Vowpal Wabbit Overview

Sharat Chikkerur

September 10, 2015

Overview

- Fast, online , scalable learning system
- Supports out of core execution with in memory model
- Scalable (Terascale)
  - 1000 nodes **??**
  - Billions of examples
  - Trillions of unique features.
- Actively developed `https://github.com/JohnLangford/vowpal_wabbit`

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

Features

- 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)

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

Detour: Feature hashing

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

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

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

- **Signed ?**

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

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

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

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`

Input format (Demo)

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

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

Example

```
vw -i model.vw  -t --daemon --quiet --port 26542
echo " xyz-example| x y z" | netcat localhost 26542
pkill -9 -f 'vw.*--port 26542'
```

Compendium of flags

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`
- Loss function `--loss_function = squared,hinge,logistic,quantile (--quantile_ta`

Learning rate update

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

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

Scale out

Stochastic gradient descent with adaptive updates

Data: λ, T
Initialization w=0, G=0 (diag
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 $G_{jj} = G_{jj} + g^2, \forall j \in 1$
end

Algorithm 1:

Multi node algorithm using gradient descent

Data: Data split across nodes $1 \ldots m$
for *iteration* $t \in 1, 2 \ldots T$ **do**
    for *node* $k \in 1, 2, \ldots m$ **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* $k \in 1, 2, \ldots m$ **do**
        Set $w^k = \bar{w}$ and $G^k = \bar{G}$;
    end
end

All reduce

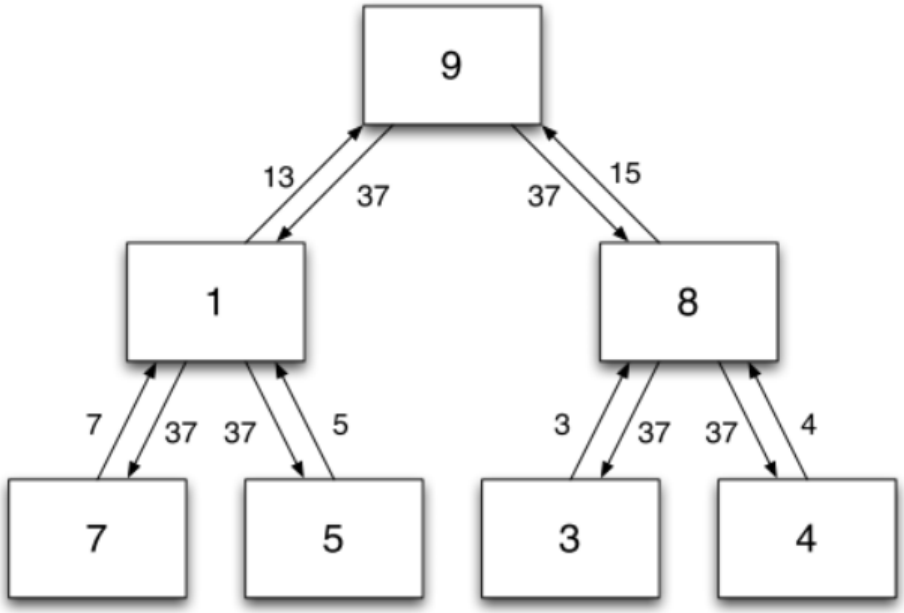


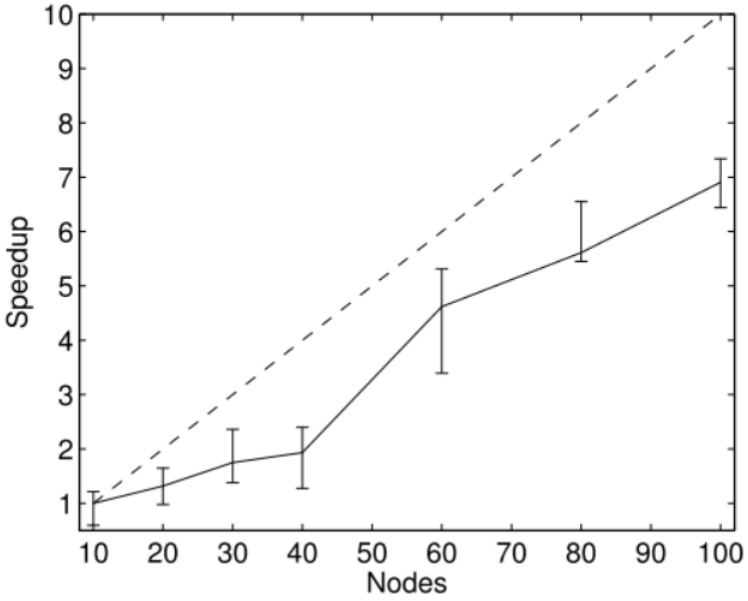
Figure 1: AllReduce operation. Initially, each node holds its own value. Values are passed up the tree and summed, until the global sum is obtained in the root node (reduce phase). The global sum is then passed back down to all other nodes (broadcast phase). At the end, each node contains the global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

Timing

	5%	50%	95%	Max	Comm. time
Without spec. execution	29	34	60	758	26
With spec. execution	29	33	49	63	10

Table 2: Distribution of computing times (in seconds) over 1000 nodes with and without speculative execution. First three columns are quantiles. Times are average per iteration (excluding the first one) for the splice site recognition problem.

Nodes	100	200	500	1000
Comm time / pass	5	12	9	16
Median comp time / pass	167	105	43	34
Max comp time / pass	462	271	172	95
Wall clock time	3677	2120	938	813

Table 3: Computing times to obtain a fixed test error on the splice site recognition data, using different numbers of nodes. The first 3 rows are averages per iteration (excluding the first pass over the data).

References

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

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.

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.

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

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

K Weinberger and A Dasgupta. Feature hashing for large scale multitask learning. *Proceedings of the 26th . . .*, 2009. URL `http://dl.acm.org/citation.cfm?id=1553516`.

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