frac{1}{\eta_t} \|\mathbf{w}\|_2^2 + \lambda_1 \|\mathbf{w}\|_1 + (\text{const}).$$

$$\mathbf{z}_{t-1} = \mathbf{g}_{1:t-1} - \sum_{s=1}^{t-1} \sigma_s \mathbf{w}_s$$

kakao

FTRL-Proximal vs OGD (update 과정)

- OGD

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{g}_t, \qquad \eta_t = \frac{1}{\sqrt{t}}$$

- FTRL-Proximal

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \left(\mathbf{g}_{1:t} \cdot \mathbf{w} + \frac{1}{2} \sum_{s=1}^{t} \sigma_s \|\mathbf{w} - \mathbf{w}_s\|_2^2 + \lambda_1 \|\mathbf{w}\|_1 \right), \qquad \sigma_{1:t} = \frac{1}{\eta_t}$$

$$\left(\mathbf{g}_{1:t} - \sum_{s=1}^{t} \sigma_s \mathbf{w}_s \right) \cdot \mathbf{w} + \frac{1}{\eta_t} \|\mathbf{w}\|_2^2 + \lambda_1 \|\mathbf{w}\|_1 + (\text{const}).$$

$$\mathbf{z}_{t-1} = \mathbf{g}_{1:t-1} - \sum_{s=1}^{t-1} \sigma_s \mathbf{w}_s$$

$$w_{t+1,i} = \begin{cases} 0 & \text{if } |z_{t,i}| \le \lambda_1 \\ -\eta_t (z_{t,i} - \operatorname{sgn}(z_{t,i}) \lambda_1) & \text{otherwise.} \end{cases}$$

kakao

FTRL-Proximal vs OGD (learning rate)

- Per-coordinate learning rate 사용
- OGD 에서는 all-coordinates가 같은 learning rate을 공유 $\eta_t = \frac{1}{\sqrt{t}}$

$$\Pr(\text{heads} \mid \text{coin}_i)$$

FTRL-Proximal vs OGD (learning rate)

- Per-coordinate learning rate 사용
- OGD 에서는 all-coordinates가 같은 learning rate을 공유 $\cancel{\eta_t = \frac{1}{\sqrt{t}}}$

$$\eta_{t,i} = \frac{1}{\sqrt{n_{t,i}}}$$

$$\Pr(\text{heads} \mid \text{coin}_i)$$

$$\eta_{t,i} = \frac{\alpha}{\beta + \sqrt{\sum_{s=1}^{t} g_{s,i}^2}},$$

FTRL-Proximal (memory saving method)

- L1 regularization
- Probabilistic Feature Inclusion
- Encoding Values with fewer bits (q2.13 encoding)
- Computing Learning Rates with Counts
- Subsampling Training Data
- …..

FTRFL algorithm (https://github.com/geffy/tffm)

- FM + FTRL-Proximal

$$\phi(w, x) = \sum_{i}^{n} \sum_{j>i}^{n} \langle v_i, v_j \rangle x_i x_j$$

$$\underset{w}{\operatorname{argmin}} \sum_{i} l(\phi(w, x), y) + \lambda \times r(w)$$

Test data \ Model	Our method (FTRFL)	FM with SGD
Day 1	**0.9836**	0.9128
Day 2	**0.9818**	0.9105
Day 3	**0.9809**	0.9021

kakao

FTRFL algorithm (https://github.com/geffy/tffm)

- FM + FTRL-Proximal

- **Regression**: $\hat{y}\boxed{(\mathbf{x})}$ can be used directly as the predictor and the optimization criterion is e.g. the minimal least square error on D.
- **Binary classification**: the sign of $\hat{y}(\mathbf{x})$ is used and the parameters are optimized for hinge loss or logit loss.
- **Ranking**: the vectors $\mathbf{x}$ are ordered by the score of $\hat{y}(\mathbf{x})$ and optimization is done over pairs of instance vectors $(\mathbf{x}^{(\mathbf{a})}, \mathbf{x}^{(\mathbf{b})}) \in D$ with a pairwise classification loss (e.g. like in [5]).

FTRFL algorithm (https://github.com/geffy/tffm)

- FM + FTRL-Proximal

- **Regression:** $\hat{y}(\mathbf{x})$ can be used directly as the predictor and the optimization criterion is e.g. the minimal least square error on D.
- **Binary classification**: the sign of $\hat{y}(\mathbf{x})$ is used and the parameters are optimized for hinge loss or logit loss.
- **Ranking**: the vectors $\mathbf{x}$ are ordered by the score of $\hat{y}(\mathbf{x})$ and optimization is done over pairs of instance vectors $(\mathbf{x}^{(\mathbf{a})}, \mathbf{x}^{(\mathbf{b})}) \in D$ with a pairwise classification loss (e.g. like in [5]).

감사합니다

kakao