Variants of Combinations of Additive and Multiplicative Updates for GRU Neural Networks

Ali H. Mirza
Department of Electrical and Electronics Engineering
Bilkent University, Ankara 06800, Turkey
mirza@ee.bilkent.edu.tr

***Abstract*—In this paper, we formulate several variants of the mixture of both the additive and multiplicative updates using stochastic gradient descent (SGD) and exponential gradient (EG) algorithms respectively. We employ these updates on the gated recurrent unit (GRU) networks. We then derive the gradient-based updates for the parameters of the GRU networks. We propose four different updates as a mean, minimum, even-odd and balanced set of updates for the GRU network. Through an extensive set of experiments, we demonstrate that these update variants perform better than simple SGD and EG updates. Overall, we observed that GRU-Mean update achieved the minimum cumulative and steady-state error performance. We also simulated the same set of experiments on the long short-term memory (LSTM) networks.**

Keywords. Additive updates, multiplicative updates, GRU, online learning, LSTM.

I. Introduction

Online learning [1] is of imperative importance that is mainstream in the machine learning [2], signal processing literature [3]. Online learning is widely used for classification [2], regression [4] and prediction purposes [5]. The online learning algorithms optimize the corresponding cost or performance measures [1]. Online learning algorithms operate sequentially on the data [1]-[5]. Batch learning algorithms are somewhat slower and expensive in performance for the training part [6]. Online learning algorithms can be linear as well as non-linear [1]-[3]. For big data applications with high dimensions, non-linear online learning methods are much suitable and perform better than linear algorithms [1].

For the time series or temporal data sequences, most commonly hidden markov models (HMMs) [7] or discrete symbolic grammar learning algorithms (SGLAs) [8] are used. Usually, SGLAs works worse on the noisy data. Similarly, the HMMs works on discrete state spaces. To alleviate these issues, recurrent neural networks (RNNs) are used to process such sequential data [9]. RNNs maps the real-valued input data sequences to real-valued output data sequences by performing gradient descent. The training of the RNN is very difficult as it suffers from the problem of vanishing and exploding gradients [10]. To mitigate this issue, we use the GRU networks [11] as the special case of the RNNs. The GRU networks have a gating structure that controls the flow of the information, unlike the RNNs. The GRU networks work well on the sequence-based tasks and are capable enough to capture the long-term data dependency [11].

978-1-5386-1501-0/18/$31.00

In order to train the RNNs or the GRU networks, we use the real-time recurrent learning (RTRL) algorithm [12] that is highly efficient in calculating the gradients. For the additive updates, stochastic gradient descent (SGD) based algorithm [13] is used for updating the parameters of the network. Besides these additive updates, exponential gradient (EG) based algorithm [14] is used to compute the multiplicative updates for updating the network parameters. In this paper, we devise some variants of the combinations of the additive and multiplicative updates. We formulate four different types of the update variants namely mean, minimum, even-odd and balanced updates.

Our main contributions in this paper are as follows:

- We proposed several variants of the mixture of the additive and multiplicative updates for the GRU network.
- These updates are generic, as we applied the same set of updates on the LSTM networks and achieved similar trends as we got for the GRU networks.
- These set of updates can be applied to any data set and must be cross-validated first in order to see which particular update is most suitable.

The organization of the paper is as follows. In Section II, we state the model and problem description. In Section II-A, we state the additive and multiplicative updates based on the loss function. In Section II-B, we mention four types of updates using a mixture of SGD and EG updates as defined in Section II-A. We then derive the gradient-based update for the GRU based network parameter in Section II-C. In Section III, we perform extensive experiments and then finally conclude the paper in Section IV.

II. Problem Description

In this paper, all vectors are column vectors and denoted by boldface lower case letters. Matrices are represented by boldface upper case letters. For a vector $\boldsymbol{u}$, $|\boldsymbol{u}|$ is the $\ell 1$-norm and $\boldsymbol{u}^T$ is the ordinary transpose. For a vector $\boldsymbol{x_t}$, $x_{t,i}$ and $\boldsymbol{x_{ti}}$ are the i^{th} element and the i^{th} column of the vector $\boldsymbol{x}_t$, respectively. Similarly, for a matrix $\boldsymbol{W}$, w_{ij} is the i^{th} row and j^{th} column entry.

Given that we have an input data sequence $\{\boldsymbol{U}_t\}_{t=1}^n$ and $\{d_t\}_{t=1}^n$ be the corresponding values. We try to estimate $\hat{d}_t$ and then calculate the loss associated with it. In $\{\boldsymbol{U}_t\}_{t=1}^n$, $\boldsymbol{u}_{t_i} \in$

$\mathbb{R}^m$ for $i, \ldots, \mu_t$, is the i^{th} component of the $\{\boldsymbol{U}_t\}_{t=1}^n$ and μ_t is the corresponding length of the incoming data sequence. μ_t may or may not be of same length for each data sequence. The i^{th} component of $\{\boldsymbol{U}_t\}_{t=1}^n$ can be a vector $\boldsymbol{u}_{t_i}$ or a scalar u_{t_i}. In order to process this variable length sequential information, we use the recurrent neural networks (RNNs). We parse the individual entries (for scalar case) and columns (for vector case) of $\boldsymbol{U}_t$ to the RNN. For the i^{th} column of $\boldsymbol{U}_t$, the RNN can be written as follows:

$$\boldsymbol{h}_{t_i} = \kappa(\boldsymbol{W}\boldsymbol{u}_{t_i} + \boldsymbol{V}\boldsymbol{h}_{(t-1)_i}), \tag{1}$$

where $\boldsymbol{h}_{t_i} \in \mathbb{R}^q$ is the state vector and $\boldsymbol{u}_{t_i} \in \mathbb{R}^m$ is the input vector for $i = 1, \ldots, \mu_t$. The RNN coefficient weight matrices are $\boldsymbol{V} \in \mathbb{R}^{q\times q}$ and $\boldsymbol{W} \in \mathbb{R}^{q\times m}$. The function $\kappa(\cdot)$ is commonly set to $\tanh(\cdot)$ and apply pointwise to vectors.

As a special case of the RNNs, we use the GRU network with one hidden layer defined as follows [11]:

$$\boldsymbol{z}_t = \sigma\big(\boldsymbol{W}^{(z)}\boldsymbol{u}_t + \boldsymbol{V}^{(z)}\boldsymbol{h}_{t-1}\big) \tag{2}$$

$$\boldsymbol{r}_t = \sigma\big(\boldsymbol{W}^{(r)}\boldsymbol{u}_t + \boldsymbol{V}^{(r)}\boldsymbol{h}_{t-1}\big) \tag{3}$$

$$\begin{aligned}\tilde{\boldsymbol{h}}_t &= \kappa\big(\boldsymbol{W}^{(h)}\boldsymbol{u}_t + \boldsymbol{V}^{(h)}(\boldsymbol{h}_{t-1}\odot \boldsymbol{r}_t)\big)\\ &= \kappa\big(\boldsymbol{W}^{(h)}\boldsymbol{u}_t + \boldsymbol{V}^{(h)}\boldsymbol{\Delta}^{(\boldsymbol{r}_t)}\boldsymbol{h}_{t-1}\big)\end{aligned} \tag{4}$$

$$\begin{aligned}\boldsymbol{h}_t &= (\boldsymbol{1}-\boldsymbol{z}_t)\odot\tilde{\boldsymbol{h}}_t + \boldsymbol{z}_t\odot\boldsymbol{h}_{t-1}\\ &= \boldsymbol{\Delta}^{(\boldsymbol{1}-\boldsymbol{z}_t)}\tilde{\boldsymbol{h}}_t + \boldsymbol{\Delta}^{(\boldsymbol{z}_t)}\boldsymbol{h}_{t-1},\end{aligned} \tag{5}$$

where $\boldsymbol{u}_t \in \mathbb{R}^m$ is the input vector and $\boldsymbol{h}_t \in \mathbb{R}^q$ is the output vector. $\boldsymbol{z}_t$ and $\boldsymbol{r}_t$ are the reset and update gates respectively. $\boldsymbol{\Delta}^{(\cdot)}$ is a diagonal matrix and the coefficient weight matrices, i.e., $\boldsymbol{W}^{(\cdot)}$ and $\boldsymbol{V}^{(\cdot)}$, are set to appropriate dimensions. The functions $\sigma(\cdot)$ and $\kappa(\cdot)$ are set to sigmoid and $\tanh$ respectively and apply pointwise to vectors. Fig. 1, shows the schematic diagram of the GRU network with update variants.

A. Additive and Multiplicative Updates

We define the loss function as $\ell_t(d_t, \hat{d}_t) = (d_t - \hat{d}_t)^2$, $\hat{d}_t = \boldsymbol{a}^T\boldsymbol{h}_t$, where $\boldsymbol{a}_t \in \mathbb{R}^q$ is the regression coefficient weight vector. Based on this loss function, i.e., ℓ_t, we evaluate the SGD updates for a given parameter lets say ψ as follows:

$$\psi_{t+1}^{(SGD)} = \psi_t^{(SGD)} + \eta\frac{\partial\ell_t}{\partial\psi} = \psi_t^{(SGD)} - 2(d_t - \hat{d}_t)\boldsymbol{a}^T\frac{\partial\boldsymbol{h}_t}{\partial\psi}.$$

Similarly, we evaluate the element-wise multiplicative updates using the EG algorithm as follows:

$$\psi_{t+1,i}^{(EG)} = \frac{\psi_{t,i}^{(EG)} r_{t,i}}{\sum_j \psi_{t,j}^{(EG)} r_{t,j}}, \tag{6}$$

where $r_{t,i} = \exp\left(-\eta\Big(\frac{\partial\ell_t}{\partial\psi}\Big)_i\right)$.

B. Combination of Updates

In this subsection, we define several variants of the combinations of the additive and multiplicative updates for the corresponding GRU network parameters. We define four updates namely (a) mean, (b) minimum, (c) even-odd and (d) balanced

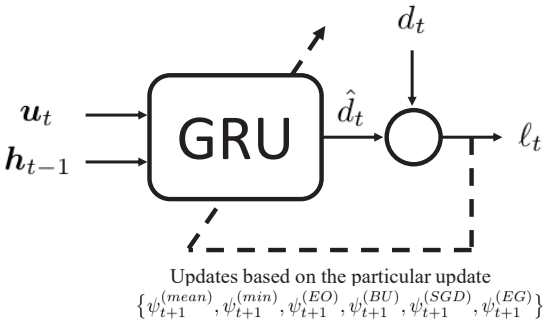

Fig. 1: Schematic diagram of the GRU network with update variants.

updates. For the mean update, in each trial we execute both SGD and EG algorithms and then take their mean to finally update the corresponding network parameters as

$$\psi_{t+1}^{(mean)} \leftarrow \frac{\psi_{t+1}^{(SGD)} + \psi_{t+1}^{(EG)}}{2}. \tag{7}$$

For the minimum update rule, in each epoch we take the minimum update done by both SGD and EG algorithms as

$$\psi_{t+1}^{(min)} \leftarrow \big\{\psi_{t+1}^{(SGD)}, \psi_{t+1}^{(EG)}\big\}. \tag{8}$$

Similarly, for the even-odd update criteria, we update the network parameters using the SGD algorithm for even index of samples and via EG algorithm using odd index of samples as

$$\psi_{t+1}^{(EO)} \leftarrow \mathbb{1}_{(\mathrm{mod}(t)=0)}\psi_{t+1}^{(SGD)} + \mathbb{1}_{(\mathrm{mod}(t)\neq 0)}\psi_{t+1}^{(EG)}, \tag{9}$$

where $\mathbb{1}_{(\phi)}$ is an indicator function that is only applicable when condition ϕ is satisfied. Finally, we carry out balanced updates, where we do SGD updates for first half of the number of samples and EG updates for the next half as

$$\psi_{t+1}^{(BU)} \leftarrow \mathbb{1}_{(t\leq T/2)}\psi_{t+1}^{(SGD)} + \mathbb{1}_{(t>T/2)}\psi_{t+1}^{(EG)}. \tag{10}$$

C. Calculation of Gradient Based Updates

In this subsection, we give the calculations needed for evaluating the updates. In each of the update, we need to evaluate $\dfrac{\partial\boldsymbol{h}_t}{\partial\psi}$. This is calculated as follows:

$$\begin{aligned}\frac{\partial\boldsymbol{h}_t}{\partial\psi} &= \boldsymbol{\Delta}^{((\boldsymbol{1}-\boldsymbol{z}_t)')}\tilde{\boldsymbol{h}}_t + \boldsymbol{\Delta}^{(\boldsymbol{1}-\boldsymbol{z}_t)}\boldsymbol{\Delta}^{(\tilde{\boldsymbol{h}}_t')} + \boldsymbol{\Delta}^{(\boldsymbol{z}_t')}\boldsymbol{h}_{t-1} +\\ &\qquad \boldsymbol{\Delta}^{(\boldsymbol{z}_t)}\boldsymbol{\gamma}_{t-1}^{(\psi)}\\ &= \boldsymbol{\Delta}^{(-\sigma'(\boldsymbol{\zeta}))}\frac{\partial\boldsymbol{\zeta}}{\partial\psi}\tilde{\boldsymbol{h}}_t + \boldsymbol{\Delta}^{(\boldsymbol{1}-\boldsymbol{z}_t)}\boldsymbol{\Delta}^{(\tanh'(\boldsymbol{\xi}))}\frac{\partial\boldsymbol{\xi}}{\partial\psi} +\\ &\qquad \boldsymbol{\Delta}^{(\sigma'(\boldsymbol{\zeta}))}\frac{\partial\boldsymbol{\zeta}}{\partial\psi}\boldsymbol{h}_{t-1} + \boldsymbol{\Delta}^{(\boldsymbol{z}_t)}\boldsymbol{\gamma}_{t-1}^{(\psi)},\end{aligned} \tag{11}$$

where $\boldsymbol{\gamma}_{t-1}^{(\psi)} = \dfrac{\partial\boldsymbol{h}_{t-1}}{\partial\psi}$, $\boldsymbol{\zeta} = \boldsymbol{W}^{(z)}\boldsymbol{u}_t + \boldsymbol{V}^{(z)}\boldsymbol{h}_{t-1}$, $\boldsymbol{\xi} = \boldsymbol{W}^{(h)}\boldsymbol{u}_t + \boldsymbol{V}^{(h)}\boldsymbol{\Delta}^{(\boldsymbol{r}_t)}\boldsymbol{h}_{t-1}$. In (11), for $\psi = w_{ij}^z$, $\left(\dfrac{\partial\boldsymbol{h}_t}{\partial w_{ij}^z}\right)_i$

is its i^{th} component and the corresponding gradient terms in (11) are calculated as follows:

$$\left(\frac{\partial \boldsymbol{\zeta}}{\partial w_{ij}^{z}}\right)_{i} = u_j + v_{ii}^{(z)} \gamma_{t-1,i}^{(w_{ij}^{z})}$$
$$\left(\frac{\partial \boldsymbol{\xi}}{\partial w_{ij}^{z}}\right)_{i} = v_{ii}^{(h)} \gamma_{t-1,i}^{(w_{ij}^{z})} \Big[r_i + h_{t-1,i} v_{ii}^{(r)} \boldsymbol{\Delta}_{ii}^{(\sigma'(\boldsymbol{\nu}))} \Big],$$

where $\boldsymbol{\nu} = \boldsymbol{W}^{(r)} \boldsymbol{u}_t + \boldsymbol{V}^{(r)} \boldsymbol{h}_{t-1}$. Similarly, for $\psi = v_{ij}^{z}$, we have

$$\left(\frac{\partial \boldsymbol{\zeta}}{\partial v_{ij}^{z}}\right)_{i} = h_{t-1,j} + v_{ii}^{(z)} \gamma_{t-1,i}^{(v_{ij}^{z})}$$
$$\left(\frac{\partial \boldsymbol{\xi}}{\partial v_{ij}^{z}}\right)_{i} = v_{ii}^{(h)} \gamma_{t-1,i}^{(v_{ij}^{z})} \Big[r_i + h_{t-1,i} v_{ii}^{(r)} \boldsymbol{\Delta}_{ii}^{(\sigma'(\boldsymbol{\nu}))} \Big].$$

D. Motivation for Trade-off of SGD and EG Updates

Based on the motivation given in [14] with respect to upper and lower bounds for both SGD and EG algorithm, we used the variants of SGD and EG updates. For the individual updates, if for predicting the desired outcome, almost all the input variables are equally related, then SGD algorithm has strong bound over EG algorithm. Similarly, if few inputs are relevant, then EG algorithm has an upper hand over SGD algorithm. Based on these observations made in [14], we used their several variants to handle the relevance of the input variables used for predicting the desired outcome.

III. Experiments

In this section, we illustrate the results of the proposed variants of the combinations of the additive and multiplicative updates. We analyze the regression performance involving all the variants of the updates using various data sets. We first perform experiments using the GRU networks. We use financial data set, i.e., Alcoa Corporation stock price data set [15], Protein Tertiary [16] and Elevators data set [17]. For all the experimental setup, we use a common learning rate of $\mu = 0.1$ for all the additive and multiplicative updates. We then perform the same experiment using the LSTM networks. In Table. I, we show the steady state error performance of all the proposed update variants for all the data sets.

In Fig. 2, we demonstrate the action prediction performance for the Elevators data set. In this data set, based on the various attributes of the F16 aircraft mechanism, we predict the required action to be taken by the aircraft. Each sample of the data set has total 18 features. Based on these 18 unique and distinct features, we efficiently predict the desired action using all the proposed variants of the additive and multiplicative updates. From Fig. 2, we observe that "GRU-Mean" shows the best performance followed by "GRU-Min". These variant performances are much better than achieved by the GRU network with simple additive and multiplicative updates.

Similarly, in Fig. 3, we demonstrate the total error performance for the Alcoa data set. In this experiment, we predict the future stock values based on the past available stock price data. From Fig. 3, we observe that the "GRU-min" updates showed the best performance followed by simple

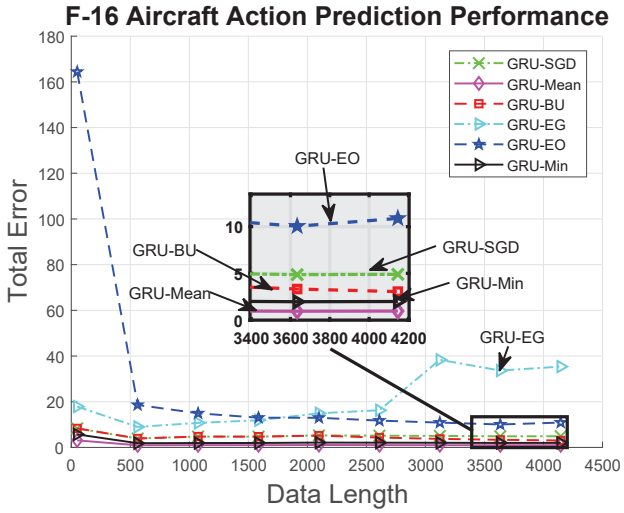

Fig. 2: Action prediction performance of the F-16 aircraft using the GRU network with simple SGD, EG updates and variants of the mixture of additive and multiplicative updates.

"GRU-SGD". After "GRU-SGD," other variants like "GRU-BU" and "GRU-Min" showed promising results. Multiplicative updates were the worst for this case. Overall, most of the update variants depicted good performance as compared with the simple additive and multiplicative update GRU network.

Finally, in Fig. 4, we estimate the leftover or residue size for the Protein Tertiary. The data is collected after several experiments on the critical assessment of structure prediction (CASP). There are total 45730 data samples each with 9 distinct features. From Fig. 4, we observe that "GRU-Mean" shows the best performance as compared to the simple GRU network.

In Table. I, we list down the steady state error performance of all the variants of the combination of additive and multiplicative updates for all the data sets. We observe from the trend in the Table. I that overall the GRU network performs better than the LSTM networks. Among the variants of updates, "GRU-Mean" exhibited the lowest steady state error among the GRU networks for all the data sets. While, for the update variants of the LSTM networks, except Alcoa data set, "LSTM-Mean" showed the best performance. For the Alcoa data set, "LSTM'Min" achieved the minimum steady state error.

Remark 1: *Rather trying the simple additive and multiplicative updates, using the variants of updates as defined in (7)-(10) can further reduce the cumulative and steady-state error. But we cannot generalize which particular variant of the update will result in best performance. So we need to cross-validate first for different data sets.*

IV. Conclusion

We proposed several variants of the mixture of additive and multiplicative updates for the GRU networks. We used four update variants namely GRU-Mean, GRU-Min, GRU-EO (even-odd) and GRU-BU (balanced update) for the GRU network.

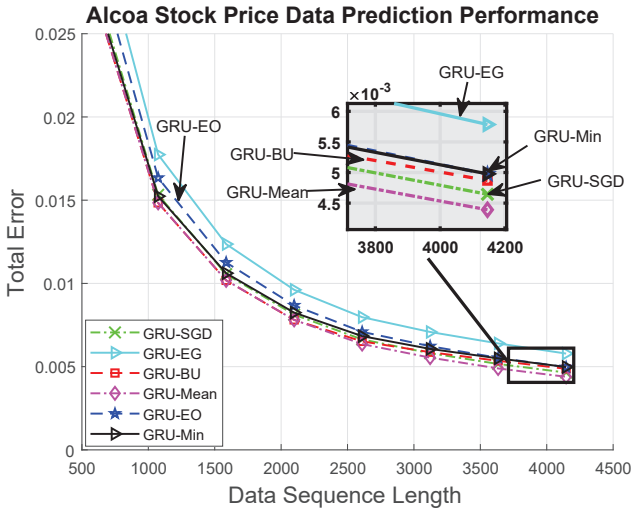

Fig. 3: Stock price prediction performance of the Alcoa Corporation using the GRU network with simple SGD, EG updates and variants of the mixture of additive and multiplicative updates.

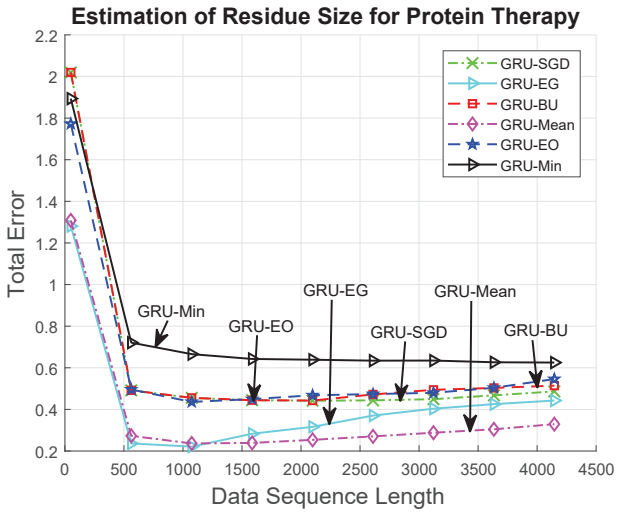

Fig. 4: Estimation of the residue size or leftover for the protein therapy using the GRU network with simple SGD, EG updates and variants of the mixture of additive and multiplicative updates.

We formulate their mathematical expression and derive the gradient-based updates for the parameters of the GRU network. We use financial and real-life data sets to demonstrate the performance of the update variants. From the extensive set of experiments, we observed that the proposed update GRU networks work better than the GRU networks with simple SGD or EG updates. Besides the GRU networks, we also employ these update variants on the LSTM networks. Both the GRU and LSTM network works better on the proposed update variants and among all of the proposed updates, "GRU/LSTM-Mean" showed the best performance for all the data sets.

TABLE I: Steady State Error Performance of the Variants of the Additive and Multiplicative Updates

Data Sets/ Algorithms	Elevators	Alcoa	Protein Tertiary
GRU-SGD	4.8980	0.0046	0.4866
GRU-EG	35.3700	0.0057	0.4427
GRU-BU	3.0510	0.0048	0.5142
GRU-EO	10.8800	0.0049	0.5458
GRU-Mean	**0.9708**	**0.0043**	**0.3292**
GRU-Min	1.9930	0.0049	0.6252
LSTM-SGD	4.8800	0.0048	0.5214
LSTM-EG	11.0210	0.0061	0.4698
LSTM-BU	3.0658	0.0047	0.5222
LSTM-EO	9.9997	0.0041	0.5479
LSTM-Mean	**0.9602**	0.0048	**0.3391**
LSTM-Min	2.5698	**0.0040**	0.6128

REFERENCES

[1] T. Anderson, *The theory and practice of online learning*. Athabasca University Press, 2008.

[2] S. Shalev-Shwartz *et al.*, "Online learning and online convex optimization," *Foundations and Trends® in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2012.

[3] D. P. Mandic, J. A. Chambers, *et al.*, *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. Wiley Online Library, 2001.

[4] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE transactions on signal processing*, vol. 52, no. 8, pp. 2165–2176, 2004.

[5] J. Van Lint, "Online learning solutions for freeway travel time prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 1, pp. 38–47, 2008.

[6] J. Duchi and Y. Singer, "Efficient online and batch learning using forward backward splitting," *Journal of Machine Learning Research*, vol. 10, no. Dec, pp. 2899–2934, 2009.

[7] P. Blunsom, "Hidden markov models," *Lecture notes, August*, vol. 15, pp. 18–19, 2004.

[8] S.-C. Zhu, D. Mumford, *et al.*, "A stochastic grammar of images," *Foundations and Trends® in Computer Graphics and Vision*, vol. 2, no. 4, pp. 259–362, 2007.

[9] L. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, 2001.

[10] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, pp. 1310–1318, 2013.

[11] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[12] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[13] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, pp. 177–186, Springer, 2010.

[14] J. Kivinen and M. K. Warmuth, "Exponentiated gradient versus gradient descent for linear predictors," *Information and Computation*, vol. 132, no. 1, pp. 1–63, 1997.

[15] Alcoa Inc. *Common Stock*. Accessed: Oct. 1, 2016. [Online]. Available: http:// finance.yahoo.com/quote/AA?ltr=1

[16] M. Lichman, "UCI machine learning repository," 2013.

[17] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework.," *Journal of Multiple-Valued Logic & Soft Computing*, vol. 17, 2011.