Online Deep Learning

Learning Deep Neural Networks on the Fly

Jia Zheng

SIST, ShanghaiTech

January 29, 2018

Outline

Outline

Online Classification Task

To learn a classifier from data streams in an online setting.

Batch Learning

1. Batch learning setting requires entire training data to be made available prior to the learning task, this is not possible for many real world tasks where new data arrives sequentially in a stream form, and may be too large to stored in memory.
2. The data may exhibit concept drift.

Existing online learning algorithms

Designed to learn shallow models, e.g., linear or kernel methods with online convex optimization, which can not learn complex nonlinear functions in complicated application scenarios.

Online Deep Learning

A possible way is to put standard Backpropagation (BP) training on only a single instance at each online round, but How to choose a proper model capacity, e.g., the depth of the network, before starting to learn the DNN online.

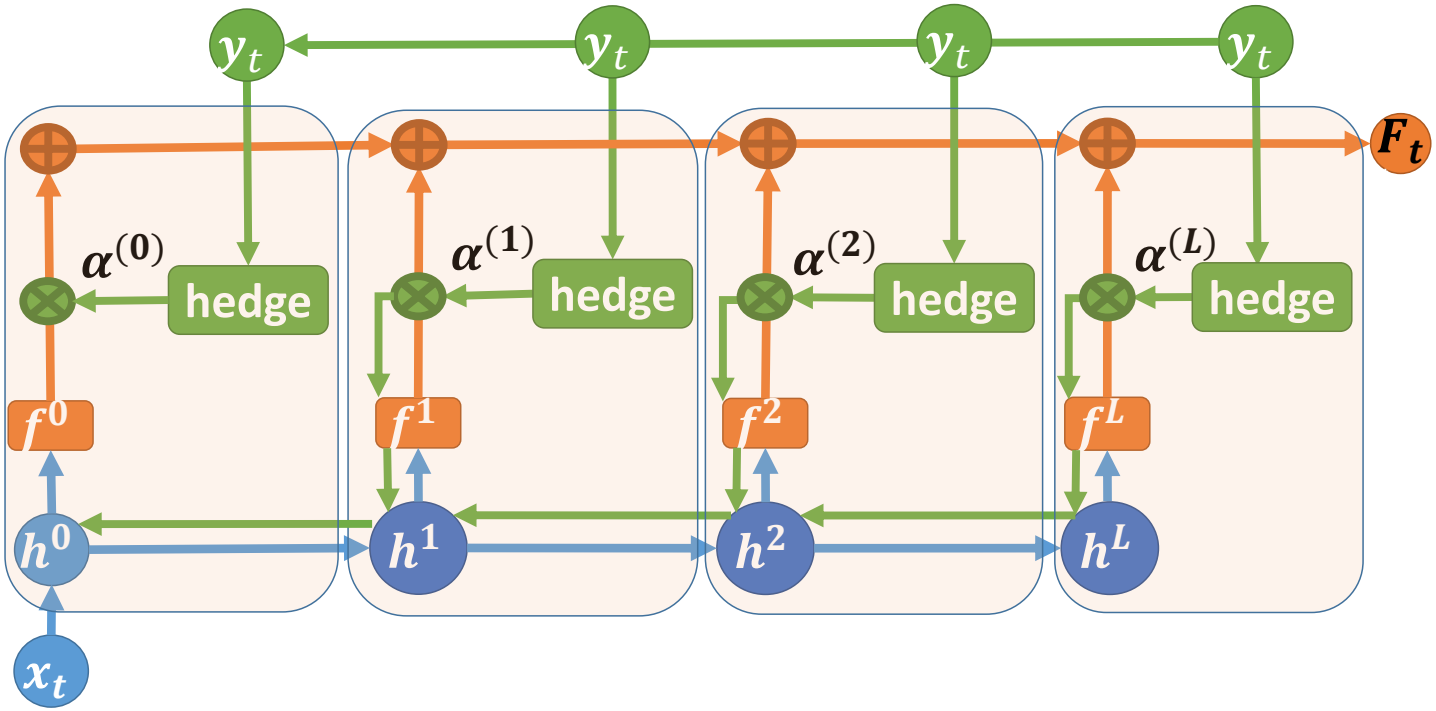
- Too complex: suffer from convergence issues, such as vanishing gradient and diminishing feature reuse
- Too simple: difficult to learn complex patterns
- Infeasible have validation data to do model selection in online setting.

Outline

Hedge Backpropagation

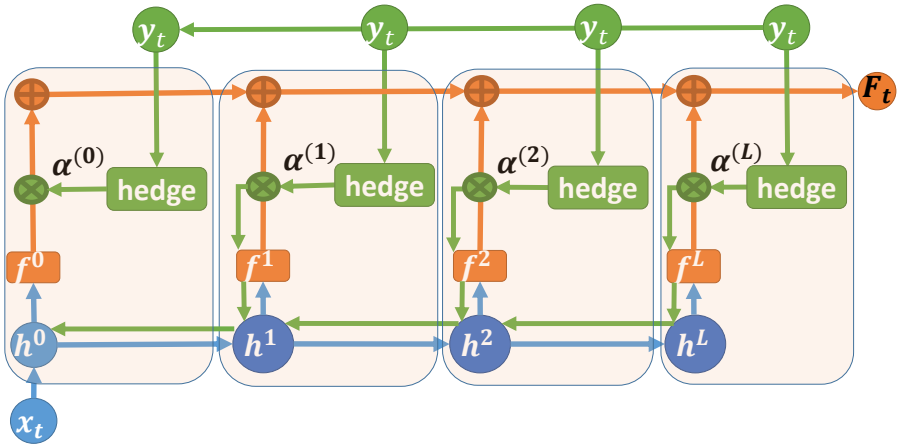
HBP uses an over-complete network, and automatically decides how and when to adapt the depth of network in an online manner.

ODL Framework using HBP



Feedforward , Softmax output, Backpropagation.

Formulation



$$
\begin{aligned}
\mathbf{F}(\mathbf{x}) &= \sum_{l=0}^{L} \alpha^{(l)} \mathbf{f}^{(l)} \quad \text{where} \\
\mathbf{f}^{(l)} &= \mathrm{softmax}(\mathbf{h}^{(l)} \Theta^{(l)}), \ \forall l = 0, \ldots, L \\
\mathbf{h}^{(l)} &= \sigma(W^{(l)} \mathbf{h}^{(l-1)}), \ \forall l = 1, \ldots, L \\
\mathbf{h}^{(0)} &= \mathbf{x}
\end{aligned}
$$

Learn α using Hedge Algorithm

1. At the first iteration, all weights α are uniformly distributed, i.e., $\alpha^{(l)} = \frac{1}{L+1}, l = 0, \ldots, L$.
2. At every iteration, the classifier $\mathbf{f}^{(l)}$ makes a prediction $\hat{y_t}^{(l)}$.
3. The classifier's weight is updated based on the loss suffered by the classifier as:

$$\alpha_{t+1}^{(l)} \leftarrow \alpha_t^{(l)} \beta^{\mathcal{L}(\mathbf{f}^{(l)}(\mathbf{x}),y)}$$

   where where $\beta \in (0, 1)$ is the discount rate parameter.
4. At the end of every round, the weights α are normalized such that $\sum_l \alpha_t^{(l)} = 1$.

α smoothing

Intuition

Shallower models tends to converge faster than deeper models, using the hedging strategy would lower α weights of deeper classifiers to a very small value, which will affect the update, result in deeper classifiers having slow learning.

Smoothing Parameter $s \in (0, 1)$

After the weight update of the classifiers in each iteration, the weights are set as: $\alpha^{(l)} \leftarrow \max\left(\alpha^{(l)}, \frac{s}{L}\right)$.

Discussion

1. HBP can identify DNN of an appropriate depth based on performance of the classifier at that depth
2. Offer a good initialization of DNN
3. Robust to vanishing gradient and diminishing feature reuse by using multi-depth architecture
4. Ensemble of multi-depth networks
5. Enable life-long learning
6. Adapt quickly to concept drifting scenarios due to hedging
7. Adapt to CNN for CV tasks

Outline

Dataset

Table: Datasets

Data	#Features	#Instances	Type
Higgs	28	5m	Stationary
Susy	18	5m	Stationary
i-mnist	784	5m	Stationary
Syn8	50	5m	Stationary
CD1	50	4.5m	Concept Drift
CD2	50	4.5m	Concept Drift

Limitation of traditional Online BP: Difficulty in Depth Selection prior to Training

Table: Online error rate of DNNs of varying Depth.

	Final Cumulative Error			Segment [0-0.5]% Error			Segment [10-15]% Error			Segment [60-80]% Error		
L	**Higgs**	**Susy**	**Syn8**	**Higgs**	**Susy**	**Syn8**	**Higgs**	**Susy**	**Syn8**	**Higgs**	**Susy**	**Syn8**
3	0.2724	0.2016	0.3936	**0.3584**	**0.2152**	**0.4269**	0.2797	**0.2029**	0.4002	0.2668	0.2004	0.3901
4	0.2688	**0.2014**	**0.3920**	0.3721	0.2197	0.4339	**0.2775**	**0.2030**	**0.3989**	0.2617	0.2004	**0.3876**
8	**0.2682**	0.2016	0.3936	0.3808	0.2218	0.4522	0.2794	0.2036	0.4018	**0.2613**	**0.1997**	0.3888
16	0.2731	0.2037	0.4025	0.4550	0.2312	0.4721	0.2831	0.2050	0.4121	0.2642	0.2027	0.3923

- In the first 0.5% of the data, the shallowest network obtains the best performance indicating faster convergence.
- In the segment from [10-15]%, a 4-layer DNN seems to have the best performance in most cases.
- In the segment from [60-80]% of the data, an 8-layer network gives a better performance. This indicates that deeper network took a longer time to converge, but at a later stage gave a better performance.
- The final error doesn't give us conclusive evidence of what depth of network would be the most suitable.

Baseline

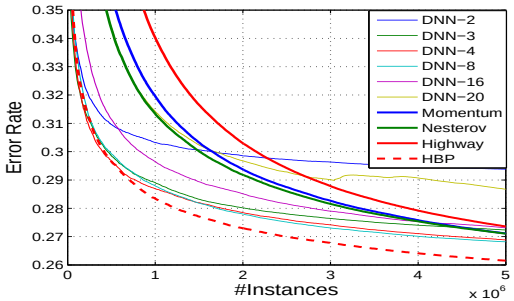
1. Learn a 20 layer DNN in online setting, with 100 units in each hidden layer, using OGD (Online Backpropagation), OGD Momentum, OGD Nesterov, and Highway Networks.
2. Compare the performance of shallower networks with fewer layers (2, 3, 4, 8, 16).
3. Compare with SoTA linear online algorithms (OGD, AROW, SCW), and kernel online learning algorithms (FOGD, NOGD).

Evaluation: Final Online Cumulative Error Rate

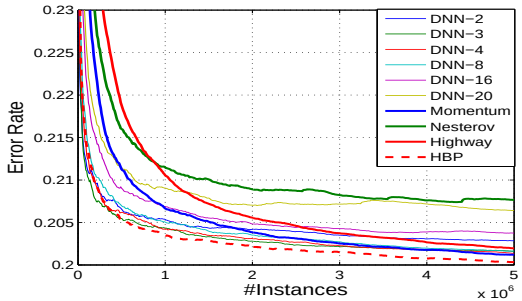
Table: Final Online Cumulative Error Rate obtained by the algorithms.

Model	Method	Layers	Higgs	Susy	i-mnist	Syn8	CD1	CD2
Linear OL	OGD	1	0.3620	0.2160	0.1230	0.4070	0.4360	0.4270
	AROW	1	0.3630	0.2160	0.1250	0.4050	0.4340	0.4250
	SCW	1	0.3530	0.2140	0.1230	0.4050	0.4340	0.4250
Kernel OL	FOGD	2	0.2973	0.2021	0.0495	0.3962	0.4329	0.4193
	NOGD	2	0.3488	0.2045	0.1045	0.4146	0.4455	0.4356
DNNs	OGD (Online BP)	2	0.2938	0.2028	0.0199	0.3976	0.4146	0.3797
	OGD (Online BP)	3	0.2724	0.2016	0.0190	0.3936	0.4115	0.3772
	OGD (Online BP)	4	0.2688	0.2014	0.0196	0.3920	0.4110	0.3766
	OGD (Online BP)	8	0.2682	0.2016	0.0219	0.3936	0.4145	0.3829
	OGD (Online BP)	16	0.2731	0.2037	0.0232	0.4025	0.4204	0.3939
	OGD (Online BP)	20	0.2868	0.2064	0.0274	0.4472	0.4928	0.4925
	OGD+Momentum	20	0.2711	0.2012	0.0310	0.4062	0.4312	0.3897
	OGD+Nesterov	20	0.2711	0.2076	0.0247	0.3942	0.4191	0.3917
	Highway	20	0.2736	0.2019	0.0241	0.4313	0.4928	0.4925
	Hedge BP (proposed)	20	**0.2615**	**0.2003**	**0.0156**	**0.3896**	**0.4079**	**0.3739**

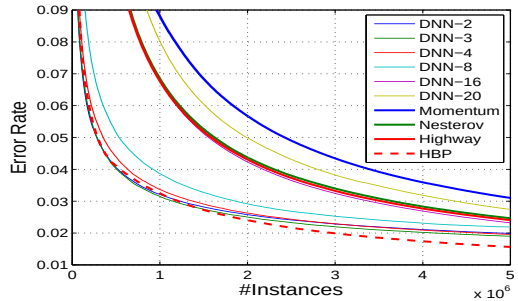
Evaluation: Convergence Rate



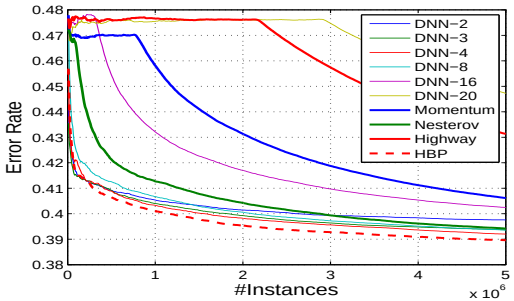
(a) HIGGS



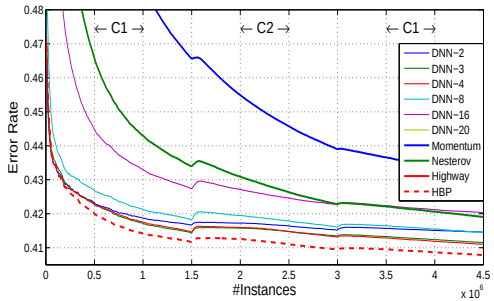
(b) SUSY



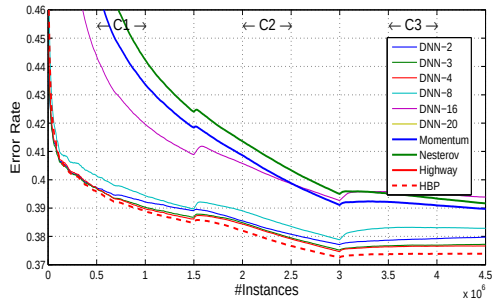
(c) Inf-MNIST (i-mnist)



(d) SYN8



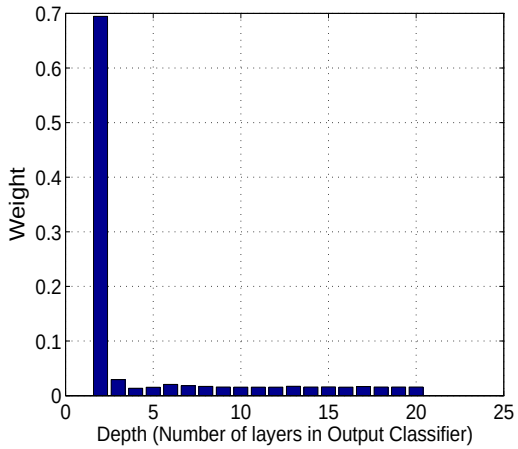
(e) Concept Drift 1 (CD1)



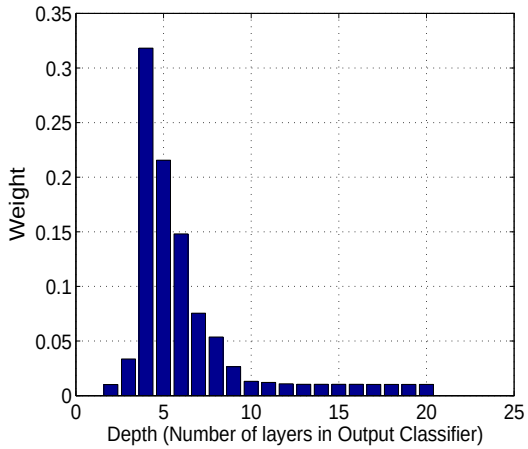
(f) Concept Drift 2 (CD2)

Figure: Convergence behavior of DNNs in Online Setting on stationary and concept drifting data.

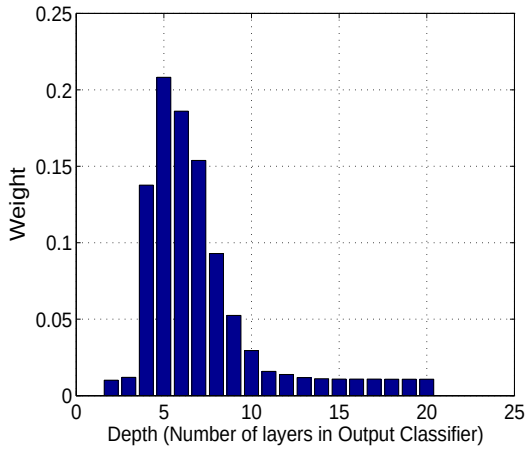
Adapting the Effective Depth of the DNN



(a) First 0.5% of Data



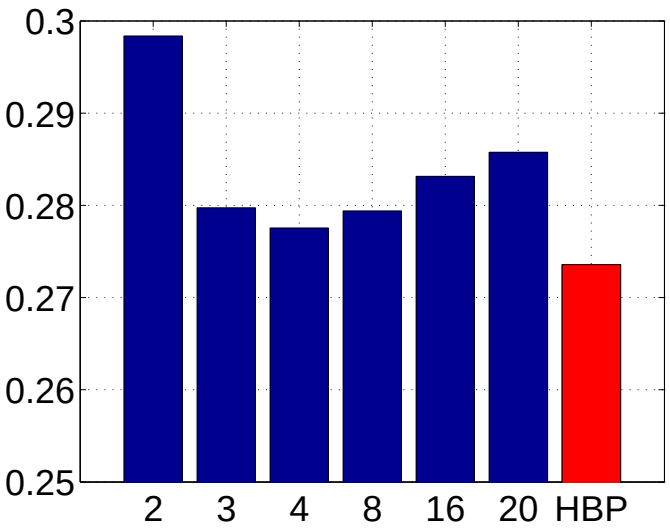
(b) 10-15% of Data



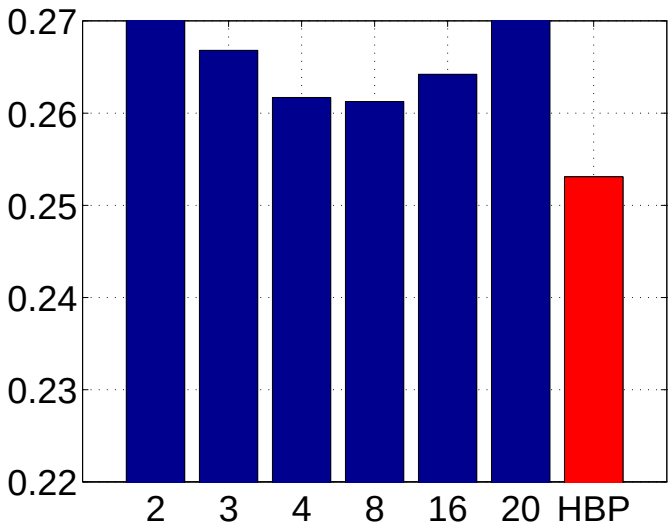
(c) 60-80% of Data

Figure: Evolution of weight distribution of the classifiers over time using HBP on HIGGS dataset.

Performance of Different Learning Stages



(a) Error in 10-15% of data



(b) Error in 60-80% of data

Figure: Error Rate in different segments of the Data. Red represents HBP using a 20-layer network. Blue are OGD using DNN with layers = 2,3,4,8,16 and 20.

Robustness of Depth of Base-Network

Table: Robustness of HBP to depth of the base network compared to traditional DNN

Depth	**12**	**16**	**20**	**30**
Online BP	0.2692	0.2731	0.2868	0.4770
HBP	0.2609	0.2613	0.2615	0.2620

Summary

- Learning DNNs in an online setting.
- Use the "shallow to deep" principle, design Hedge Backpropagation.
- HBP uses a hedging strategy to make predictions with multiple outputs from different hidden layers of the network.
- Automatically identify how and when to modify the effective network capacity in a data-driven manner.

Reference

 Doyen Sahoo et al. "Online Deep Learning: Learning Deep Neural Networks on the Fly". In: *arXiv preprint arXiv:1711.03705* (2017).

Thanks

Thanks for Attention!