Value estimation

Generalized linear models & Stochastic gradient descent

Sharat Chikkerur
sharat@alum.mit.edu

Principal Data Scientist Lead
Microsoft.

Overview

- Value estimation
- Generalized linear models
- Gradient Descent
- Demos

Value estimation

Value estimation

- Bidding value is the optimal strategy in a second price auction (Vickrey [1961])

Challenges

- Optimization objectives can be different
  - (CPM, CPC, CPA, VPA etc.)
- Downstream objectives may conflict with upstream objectives
  - high CTR impression/user may be low CVR impression/user
- Data arrives at different rates, quantity (attribution windows) and have different maturity scales
  - CTR models have high volume and granular features
  - CVR models will have lower data volume

Factored value formulation

Factored formulation for downstream (revenue) optimization

$$VPI = E[V|A] * P(A|Click) * P(Click|Impression)$$

$E[V|A]$	Value per action	Linear/Log-Linear/Poisson regression
$P(A|\mathrm{Click}) or E[A|\mathrm{Click}]$	Actions per click	Logistic/Poisson Regression
$P(\mathrm{Click}|\mathrm{Impression})$	CTR	Logistic regression

Fortunately, all of these regression can be implemented using a general setup.

Generalized linear models

A generalized linear predictor specifies

- A linear predictor of the form $\eta(x) = w^T x$
- A mean estimate μ
- A link function $g(\mu)$ such that $g(\mu) = \eta(x)$ that relates the mean estimate to the linear predictor.

This framework supports a variety of regression problems

Linear regression	$\mu = w^T x$
Log-linear regression	$\log(\mu) = w^T x$
Logistic regression	$\log(\frac{\mu}{1-\mu}) = w^T x$
Poisson regression	$\log(\mu) = w^T x$

Optimization

VW solves optimization of the form

$$\sum_i l(w^T x_i; y_i) + \lambda R(w)$$

Here, $l()$ is convex, $R(w) = \lambda_1 |w| + \lambda_2 ||w||^2$.

VW support a variety of loss function

Linear regression	$(y - w^T x)^2$
Logistic regression	$\log(1 + exp(-yw^T x))$
SVM regression	$\max(0, 1 - yw^T x)$
Quantile regression	$\tau(w^T x - y) * I(y < w^T x) + (1 - \tau)(y - w^T x) I(y > w^T x)$
Poisson regression	$y \log(y) - y \log(w^T x) - (y - \exp(w^T x))$

Deep dive: Gradient descent

If the loss function is convex and parametized by w, we can minimize risk by gradient descent.

$$w_{t+1} = w_t - \eta \frac{1}{n} \sum_{i}^{n} \nabla_w l(f_w(x_i), y_i)$$

This converges in linear time ($-\log(\textit{residual}) \sim t$).

We can speed up convergence by using second order information

$$w_{t+1} = w_t - H_w^{-1} \frac{1}{n} \sum_{i}^{n} \nabla_w l(f_w(x_i), y_i)$$

where $H_w = \nabla^2 l(f_w(x_i), y_i)$ This converges in linear time ($-\log\log(\text{residual}) \sim t$).

Stochastic gradient descent

We replace the real gradient $\frac{1}{n}\sum_i^n \nabla_w l(f_w(x_i), y_i)$ with a instantaneous estimate

$$w_{t+1} = w_t - \eta_t \nabla_w l(f_w(x_i), y_i)$$

Note that the scaling factor has also been replaced with a time variant version. At each step t, the example is **randomly** picked. Good convergence is obtained using $\eta_t \sim \frac{1}{t}$ or $\eta_t \sim \frac{1}{\sqrt{t}}$

The rate of convergence is much slower than batch version of gradient descent

	GD	2nd order GD	SGD
Iterations to accuracy (ρ)	$\log(\frac{1}{\rho})$	$\log\log(\frac{1}{\rho})$	$\frac{1}{\rho}$

Flavors of SGD

Variants of SGD differ in several dimensions

- Learning rate schedule : Determines how η_t is updated.
  - Adaptive McMahan et al. [2013]
  - Normalized Ross et al. [2013]
  - Importance aware Karampatziakis and Langford [2010]
- Weight update: Determines how w_t is computed based on $w_1 \dots w_t, \eta_1 \dots \eta_t$
  - Ordinary SGD. Optimizes w_t
  - Averaged SGD. Averages $w_{1\dots t}$
- Loss functions: Determines how gradient is computed based on $l(x_1, y_1) \dots l(x_t, y_t)$
  - Ordinary SGD. Optimizes using last w_t
  - RDA, FTRL. Optimizes based on all previous updates $w_{1\dots t}$.

Learning rate update schedule: SGD

`--sgd`

$$\eta_t = \lambda d^k \frac{t_0}{(t_0 + t)^p}$$

λ	`-l`
d	`--decay_learning_rate`
t_0	`--initial_t`
p	`--power_t`

Learning rate update schedule: Adaptive

`--adaptive`

- Scales the update based on all the previous gradient values.
- Useful for different dynamic ranges

Data: λ, T
Initialization w=0, G=0 (diagonal matrix) ;
for *i = 1,2,... m* **do**
  Set $g = \nabla_w l(w^T x_i; y_i)$;
  Set $w = w - G^{-\frac{1}{2}} s(w, x, y)$;
  Set $G_{jj} = G_{jj} + g^2, \forall j \in 1 \dots d$;
end

Learning rate update schedule: Invariant

`--invariant`

- Importance weights are useful in many applications: subsampling, boosting
- Algorithm invariant way of implementing importance weight is to replicate the example.
- Many implementations choose to scale the gradient instead. This may cause updates to overshoot and is equivalent to having a large learning rate.
- Importance **aware** updates ensure that the updates with importance weights **h** are equivalent to the updates applied when the instance is presented **h** times.

Learning rate update schedule: Invariant

`--invariant` Common approach

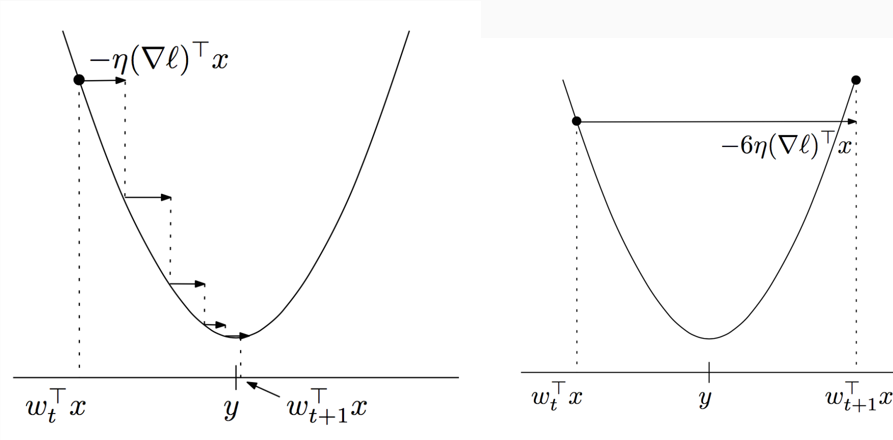
$$w_{t+1} = w_t - h\eta\nabla_w l(w_t^T x_t, y_t)$$

This is not the same as training on the example twice

$$v = w_{t+1} = w_t - \eta\nabla_w l(w_t^T x_t, y_t)$$

$$w_{t+2} = v - \eta\nabla_w l(v^T x_t, y_t)$$

Learning rate update schedule: Invariant



Learning rate update schedule: Invariant

For linear models $\nabla_w l = \frac{\partial l}{\partial p} x$.

$$w_{t+1} = w_t - s(h)x$$

. The scaling factor $s(h)$ has the recursive form

$$s(h+1) = s(h) + \eta \frac{\partial l}{\partial p}, p = (w_t - s(h)x)^T x$$

For squared loss, it turns out that

$$s(h) = \frac{w_t^T x - y}{x^T x}(1 - (1 - \eta x^T x)^h)$$

Learning rate update schedule: Invariant

The importance aware update can be determined for many loss functions Karampatziakis and Langford [2010]

Table 1: Importance Invariant and Imp^2 (cf. section 5) Updates for Various Loss Fu

Loss	$\ell(p, y)$	Invariant Update $s(h)$
Squared	$\frac{1}{2}(y-p)^2$	$\frac{p-y}{x^\top x}\left(1 - e^{-h\eta x^\top x}\right)$
Logistic	$\log(1+e^{-yp})$	$\frac{W(e^{h\eta x^\top x + yp + e^{yp}}) - h\eta x^\top x - e^{yp}}{y x^\top x}$ for $y \in \{-1, 1\}$
Exponential	e^{-yp}	$\frac{py - \log(h\eta x^\top x + e^{py})}{x^\top x y}$ for $y \in \{-1, 1\}$
Logarithmic	$y \log \frac{y}{p} + (1-y)\log\frac{1-y}{1-p}$	if $y = 0$ $\frac{p-1+\sqrt{(p-1)^2+2h\eta x^\top x}}{x^\top x}$ if $y = 1$ $\frac{p-\sqrt{p^2+2h\eta x^\top x}}{x^\top x}$
Hellinger	$2(1-\sqrt{py}-\sqrt{(1-p)(1-y)})$	if $y = 0$ $\frac{p-1+\frac{1}{4}(12h\eta x^\top x + 8(1-p)^{3/2})^{2/3}}{x^\top x}$ if $y = 1$ $\frac{p-\frac{1}{4}(12h\eta x^\top x + 8p^{3/2})^{2/3}}{x^\top x}$
Hinge	$\max(0, 1-yp)$	$-y\min\left(h\eta, \frac{1-yp}{x^\top x}\right)$ for $y \in \{-1, 1\}$
τ-Quantile	if $y > p$ $\tau(y-p)$ if $y \le p$ $(1-\tau)(p-y)$	if $y > p$ $-\tau\min(h\eta, \frac{y-p}{\tau x^\top x})$ if $y \le p$ $(1-\tau)\min(h\eta, \frac{p-y}{(1-\tau)x^\top x})$

Learning rate update schedule: Normalized

`--normalized`

- Features can have different dynamic ranges (scales) – usually got rid of by pre-scaling
- Offline mean-variance normalization may be expensive. No online version of normalization.
- Regret bounds for regular SGD algorithms depend on the norm of input.

Normalized updates

Intuition Ross et al. [2013]

- Keep track of the max value for each dimension
- If current value exceeds current max, scale down the weight as if new max was known all along
- Accumulate scaled value as pseudo-count to modulate learning rate.

Algorithm 1 NG(learning_rate η_t)

1. Initially $w_i = 0$, $s_i = 0$, $N = 0$
2. For each timestep t observe example (x, y)
   (a) For each i, if $|x_i| > s_i$
      i. $w_i \leftarrow \frac{w_i s_i^2}{|x_i|^2}$
      ii. $s_i \leftarrow |x_i|$
   (b) $\hat{y} = \sum_i w_i x_i$
   (c) $N \leftarrow N + \sum_i \frac{x_i^2}{s_i^2}$
   (d) For each i,
      i. $w_i \leftarrow w_i - \eta_t \frac{t}{N} \frac{1}{s_i^2} \frac{\partial L(\hat{y}, y)}{\partial w_i}$

Algorithm 1 NG(learning_rate η_t)

1. Initially $w_i = 0$, $s_i = 0$, $N = 0$
2. For each timestep t observe example (x, y)
   (a) For each i, if $|x_i| > s_i$
      i. $w_i \leftarrow \frac{w_i s_i^2}{|x_i|^2}$
      ii. $s_i \leftarrow |x_i|$
   (b) $\hat{y} = \sum_i w_i x_i$
   (c) $N \leftarrow N + \sum_i \frac{x_i^2}{s_i^2}$
   (d) For each i,
      i. $w_i \leftarrow w_i - \eta_t \frac{t}{N} \frac{1}{s_i^2} \frac{\partial L(\hat{y}, y)}{\partial w_i}$

FTPRL

`--ftrl`

Reformulation of gradient descent: Standard gradient descent update with learning rate η can be rewritten as the solution to

$$x_{t+1} = \operatorname*{argmin}_x \left(g_t\ x + \frac{1}{2\eta} ||x - x_t||_2^2 \right)$$

Solving the argmin yield the familiar update rule

$$x_{t+1} = x_t - \eta g_t$$

For adaptive updates, η is replaced by η_t.

Sparse updates

FOBOS (Duchi and Singer [2009]) explicitly adds L1 penalty to the optimization

$$x_{t+1} = \operatorname*{argmin}_x \left(g_t x + \lambda ||x||_1 + \frac{1}{2\eta} ||x - x_t||_2^2 \right)$$

FTRL (Follow the regularized leader)

RDA (Xiao [2010]) optimizes over all the previous gradient steps.

$$x_{t+1} = \operatorname*{argmin}_x \left(g_{1:t}x + \lambda ||x||_1 + \frac{1}{2\eta}||x||_2^2 \right)$$

Note: In RDA, L2 regularization is proximal to the origin.

FTPRL (Follow the *proximal* regularized leader)

FTPRL provides regularization around the previous updates instead of the origin (like RDA).

$$x_{t+1} = \operatorname*{argmin}_x \left(g_{1:t}x + \lambda||x||_1 + \frac{1}{2}\sum_{s=1}^{t} \sigma_s ||x - x_s||_2^2 \right)$$

Adding L2 regularization, we have

$$x_{t+1} = \operatorname*{argmin}_x \left(g_{1:t}x + \lambda_1||x||_1 + \lambda_2||x||_2^2 + \frac{1}{2}\sum_{s=1}^{t} \sigma_s ||x - x_s||_2^2 \right)$$

Here $\sigma_{1:t} = \eta_t$

Optimization

$$x_{t+1} = \underset{x}{\operatorname{argmin}} \left(g_{1:t}x + \lambda_1 ||x||_1 + \lambda_2 ||x||_2^2 + \frac{1}{2} \sum_{s=1}^{t} \sigma_s ||x - x_s||_2^2 \right)$$

$$x_{t+1} = \underset{x}{\operatorname{argmin}} \left(\left(g_{1:t} - \sum_{s=1}^{t} \sigma_s x_s) \right) x + \left(\lambda_2 + \frac{\sigma_{1:t}}{2} \right) x^2 + \lambda_1 ||x||_1 \right)$$

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
$(\forall i \in \{1, \dots, d\})$, 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

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

  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

Summary

- Default behavior `--normalized --invariant --adaptive`
- If you have variable dynamic ranges, rely on `--adaptive`
- If using importance weights, rely on `--invariant`
- If you can't afford multiple passes through the data, rely on `--ftrl`

Regression

- Linear regression `--loss_function square`
- Quantile regression
  `--loss_function quantile --quantile_tau <=0.5>`

Binary classification

- Note: a linear regressor can be used as a classifier as well
- Logistic loss
  `--loss_function logistic, --link logistic`
- Hinge loss (SVM loss function) `--loss_function hinge`
- Report binary loss instead of logistic loss `--binary`

Gradient update rule

- Classic SGD `--sgd`
- Adaptive learning rate `--adaptive`
- Per-feature normalized `--normalized`
- Second order update
  `--bfgs --conjugate_gradient --mem`
- Follow the leader proximal
  `--ftrl --ftrl_alpha --ftrl_beta`

SGD parameters

- L1 regularization `--l1`
- L2 regularization `--l2`

Demo/Questions?

References

Alekh Agarwal, Oliveier Chapelle, Miroslav Dudík, and John Langford.

AzureML. Azureml: Anatomy of a machine learning service. *Journal of Machine Learning Reserach*, 50, 2016.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3: 993–1022, 2012. ISSN 15324435. doi: 10.1162/jmlr.2003.3.4-5.993.

Andrei Broder. Computational advertising and recommender systems. *Proceedings of the 2008 ACM conference on Recommender systems SE - RecSys '08*, pages 1–2, 2008. doi: doi:10.1145/1454008.1454009. URL `citeulike-article-id:4026181{%}5Cnhttp://dx.doi.org/10.1145/1454008.1454009`.

Tushar Chandra, Eugene Ie, Kenneth Goldman, Tomas Lloret

Llinares, Jim McFadden, Fernando Pereira, Joshua Redstone, Tal Shaked, and Yoram Singer. Sibyl: a system for large scale machine learning. *Keynote I PowerPoint presentation, Jul*, 28, 2010.

Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *The Journal of Machine Learning Research*, 13(1):165–202, 2012.

John Duchi and Yoram Singer. Efficient Online and Batch Learning Using Forward Backward Splitting. *Journal of Machine Learning Research*, 10:2899–2934, 2009. ISSN 15324435.

Yoav Freund, Robert Schapire, and N Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14 (771-780):1612, 1999.

Jarvis Haupt and Robert Nowak. Signal reconstruction from noisy random projections. *Information Theory, IEEE Transactions on*, 52(9):4036–4048, 2006.

Matthew D Hoffman, David M Blei, and Francis Bach. Online Learning for Latent Dirichlet Allocation. *Advances in Neural Information Processing Systems*, 23:1–9, 2010. ISSN 08912017. doi: 10.1145/1835804.1835928.

Nikos Karampatziakis and John Langford. Online importance weight aware updates. *arXiv preprint arXiv:1011.1576*, 2010.

C. Karande, A. Mehta, and R. Srikant. Optimizing budget constrained spend in search advertising. *WSDM '13*, page 697, 2013.

J Langford, L Li, and A Strehl. Vowpal wabbit online learning project, 2007.

Kc Lee, Ali Jalali, and Ali Dasdan. Real Time Bid Optimization with Smooth Budget Delivery in Online Advertising. *arXiv preprint arXiv:1305.3011*, pages 1–13, 2013.

H Brendan McMahan, Gary Holt, D Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. Ad click prediction: a view from the trenches. *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230, 2013. doi: 10.1145/2487575.2488200.

Steffen Rendle. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, May 2012. ISSN 2157-6904.

Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks: estimating the click-through rate for new ads.

In *Proceedings of the 16th international conference on World Wide Web*, pages 521–530. ACM, 2007.

Stéphane Ross, Paul Mineiro, and John Langford. Normalized online learning. *arXiv preprint arXiv:1305.6646*, 2013.

Qinfeng Shi and John Langford. Hash Kernels for Structured Data. *Journal of Machine Learning Research*, 10:2615–2637, 2009.

William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 16(1):8–37, 1961.

K Weinberger and A Dasgupta. Feature hashing for large scale multitask learning. *Proceedings of the 26th ...*, 2009.

Lin Xiao. Dual Averaging Methods for Regularized Stochastic Learning and Online Optimization. *Journal of Machine Learning Research*, 11:2543–2596, 2010. ISSN 15324435.

Weinan Zhang, Yifei Rong, Jun Wang, Tianchi Zhu, and Xiaofan

Wang. Feedback control of real-time display advertising. Technical report, 2016.

Martin A Zinkevich, Markus Weimer, Alex Smola, and Lihong Li. Parallelized stochastic gradient descent.