Vowpal Wabbit

Scalable machine learning library

Sharat Chikkerur
sharat@alum.mit.edu

Principal Data Scientist Lead
Microsoft.

Vowpal Wabbit [1]

VOWPAL WABBIT

- Fast, online , scalable learning system
- Supports out of core execution with in memory model
- Scalable (Terascale)
  - 1000 nodes [13, 1]
  - Billions of examples
  - Trillions of unique features.
- Actively developed

[1] https://github.com/JohnLangford/vowpal_wabbit

Features

- Command line support
- Multi-langauge support: Python, C#, Java bindings
- Flexible input format: Allows free form text, identifying tags, multiple labels
- Speed: Supports online learning
- Scalable
  - Streaming removes row limit.
  - Hashing removes dimension limit
- Distributed training: models merged using *AllReduce* operation.
- Cross product features: Allows multi-task learning

Vowpal wabbit tricks

- Feature hashing
    - Makes the model fit on one machine.
    - Provides some degree of regularization
- Caching
    - Converts sparse input data to compact dense binary format. Makes multiple passes faster
- MPI style *AllReduce*
    - Provides scaling over large amounts of data
- Namespaces and interactions
    - Makes specifying cross product features easy
    - Allows multi-task learning using a single model (e.g. personalized spam filter)

Detour: Feature hashing

- Feature hashing can be used to reduce dimensions of sparse features.
  - Unlike random projections [9] , retains sparsity
  - Preserves dot products (random projection preserves distances).
  - Model can fit in memory.
- **Unsigned** [19]

  Consider a hash function $h(x) : [0 \dots N] \to [0 \dots m], m << N$.

  $$\phi_i(x) = \sum_{j:h(j)=i} x_j$$

- **Signed** [21]

  Consider additionaly a hash function $\xi(x) : [0 \dots N] \to \{1, -1\}$.

  $$\phi_i(x) = \sum_{j:h(j)=i} \xi(j) x_j$$

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	$y \log y - yw^T x - y + \exp(w^T x)$

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	$\mu = w^T x$
SVM regression	$\mu = w^T x$
Quantile regression	$\mu = w^T x$
Logistic regression	$\log(\frac{\mu}{1-\mu}) = w^T x$
Poisson regression	$\log(\mu) = w^T x$

Swiss army knife of online algorithms

- **Binary classification**
- **Multiclass classification**
- **Linear regression**
- **Quantile regression**
- **Topic modeling (online LDA)**
- Structured prediction
- Active learning
- **Recommendation (Matrix factorization)**
- Contextual bandit learning (explore/exploit algorithms)
- Reductions

Vowpal Wabbit: One stop shop

Challenge	Variant	VW option
Filtering	Quantile regression	`--loss_function quantile`
Item recommendation	Content based filtering	`--lrq`
	Collaborative filtering	`--rank`
Value estimation	Linear regression	`--loss_function square`
	Logistic regression	`--loss_function logistic`
	Poisson regression	`--loss_function poisson`
Budget allocation	Bandit optimization	`--cb`
Creative testing	Bandit optimization	`--cb`
User segmentation	Multiclass classification	`--ooa | --csooa`
	Topic modeling	`--lda`

Practical matters

Input format

Label Importance [Tag]—namespace Feature . . . — namespace Feature . . .

namespace = String[:Float]

feature = String[:Float]

Examples:

- 1 — 1:0.01 32:-0.1
- example—namespace normal text features
- 1 3 tag—ad-features ad description —user-features name address age

Input options

- data file -d datafile
- network --daemon --port <port=26542>
- compressed data --compressed
- stdin cat <data> — vw

Manipulation options

- ngrams `--ngram`
- skips `--skips`
- quadratic interaction `-q args`. e.g `-q ab`
- cubic interaction `--cubic args`. e.g. `--cubic ccc`

Output options

- Examining feature construction `--audit`
- Generating prediction `--predictions` or `-p`
- Unnormalized predictions `--raw_predictions`
- Testing only `--testonly` or `-t`

Model options

- Model size `--bit_precision` or `-b` . Number of coefficients limited to 2^b
- Update existing model `--initial_regressor` or `-i`.
- Final model destination `--final_regressor` or `-f`
- Readable model definition `--readable_model`
- Readable feature values `--invert_hash`
- Snapshot model every pass `--save_per_pass`
- Weight initialization `--initial_weight` or `--random_weights`

Scaling

Stochastic gradient descent with adaptive updates

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

Algorithm 1: Single node algorithm

Multi node algorithm using gradient descent

Data: Data split across nodes $1 \dots m$

for *iteration* $t \in 1, 2 \dots T$ **do**

  for *node* $k \in 1, 2, \dots m$ **do**

    Compute w^k and G^k ;

  end

  Compute a weighted average $\bar{w}$ using *AllReduce* ;

  $\bar{w} = \left(\sum_k G^k\right)^{-1} \left(\sum_k w^k G^k\right)$;

  Compute a weighted average $\bar{G}$ using *AllReduce* ;

  $\bar{G} = \left(\sum_k G^k\right)^{-1} \left(\sum_k (G^k)^2\right)$;

  for *node* $k \in 1, 2, \dots m$ **do**

    Set $w^k = \bar{w}$ and $G^k = \bar{G}$;

  end

end

All reduce

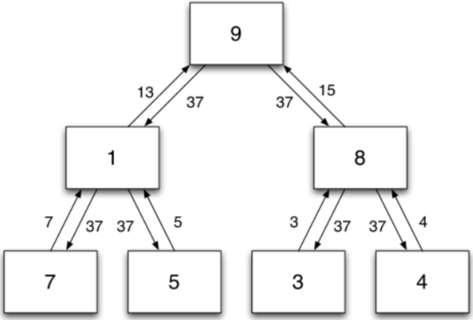


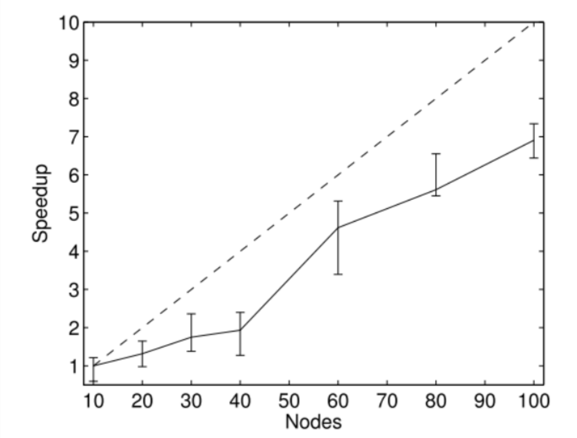
Figure 1: AllReduce operation. Initially, each node holds its own value. Values are passed up the tree and summed, until the global sum is obtained in the root node (reduce phase). The global sum is then passed back down to all other nodes (broadcast phase). At the end, each node contains the global sum.

Empirical results

Datasets

- Display ad CTR
    - 2.3B examples
    - 125 non zero features per instance.
    - hashed to 24 bits
- Splice site recognition
    - 50M examples
    - 3300 non zero features per instance

Speedup



- Results for display advertising dataset
- Relative to time required for a 10 node run
- All data points have the same test error

RTB Optimization using Vowpal Wabbit

Challenge	Variant	VW option
Filtering	Quantile regression	`--loss_function quantile`
Item recommendation	Content based filtering Collaborative filtering	`--lrq` `--rank`

Value estimation	Linear regression
	Logistic regression
	Poisson regression
		`--loss_function square`
		`--loss_function logistic`
		`--loss_function poisson`

Budget allocation	Bandit optimization	`--cb`
Creative testing	Bandit optimization	`--cb`
User segmentation	Multiclass classification Topic modeling	`--ooa | --csooa` `--lda`

Demos

Installing

Mac

```
git clone https://github.com/JohnLangford/vowpal_wabbit
make
```

Linux

```
apt-get install libboost-program-options-dev zlib1g-dev
apt-get install libboost-python-dev
git clone git://github.com/JohnLangford/vowpal_wabbit.git
make
```

Regression (Demo)

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

Binary classification (Demo)

- 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`

Multiclass classification (Demo)

- One against all `--oaa <k>`
- Error correcting tournament `--ect <k>`
- Online decision trees `---log_multi <k>`
- Cost sensitive one-against-all `--csoaa <k>`

LDA options (Demo)

- Number of topics `--lda`
- Prior on per-document topic weights `--lda_alpha`
- Prior on topic distributions `--lda_rho`
- Estimated number of documents `--lda_D`
- Convergence parameter for topic estimation `--lda_epsilon`
- Mini batch size `--minibatch`

Daemon mode (Demo)

- Loads model and answers any prediction request coming over the network
- Preferred way to deploy a VW model
- Options
  - `--daemon`. Enables demon mode
  - `--testonly` or `-t`. Does not update the model in response to requests
  - `--initial_model` or `-i`. Model to load
  - `--port <arg>`. Port to listen to the request
  - `--num_children <arg>`. Number of threads listening to request

Python (Demo)

- VW supports two python interfaces
    - pyvw provides low level wrapper
    - sklearn_vw provides sklearn compatible interface

References

[1] Alekh Agarwal, Oliveier Chapelle, Miroslav Dudík, and John Langford.

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

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2012.

[4] Andrei Broder. Computational advertising and recommender systems. *Proceedings of the 2008 ACM conference on Recommender systems SE - RecSys '08*, pages 1–2, 2008.

[5] 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.

[6] 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.

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

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

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

[10] 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.

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

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

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

[14] 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.

[15] 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.

[16] Steffen Rendle. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, May 2012.

[17] 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.

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

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

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

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

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

[23] Weinan Zhang, Yifei Rong, Jun Wang, Tianchi Zhu, and Xiaofan Wang. Feedback control of real-time display advertising. Technical report, 2016.

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