# Deep Architectures on Drifting Concepts: A Simple Approach

Leonardo R. Morelli, Pablo M. Granitto, and Guillermo L. Grinblat

CIFASIS – French Argentine International Center for Information and Systems Sciences, AMU (France) / UNR–CONICET (Argentina), Bv. 27 de Febrero 210 bis, Rosario, S2000EZP, Argentina
{granitto,grinblat}@cifasis-conicet.gov.ar
http://www.cifasis-conicet.gov.ar

**Abstract.** Many real-world problems may vary over time. These non stationary problems have been widely studied in the literature, often called drifting concepts problems. Recently, deep architectures have drawn a growing attention, given that they can easily model functions that are hard to approximate with shallow ones and an effective way of training them have been discovered. In this work we adapt a deep architecture to problems that present concept drift. To this end we show a way of combining them with a widely known drifting concept technique, the Streaming Ensemble Algorithm. We evaluate the new method using appropriate drifting problems and compare its performance with a more traditional approach. The results obtained are promising and show that the proposed variation is effective at combining the expressive power of a deep architecture with the adaptability of SEA.

**Keywords:** Drifting concepts, deep architectures

## 1 Introduction

Although most pattern recognition models assume that the data under analysis is independent and identically distributed, many real world problems vary over time. This means that the samples obtained in the past may not be representative of the current version of the problem. In most cases, the underlying causes and characteristics of these changes are not evident from the data. Under such circumstances, it is desirable for the pattern recognition method to be able to adapt to the concept drift as fast as possible giving less importance to older samples, but at the same time harnessing information from these samples when it is still useful.

The methods proposed in the literature assume different kinds of changes in the data. Some are appropriate for abrupt concept changes [1–7]. These methods should detect the change as soon as possible and react in an appropriate way, usually training a traditional model from scratch with only the newer samples. Other models [8–14] are designed with gradual drift in mind, which is the case where the data changes slowly from one distribution to another. In this situation

an interesting problem is how to use efficiently the information from the full dataset. A third class of models tackle the recurrent case, assuming that the data distribution that emerges after a change is probably one already seen [2, 15].

Deep learning attempts to create models composed by several layers of adaptive nonlinear filters, as opposed to traditional shallow models, such as most used artificial neural networks or support vector machines, which are composed by one or two layers of filters. Deep architectures can easily model functions that are hard to approximate with a shallow model. One example is the parity function [16]. A model with less than the required depth may need exponentially more units than a model with the optimal length to approximate these functions.

Despite the theoretical advantages, these architectures have been of limited use because of the practical difficulty to train them. Methods that are efficient for shallow architectures, such as backpropagation, get stuck at poor local minima when used with a deep one. A breakthrough to this problem was presented by Hinton in 2006 [17]. It is based on two stages: In the first one, so called *pretraining stage*, the model is trained greedily one layer at a time with the Contrastive Divergence algorithm [17] for Restricted Boltzmann Machines (RBM, [18]). In the second stage the model is trained by gradient descent. After this discovery, these architectures have drawn the growing interest of the research community.

In this work we present a method that integrates a deep model with a non stationary problem technique, thus allowing the approximation of non stationary functions that are hard for a shallow model. Moreover, deep architectures can capture more useful information from out-of-distribution samples than shallow ones [19]. This suggests that out-of-date samples, obtained before the concept drift, can be useful for training a deep model despite they do not represent the current concept. Our model takes this into account.

This paper is organized as follows. In the next section we discuss previous works on concept drift and deep architectures. In Section 3 we introduce our new method. Then, in Section 4 we present some empirical results and, finally, Section 5 closes the work with some conclusions.

## 2 Previous Work

Drifting concepts are specific classification problems in which the labels of examples or the shape of the decision boundaries change over time. Kolter et al. [20] include a lengthy review of the state of the art in the field, starting from the early work of Schlimmer and Granger Jr. [21].

There are two main approaches to concept drift in the literature: sample selection or weighting and ensemble methods [22, 23]. We will start discussing the sample selection or weighting methods.

The most common solution to the drifting concepts problem is to use a temporal window of a given length and to build a different classifier (or adapt a previous one) for each window [1–3, 24–26]. This brings up the problem of window size: a window too big will make the algorithm to have a slow reaction to change, and a window too small will make it too sensitive to noise in the data. As

a potential solution, many algorithms include an adaptive window size [2,24,25]. Sample weighting methods can be viewed as a softening of the temporal window strategy, which gives "hard" (0/1) weights to (older/newer) examples. Several methods have been proposed for this, as using a simple linear decreasing function for the relative weight [27] or an exponentially decaying function [28].

When assuming a certain kind of change, more specific algorithms can be developed. In the case of abrupt changes, Castillo et al. [29] and also Lanquillon [26] use statistical quality control to determine if there is a concept change in a given batch. When this happens, a new classifier is constructed from scratch using only the samples considered to belong to the new context. This can be viewed as an adaptive windows size policy: the temporal window steadily grows until a change is detected, in which case it abruptly shrinks. Other tests have been proposed for change detection [6, 7, 30].

Regarding the ensamble methods, several authors discussed its use for drifting concepts, proposing algorithms based on fixed or variable size ensembles of weighted or unweighted experts. The SEA algorithm [4], in which this work is based, maintains a fixed-size ensemble using a heuristic replacement strategy. Wang et al. [31] used a similar strategy, adding weights to each member of the ensemble according to a measure of its performance. Kolter and Maloof [20,32] introduced an ensemble method based on a previous work of Littlestone et al. [33]. Their algorithm dynamically creates and removes weighted experts in response to changes in performance.

After Hinton's seminal work on deep network training [17], a great research effort has been done in several directions. New training algorithms have been proposed [34–36]. Erhan et al. [37] conducted an extensive experimental study to determine whether the effect of the pretraining stage could be viewed as a helping optimization or having a regularization effect. Bengio and coworkers [19] proved that deep architectures can benefit from out-of-distribution samples, relating the area with self-taught learning [38]. Other works introduced these techniques to semi-supervised tasks [39].

A good survey of the area can be found in [16]. It is worth noting that, to ours knowledge, there only has been one preliminary attempt to adapt a deep network to the drifting concept scenario [40]. In that work the authors propose to generate new samples with the previous model in order to have more training samples when a new data batch arrives.

In the following subsections we will introduce the algorithms used in this work.

### 2.1 The SEA Algorithm

SEA creates an ensemble from a data sequence divided in batches. The algorithm, depicted in Figure 1, is not too complex. Each time a new batch arrives, a new classifier is trained with it. In the original paper this classifier was a decision tree, but the algorithm can be used with any appropriate model. The batch is then used to obtain a quality measure for the classifier corresponding to the previous batch, based on its performance and diversity. This quality measure is

also calculated for each member of the ensemble. Finally, if the new classifier has a better quality than any of the members of the ensemble, the one with the poorest quality is replaced.

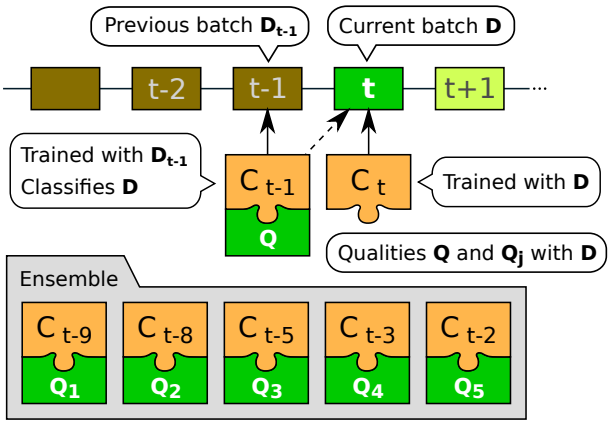


**Fig. 1.** The SEA algorithm. When batch $D$ arrives a new classifier, $C_t$, is created. The qualities of the previous classifier, $C_{t-1}$, and the classifiers of the ensemble are evaluated with D and the worst one is discarded.

### 2.2 Stacked Restricted Boltzmann Machines

Hinton et al. [17] showed how to train deep models for different problems. For classification problems in particular, the pretrain stage is used to initialize the weights of a feedforward net, which are then finetuned with backpropagation. The models trained in this way are usually called *Stacked Restricted Boltzmann Machines* (SRBM) [35]. In the following paragraphs we will give a more detailed introduction to them.

A Restricted Boltzmann Machine is a generative model, with parameters $W$, matrix and $a$ and $b$, vectors, based on a bilinear energy function $E$ given by

$$E(v, h) = -a^T v - b^T h - h^T W v,$$

where $v$ represent a vector of visible variables and $h$ a vector of hidden variables. The probability of a given configuration of visible and hidden variables $(v, h)$ is

$$p(v, h) = \frac{e^{E(v,h)}}{Z},$$

where $Z$ is the *partition function*. The probability of a given visible configuration can be obtained by marginalizing out the hidden variables $h$.

This model can be trained by minimizing the log-likelihood over the training set with gradient descent. To obtain the gradient we need to calculate two

expectations, one is easily computed but for the other a sum over all posible configurations of $v$ and $h$ is needed. To solve this, a stachastic estimator is used, sampling $v$ and $h$ according to the probability $p(v, h)$ assigned by the model. This can be obtained running a Gibbs sampling chain until convergence. To reduce this cost even further a second approximation is used, the Contrastive Divergence algorithm. It runs the Gibbs chain for only $k$ steps (CD-$k$). In practice, using $k = 1$ gives good results.

We can obtain a deep net simply by stacking several RBMs. The first one is trained over the training set, and $p(h|x)$ is calculated for each sample $x$. Then a second RBM is trained over these probabilities. The process continues until we have the desired depth. It is worth noting that the training of all the RBMs can be done simultaneously [35].

After this stage, for classification problems a classifier (for example, a linear classifier) is added to the top of the stack. The visible units biases (i.e. the $a$ parameters of each RBM) are discarded and the whole net is fine-tuned by backpropagation.

## 3 The algorithm

One difficulty of directly using these models with SEA is that if each data batch is small enough to let the algorithm react quickly to changes, the batch will have very few examples to train a deep network.

To solve this, and taking into account [19], we assume that the features obtained in the lower layers of the net will remain useful even when there is a concept change. Because of that we will have only one net pretrained with all the previous data and take into account the non-stationarity on the fine-tuning stage.

The algorithm is depicted in Figure 2. We maintain a base net, that is updated each time a new batch arrives, applying a fixed number of iterations of CD to each layer one after the other (step 1 in the figure). In order to create a new expert, we copy this base net (step 2) and fine-tune the copy using the new batch (step 3). Once the new expert is trained, SEA continues with its usual execution.

## 4 Empirical Evaluation

In order to evaluate the performance of the proposed method, we conducted several experiments on a group of non stationary problems based on the MNIST dataset [41], a widely used example in the deep learning literature.

MNIST is a dataset of images of handwritten digits. It has 70000 samples, already separated in 60000 training samples and 10000 test samples. The original images were black and white (only two levels). The digits were scaled, maintaining their aspect ratio, and adjusted to a square of 20x20 pixels. The normalization algorithm used an anti-aliasing technique and, as a result, the final images contain gray levels. Finally, the images were centered in a 28x28 pixels square.

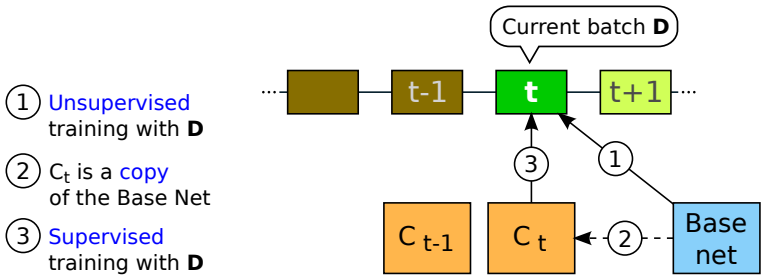


**Fig. 2.** The proposed algorithm. When batch $D$ arrives, first the base net is updated, then a copy is fine tuned with $D$. Following that, SEA evaluates the qualities of the previous classifier $C_{t-1}$ and the classifiers of the ensemble (not shown).

We generated 6 non stationary problems. All are classification problems, where the target is to separate odd from even digits. We separated the training samples in three sets, $D_0$, $D_1$, and $D_2$ composed respectively by digits $\{0, 1, 2\}$, $\{3, 4, 5\}$ and $\{6, 7, 8, 9\}$. The *Abrupt* dataset was generated sampling first from $D_0$, from $D_1$ when we ran out of samples, and finally from $D_2$. In the *RecurrentAbrupt* dataset there are also recurrent changes. As in the previous dataset, we sample from $D_0$, $D_1$ and $D_2$ in order, but we save some samples in order to revisit each distribution once. In the *Gradual* dataset the changes are not abrupt, i.e. there are periods of time where two distributions coexist.

Figure 3 shows a representation of these three datasets through time. In it we see, for any given time, the probability of sampling from $D_0$, $D_1$ and $D_2$. The times $t_n$ mark the abrupt changes present in the first two datasets and the beginning and ending of the gradual changes in the last one.

Finally, another three datasets were generated in the same way, changing only the composition of $D_0$, $D_1$ and $D_2$ to $\{0, 1, 8\}$, $\{3, 4, 6, 7\}$ and $\{2, 5, 9\}$. In this way, similar looking digits (0 and 6, 8 and 3, 1 and 7, etc.) appear in consecutive distributions. These are the *Abrupt2*, *RecurrentAbrupt2* and *Gradual2* datasets.

We compared the performance obtained in these problems by the combination of SEA and SRBM, with the results of SEA with Support Vector Machines (SVM) as base classifiers, using a Gaussian kernel [42]. Some initial tests were performed also with SEA and decision trees, but, given the superior performance of the SEA + SVM solutions over the SEA + decision tree in these problems, the main tests were conducted only with SEA + SRBM and SEA + SVM.

A disadvantage that deep architectures have over traditional methods is that they need, in general, much more time to train. This time can be reduced using more powerful hardware, for example using GPUs to train the network [43], but the disadvantage is still there. Moreover, there are more hiperparameters to optimize in an SRBM than in a SVM. We have taken into account the training time needed for both methods, making a more thorough search for optimum hiperparameters for the SEA + SVM models than for the SEA + SRBM. The hiperparameters search was conducted with cross validation over the Abrupt

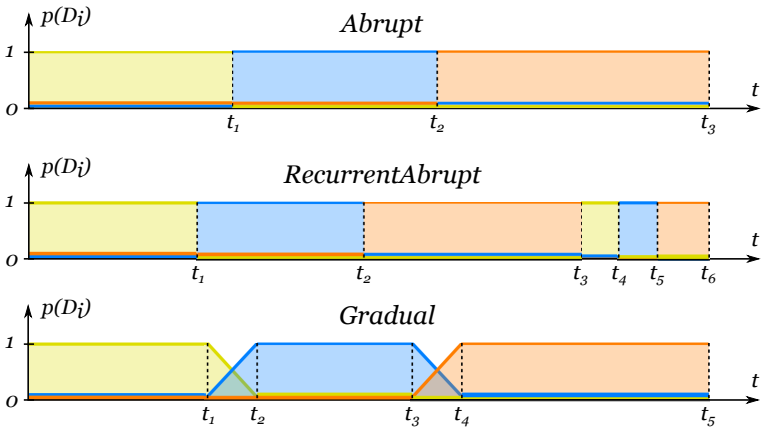


**Fig. 3.** The three non stationary problems used in this work. $D_0$, $D_1$ and $D_2$ are depicted respectively in yellow, blue and orange. The lines show the probabilities at each time of sampling from either $D_0$, $D_1$ or $D_2$.

dataset, using the optimum values found there in the other five cases. Table 1 shows the hiperparameters used in the tests.

| | Hiperparameter | Value |
|---|---|---|
| SEA + SRBM | SEA Batch size | 40 |
| | SEA Ensemble size | 5 |
| | SRBM Architecture | $784 \times 1000 \times 500 \times 500 \times 2000 \times 2$ |
| | SRBM Learning rate | 0.1 |
| | SRBM Training epochs | 10 |
| | SRBM Mini-batch size | 5 |
| SEA + SVM | SEA Batch size | 40 |
| | SEA Ensemble size | 5 |
| | SVM C | 6 |
| | SVM Gamma | 0.013 |

**Table 1.** Hiperparameters used in the tests.

Table 2 shows the test results. In it we see the accuracy of both methods for the complete datasets. The results are the mean value and standard deviation on 30 trials.

The Figures 4, 5 and 6 show the accuracy of both methods over time in the Abrupt, RecurrentAbrupt and Gradual datasets, respectively. At each point the lines show the mean value for 10 consecutive batches, that is 400 samples, excepting the periods where there is a change or the models are adapting, where each point corresponds to one batch in order to better appreciate the evolution in those periods. The other three datasets show a similar behavior.

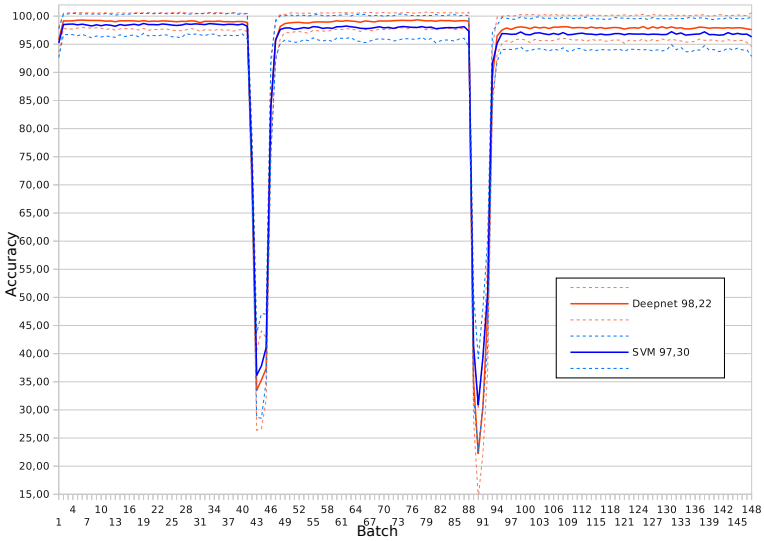


**Fig. 4.** The variation of the accuracy over time for the two methods on the Abrupt dataset. The mean over 30 trials is plotted (continuous line) ± its standard deviation (dashed lines).

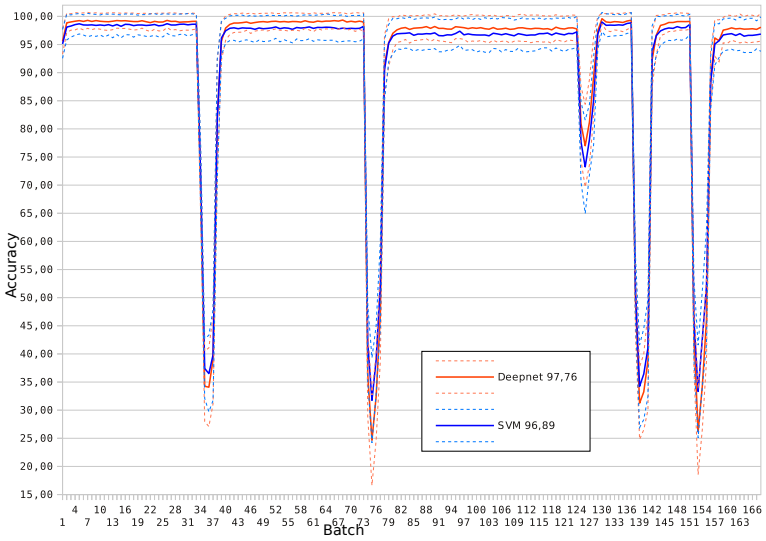


**Fig. 5.** The variation of the accuracy over time for the two methods on the RecurrentAbrupt dataset. The mean over 30 trials is plotted (continuous line) ± its standard deviation (dashed lines).

| Dataset | Accuracy % ($\pm\sigma$) | |
|---|---|---|
| | SEA + SRBM | SEA + SVM |
| Abrupt | 98.22 (0.05) | 97.30 (0.06) |
| Abrupt2 | 98.22 (0.04) | 96.92 (0.05) |
| RecurrentAbrupt | 97.75 (0.05) | 96.89 (0.07) |
| RecurrentAbrupt2 | 97.80 (0.06) | 96.52 (0.05) |
| Gradual | 98.16 (0.05) | 97.18 (0.06) |
| Gradual2 | 98.16 (0.07) | 96.84 (0.07) |

**Table 2.** The results of all tests. The columns show, for each method and dataset, the mean accuracy over 30 trials and its standard deviation.

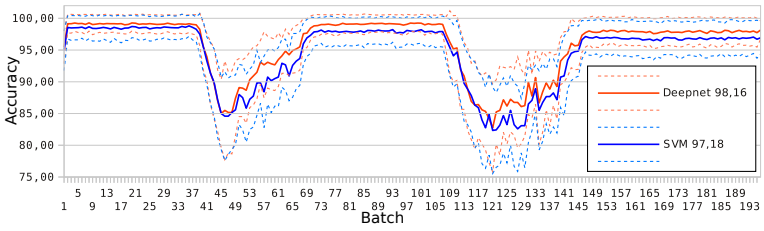


**Fig. 6.** The variation of the accuracy over time for the two methods on the Gradual dataset. The mean over 30 trials is plotted (continuous line) ± its standard deviation (dashed lines).

In the results we can see a noticeable superiority of the SEA + SRBM model over the SEA + SVM one. This is expected, since deep architectures show better results in the original MNIST problem. As can be seen, the utilization of previous points was beneficial, since with only the 40 samples in each batch would not be possible to train an SRBM.

The recovery after changes is given by the adaptive capacity of SEA, using SRBM or SVM as the base algorithm. The final performance of the ensemble when the data distribution is stabilized (between changes) is achieved only when all the experts created before the change are replaced by new ones.

In Figure 4 we can see a small performance degradation after the first and second abrupt change. This is related to the particular composition of each $D_i$ and the number of digits each one has (3 or 4). This makes some distributions a little harder than others. In Figure 5 we can see that when a distribution appears for a second time (recurrence) the performance is similar to the first time, which would not be the case if the degradation is due to some difficulty of the algorithm to cope with change.

## 5 Conclusions and Future Work

In this work we presented an adaptation of a method based on a deep architecture to the non stationary case. It was evaluated using non stationary datasets based on a widely tested real-world application, the MNIST problem.

The results obtained are promising. A classifier used by SEA is trained only with a very small fraction of the already seen data, this is one of the causes of its adaptability. Although a typical deep network needs a large amount of data to adjust its huge number of parameters (the networks used in our tests have around 2.5 million of them), which makes them unusable with SEA, the results show that the proposed variation is effective at combining the expressive power of a deep architecture with the adaptability of SEA.

Two lines of research can be pursued in the future. First, to study the use of techniques that allow an early detection of an abrupt change in order to give more weight to the classifiers created after the change. This would allow a faster response to an abrupt change. Second, the use of a more sophisticated non stationary method, such as Dynamic Weighted Majority, can be investigated. DWM uses a dynamic sized ensemble and each expert has a different weight, but it retrains every expert each time a new data arrives, which is prohibitive with SRBMs. A way of circumventing this problem is needed.

## Acknowledgements

We acknowledge partial support from ANPCyT.

## References

1. Mitchell, T., Caruana, R., Freitag, D., McDermott, J., Zabowski, D.: Experience with a Learning Personal Assistant. Communications of the ACM 37, 81–91 (1994)
2. Widmer, G., Kubat, M.: Learning in the Presence of Concept Drift and Hidden Contexts. Machine Learning 23, 69–101 (1996)
3. Salganicoff, M.: Tolerating Concept and Sampling Shift in Lazy Learning Using Prediction Error Context Switching. Artificial Intelligence Reviews 11 (1-5), 133–155 (1997)
4. Street, W.N., Kim, Y.: A Streaming Ensemble Algorithm (SEA) for Large-Scale Classification. In: 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 377–382. ACM Press, New York (2001)
5. Bach, S.H., Maloof, M.A.: Paired Learners for Concept Drift. In: 8th IEEE International Conference on Data Mining (ICDM 2008), pp. 23–32. IEEE Computer Society (2008)
6. Alippi, C., Roveri, M.: Just-in-Time Adaptive Classifiers–Part I: Detecting Nonstationary Changes. IEEE Transactions on Neural Networks 19 (7), 1145–1153 (2008)
7. Alippi, C., Roveri, M.: Just-in-Time Adaptive Classifiers–Part II: Designing the Classifier. IEEE Transactions on Neural Networks 19 (12), 2053–2064 (2008)
8. Helmbold, D.P., Long, P.M.: Tracking Drifting Concepts by Minimizing Disagreements. Machine Learning 14, 27–45 (1994)
9. Bartlett, P.L.: Learning with a Slowly Changing Distribution. In: 5th Annual Workshop on Computational Learning Theory (COLT '92), pp. 243–252. ACM, New York (1992)
10. Bartlett, P.L., Ben-Dabid, S., Kulkarni, S.R.: Learning Changing Concepts by Exploiting the Structure of Change. Machine Learning 41 (2), 153–174 (2000)

11. Barve, R.D., Long, P.M.: On the Complexity of Learning from Drifting Distributions. In: Workshop on Computational Learning Theory, pp. 170–193. Morgan Kaufmann Publishers (1996)
12. Freund, Y., Mansour, Y.: Learning Under Persistent Drift. In: 3rd European Conference on Computational Learning Theory (EuroCOLT '97), pp. 109–118. Springer-Verlag, London (1997)
13. Alippi, C., Boracchi, G., Roveri, M.: Just-in-Time Classifiers: Managing the Slow Drift Case. In: International Joint Conference on Neural Networks (IJCNN 2009), pp. 114–120. IEEE (2009)
14. Grinblat, G.L., Uzal, L.C., Ceccatto, H.A, Granitto, P.M.: Solving Non-Stationary Classification Problems with Coupled Support Vector Machines. IEEE Transactions on Neural Networks 22 (1), 37–51 (2011)
15. Koychev, I.: Tracking Changing User Interests Through Prior-Learning of Context. In: AH 2002. LNCS, vol. 2347, pp. 223–232. Springer, Heidelberg (2002)
16. Bengio, Y.: Learning Deep Architectures for AI. Foundations and Trends in Machine Learning 1 (2), 1–127 (2009)
17. Hinton, G. and Salakhutdinov, R.: Reducing the Dimensionality of Data with Neural Networks. Science 5786 (313), 504–507 (2006)
18. Smolensky, P.: Information Processing in Dynamical Systems: Foundations of Harmony Theory. Parallel Distributed Processing 1: Foundations, 194–281, MIT Press, Cambridge, MA (1986)
19. Bengio, Y. and others: Deep Learners Benefit More from Out-of-Distribution Examples. In: JMLR Workshop and Conference Proceedings Volume 15: AISTATS 2011, pp. 164–172. MIT Press, Cambridge, MA (2011)
20. Kolter, J.Z., Maloof, M.A.: Dynamic–Weighted Majority: An Ensemble Method for Drifting Concepts. Journal of Machine Learning Research 8, 2755–2790 (2007)
21. Schlimmer, J.C., Granger Jr., R.H.: Incremental Learning from Noisy Data. Machine Learning 1, 317–354 (1986)
22. Stanley, K.O.: Learning Concept Drift with a Committee of Decision Trees. Department of Computer Sciences, University of Texas at Austin, Tech. Rep. UTAI-TR-03-302 (2003)
23. Tsymbal, A.: The Problem of Concept Drift: Definitions and Related Work. Computer Science Department, Trinity College Dublin, Tech. Rep. TCD-CS-2004-15 (2004)
24. Klinkenberg, R., Renz, I.: Adaptive Information Filtering: Learning in the Presence of Concept Drifts. In: Workshop Notes of the ICML/AAAI-98 Workshop Learning for Text Categorization, pp. 33–40. AAAI Press (1998)
25. Klinkenberg, R., Joachims, T.: Detecting Concept Drift with Support Vector Machines. In: 17th International Conference on Machine Learning, pp. 487–494. Morgan Kaufmann Publishers (2000)
26. Lanquillon, C.: Enhancing Test Classification to Improve Information Filtering, PhD Thesis. Faculty of Computer Science, University of Madgdeburg, Germany (2001)
27. Koychev, I.: Gradual Forgetting for Adaptation to Concept Drift. In: Workshop Current Issues in Spatio-Temporal Reasoning (ECAI 2000), pp. 101–106. IOS Press (2000)
28. Klinkenberg, R.: Learning Drifting Concepts: Example Selection vs. Example Weighting. Intelligent Data Analysis 8, 281–300 (2004)
29. Castillo, G., Gama, J., Medas. P.: Adaptation to Drifting Concepts. In: EPIA 2003. LNCS, vol. 2902, pp. 279–293. Springer, Heidelberg (2003)

30. Koychev, I., Lothian, R.: Tracking Drifting Concepts by Time Window Optimization. In: 25th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, pp. 46–59. Springer (2005)
31. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining Concept-Drifting Data Streams Using Ensemble Classifiers. In: 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 226–235. ACM Press, New York (2003)
32. Kolter, Z., Maloof, M.A.: Dynamic Weighted Majority: A New Ensemble Method for Tracking Concept Drift. In: IEEE International Conference on Data Mining (ICDM 2003), pp. 123–130. IEEE Computer Society (2003)
33. Littlestone, N., Warmuth, M.K.: The Weighted Majority Algorithm. Information and Computation 108 (2), 212–261 (1994)
34. Salakhutdinov, R.: Learning Deep Boltzmann Machines Using Adaptive MCMC. In: 27th International Conference on Machine Learning, pp. 943–950. Omnipress (2010)
35. Larochelle, H., Bengio, Y., Louradour, J., Lamblin, P.: Exploring Strategies for Training Deep Neural Networks. Journal of Machine Learning Research 10, 1–40 (2009)
36. Adams, R., Wallach, H., Ghahramani, Z.: Learning the Structure of Deep Sparse Graphical Models. In: 13th International Workshop on Artificial Intelligence and Statistics, pp. 1–8 (2010)
37. Erhan, D., Bengio, Y., Courville, A.C., Manzagol, P.A., Vincent, P., Bengio, S.: Why Does Unsupervised Pre-training Help Deep Learning?. Journal of Machine Learning Research 11, 625–660 (2010)
38. Raina, R., Battle, A., Lee, H., Packer, B., Ng, A.Y.: Self-taught Learning: Transfer Learning from Unlabeled Data. In: 24th International Conference on Machine Learning, pp. 759–766. ACM (2007)
39. Weston, J., Ratle, F., Collobert, R.: Deep Learning via Semi-supervised Embedding. In: 25th International Conference on Machine Learning, pp. 1168–1176. ACM (2008)
40. Calandra, R., Raiko, T., Deisenroth, M. P., Montesino Pouzols, F.: Learning Deep Belief Networks from Non-Stationary Streams. In: Lecture Notes in Computer Science 7553, Part 2, 379–386 (2012)
41. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-Based Learning Applied to Document Recognition. In: Proceedings of the IEEE 86 (11), pp. 2278–2324 (1998)
42. Vapnik, V. The nature of statistical learning theory. New York: Springer (1995).
43. Raina, R., Madhavan, A., Ng, A. Y.: Large-Scale Deep Unsupervised Learning Using Graphics Processors. In: 26th International Conference on Machine Learning, pp. 873–880. ACM (2009)