A Network of Layered, Localized Multi-scale Decision Boundary Components for Gradual Pattern Drift Adaptation

The problem of training a classifier using a static set of patterns for training data, even using good feature selection, hyperparameter search and cross-validation methods to avoid over-fitting while optimizing generalization performance, is exacerbated if the patterns from the classes of interest are not, in practice, static. This problem is mainly treated in the literature on "concept drift", and is seen in such disparate applications as ECG monitoring in the ICU (patterns of the ECG of "normal" beats drift over a long period of patient monitoring), the evolving interests of a consumer in different types of film or music, or in the problem of discrimination of threats from non-threats in aircraft protection systems where the aircraft may be operating in gradually changing atmospheric conditions. It is easy to imagine scenarios that would make it difficult for almost any system trained on a static set of data to continue to provide high accuracy classification for data which is drifting gradually with changing operating conditions or distributions of data in the field.

Primary features of the proposed approach:

- The use of multi-scale components to compose / approximate the decision boundary
- The components are discriminative, focusing on the decision boundary regions
- A hierarchical approach, from coarse to fine scale, to gradual pattern drift adaptation
- An assumption that the details of decision boundaries (at finer scales) drift in concert with the coarser scale components of decision boundaries, but also exhibit some independence
- The use of a network comprised of localized linear discriminants which are trained and updated incrementally, using a large margin / soft margin perceptron training technique
- Generation of an ensemble of classifiers over scale

The original inspiration:

To illustrate, in more detail, the potential for a multi-scale method to handle drift is the example of drift in formant frequencies of vowels in an individual's speech as they grow from child to adult. All vowels will drift slowly over the years with vocal tract size, and the necessity of producing recognizable speech will constrain the margins between vowels to be maintained (and therefore drift roughly in sync with the vowel clusters). This would be an example of a coarse scale drift (all vowels moving together in F1/F2 space) with individual fine scale drift for each vowel cluster that is roughly tied to the overall drift, but also specific in detail to each vowel. Likewise the between-class (vowel) margins will drift roughly with the global drift, but shift, rotate, and/or warp at finer scales as the shape of the vocal tract changes (the vocal tract will not just simply scale with age), in combination with the fundamental frequency (F0) which will also change in complex ways (length and tension of the vocal folds, glottal pressure, etc.). Since formant frequency is really an estimate of the centers of gravity of spectral peaks formed by the convolution of the fundamental with the vocal tract resonances, it is also unlikely that the pattens of variation with repetition, which form the between class boundary regions in F1 / F2 space, will simply scale with age. See [3] and [17] for a discussion of these changes in terms of average formant frequencies and the effect of F0 harmonic spacing on formant estimation. In particular, see figure 27 [17] for average F1/F2 values, and figures 22 – 26 [17] for formant variance with age.

Since the global drift of clusters of data over time (with vocal tract size), along with associated changes (shape and articulatory patterns) affecting the detailed shapes of individual clusters, an optimized classifier could be viewed as one having a multi-scale decomposition of the decision boundary that is flexible enough to adapt to gradual drift over time by adjusting the size, orientation, and position of each component without changing the basic structure of the network. It would also be easy to visualize an analogous scenario involving the size and relationship of facial features as they change from infancy to adulthood. Of course the time scales we are actually concerned with here start with the time it takes, say, for ECG morphology to drift, or atmospheric conditions to change over the course of a flight.

A quick look at some relevant literature:

Interest in concept (pattern) drift has grown recently due to the applications in data mining, "big data", and customized online services. One reason for this interest is that beyond the issue of adaptation to drifting patterns, much of the new data available is unlabeled, making the problem that much more difficult, and the importance of algorithms which can effectively continue to learn under these conditions an important area for research. Jerome Friedman from Stanford gave a talk at MIT CSAIL entitled “Estimating Class Counts in Unlabeled Data” in 2014 [20], discussing this problem. Pechenizkiyl, Žliobaitė, Bifet, and Bouchachia state that "Over the last decade research related to learning with concept drift has been increasingly growing and many drift-aware adaptive learning algorithms have been developed" [1]. Despite this recent activity, the author is not aware of any "drift aware" adaptive learning algorithms which are structured with a multi-scale view of the data. Related algorithms do exist which deal with the underlying problem of scale through a different perspective, through adjustment of the duration of windows in time over which data for classifier re-training / updating is utilized. One example is described in [18], but this algorithm, as are many of those that deal with temporal scale, are concerned with supervised learning. While there exist a number of algorithms designed to build decision boundaries based on a piecewise linear elements, and these may in fact develop over multiple scales, there is only one algorithm that has been identified which is explicitly designed to utilize multi-scale components in developing decision boundaries ("Multi-Scale Kernel Methods for Classification" [13]).To date, there do not appear to be any algorithms which explicitly decompose class distributions and decision boundaries in order to treat concept drift in a differential way with regard to scale in feature space.

We then assume that it is important, for a classifier which is designed to be adaptive to drifting patterns, to follow not only the movement of data within clusters at a global level, but the details of the data at the edges of the distribution as well. This is where the concept of multi-scale approximations of cluster boundaries is applicable. In the case of classification, decision boundaries could be viewed as composable from smooth, piecewise local discriminant functions at multiple scales. In this proposal, the hidden units in this new architecture are localized linear discriminants (LLD's, see [5]). Apologies to the reader for the repetitive use of a non-standard acronym, but it's in the original paper, seems to be a pretty good descriptor of the nature of this hidden unit, and yields the same space efficiency as any other three letter acronym. Each LLD is nothing more than a linear discriminant localized using two Gaussian functions in conjunction with a sigmoid which provides a smooth transition across the between class margin. In this design, the network will be constructed by adding hidden units, one by one, to refine the shape of local portions of the overall decision boundary. This will be an iterative process in which the scale of the LLD’s is decreased from coarse to fine, as the process of network construction focuses on finer and finer details of the decision boundary. This is analogous to function approximation using multi-scale radial basis functions, with the distinctions that it is a discriminative function being approximated, based on localized basis functions which have a discriminative aspect to their shape. It is not hard to find examples of methods for multi-scale RBF function approximation in the literature. Also moving in the direction of the more complex shape of the LLD, many methods depart from the use of purely isotropic RBF's, as in [8] where multi-scale anisotropic RBF's are utilized with volumetric data, as well as in [9] in which arbitrarily-oriented elliptical RBFs are used to fit data on an irregular grid, or in [10] where oriented DOG's are used, “since the DoG function closely approximates a … (Laplacian of Gaussian) function, the representation can be considered a hierarchy of the 3D edges”. Here the analogy between a multi-scale approximation of edges of a 3D object, and multi-scale approximation to decision boundaries (although in an n-dimensional space) should be appropriate. This comparison is also apparent in the basis functional form used in [14], where “Implicit surface f(x) = 0 separates the space into two parts: f(x) > 0 and f(x) < 0. Let us assume that the orientation normals are pointing into the part of space where f(x) > 0. Thus f(x) has negative values outside the surface and positive values inside the surface.” In fact, the illustration of the basis function in figure 4 in [14] has a striking resemblance to the illustration of the LLD in figure 3 (in [5]), which shows pairs of LLD's as one of the hidden units is iteratively grown. One more example of the utilization of multi-resolution discriminant functions to model edges in images is given in [11]. It is also a coincidence that in [14], the statement “Thus a hierarchical approach with locally supported basis functions where data reconstructed at coarser levels serve as carriers for finer levels may substantially accelerate scattered data fitting” is also appropriate in terminology for our purpose of pattern drift adaption. Clearly the use of the term “carrier” in [14] is meant in the sense of hierarchical approximation, but in our case the term is also quite appropriate in that the coarser scale LLD's are literally meant to “carry” the associated (ie nearby in feature space) LLD's at finer scales along in feature space, as the coarser scale LLD's are adaptively updated using new data. Each set of LLD's at finer scales, in turn, are meant to carry the LLD's at even finer scales as they adapt.

The basic argument for the use of multi-scale components for concept drift adaptation:

In [13] it is argued that the primary benefit of multi-scale approximations to decision boundaries in terms of matching the complexity of local portions of the overall decision boundary to the amount of training data available at each between-class boundary region over the training dataset. This is a treatment of the training data that considers the required complexity of local portions of the decision boundary for best generalization performance. Here we hypothesize that an architecture providing a set of multi-scale decision boundary components allows for a more principled way of adapting to pattern drift. The basic concept behind the support vector machine, margin maximization, is dependent upon the identification of support vectors which define the two boundary hyperplanes of the margin. All other training data is ignored in the focus on the details of the data in the region of the margin, under the standard assumption that the position and orientation of the between-class boundary region is static. The question is then how to adapt, using an SVM in this example, to pattern drift, which will most likely affect the margin region. The basic premise is that the support vectors shift, to some degree, in concert with the shift in the entire ensemble of data from each class. If we were to focus exclusively on adaptation of the support vectors using new data falling close to the support vectors or margin boundaries, and classified with sufficient confidence to gradually update the SVM hyperplane, then all would be well. The problem is that the support vectors generally represent a relatively small fraction of the data from each class. Since the frequency of occurrence of data near the margins is relatively low, and if the overall location and orientation of the data from each class shifts before examples near support vectors (or more generally, near the margins) can be utilized for unsupervised adaptation, then the margin region, which had been established using static training data, could be effectively “swamped” by the data from one class shifting across the margin. We would be ignoring the evidence from all regions beyond the margins that drift was occurring if we used only the SVM margin region to update the SVM. At the same time the error rate of the classifier would increase, with the margin region being updated using incorrectly classified training tokens assumed to be valid for updates. While updating an SVM in an online fashion is admittedly impractical, it illustrates the problem of focusing on the margin region to track drift. This sets up our argument for use of both coarser and finer scale components of the decision boundary in order to better track drift.

One solution to this problem is to construct a classifier which can essentially update it's training as new, drifting patterns are processed. In unsupervised operation, if we only select the results of the higher confidence classifications as representing hypothesized "ground truth" to update the classifier's decision boundaries then the classifier would have the ability to (slowly) adapt to drifting patterns. Since pattern data typically exists in clusters, adjusting both the position and orientation of functions which represent each cluster, based on higher-confidence classifications, should also provide adjustment that is concomitant with patterns which lie at the edges of those clusters, to some degree. Clearly the classification of incoming data can only replace ground truth labeling to the extent that clusters are relatively compact and the distribution of outliers is manageable, and that the drift is slow enough for the classifier to be updated to track the drifting distributions effectively, before it is “left in the dust”. It should be re-emphasized that this proposal is meant to optimize performance of a classifier only in applications where patterns are expected to drift relatively slowly, such as with the normal beat morphology of the ECG over the course of long-term monitoring (as in the CCU or ICU, or possibly Holter-type monitoring). This type of concept drift has been referred to as “virtual drift” ([1], [2]). Applications such as these are important enough to devote attention specifically to optimizing classification performance by hopefully improving the means of tracking drift.

Overview of the localized linear discriminant network training and structure:

The basic localization function (detailed in [5]), uses two Gaussian functions for localization (one along the normal vector, and one in all directions orthogonal to the normal vector) in conjunction with a sigmoid effecting a "transition region", whose width is scaled to the width of the margin, so that the classification confidence varies smoothly from one class to the other as the between-class margin region is crossed (and defined as zero at the hyperplane surface). Each LLD is “grown” from a seed pair of tokens from each class (we will only discuss the two class case here, for simplicity), with neighboring tokens being added until the modified perceptron training algorithm yields an error measure that is above threshold, indicating that the extent of the local portion of the decision boundary which can be modeled by a locally linear component has been exceeded. This is a large margin / soft margin perceptron training algorithm in which all tokens belonging to the local training set have a weighted contribution to the orientation of the LLD normal vector. The value of the sigmoid associated with each token determines the weight of the contribution from each token, noting that the width of the sigmoid is constantly adapted to a running estimate of the width of the margin at each iteration. During this process, the normal vector origin is also continuously updated to an estimate of the center of gravity of the training data at the margins (the data that would be candidates for support vectors). For optimized convergence, the updates to the normal vector rotation, and the origin center updates (both parallel and orthogonal to the normal vector) are updated independently to allow more effective adaptation of the learning rates for each component of the updates.

After the acceptable limit of the set of local training data is reached, the two Gaussian localization functions are developed. One Gaussian localizes the effect of the LLD along the direction of the LLD normal vector. All directions parallel to the hyperplane are collapsed into a second dimension, and the second Gaussian is used to model the extent of the local training data along this second “dimension”. The product of these two Gaussians, multiplied by the sigmoid, becomes the composite localization function for each LLD. A network is constructed by adding LLD's one-by-one, using a single LLD at the output layer (for this two-class example) to coordinate the outputs of all of the LLD's in the single hidden layer in order to minimize the error rate of the network as a classifier. The LLD-based output layer is effectively nothing more than a way of balancing the weights of the localized hidden units with respect to cooperation and competition, while also providing a measure of classification confidence.

The starting point for addition of new LLD's is based on the output of the entire network, in order to focus on training tokens generating the highest error (or lowest classification confidence). If the error is too high (or classification confidence too low) at any training token, training of a new LLD will be initiated, seeded by the token in question and it's nearest neighbor from the opposite class. This approach is similar to the AdaBoost algorithm in it's iterative focus on error-generating tokens in the training dataset and it's method of adding linear discriminants (localized, in this network) to the network in an attempt to minimize error. Obviously, the output layer LLD is re-trained after the addition of each new unit to the hidden layer. It would also be possible to re-train the output layer after the addition of each new training token to each new LLD (this was not in the original system), and might improve the choice of the local training tokens for each new LLD with respect to accuracy.

While this is, effectively, a single hidden layer network, it should be kept in mind that the feature space used as input may be one that is a set of feature extraction operators, from simple to complex, developed through unsupervised or supervised learning or defined such as in a deep scattering network ([4], [6], [19]). There is nothing preventing us from taking all features output from all layers of a deep network as inputs, which is exactly what is intended in the use of deep scattering networks. It is also possible to use a feature selection procedure independently for each LLD; each LLD could then be viewed as operating in a distinct feature subspace. This is almost certainly preferable in controlling the complexity of the overall network, as well as optimizing the generalization performance of the local discriminators.

Modification of the LLD network to encourage a range of scales for the LLD units:

This composition of decision boundaries should be relatively efficient, as opposed to constructing decision boundaries using components at a single scale (see [13]). A very loose analogy could be made to the use of wavelets for signal representation, if the wavelet coefficients below a given threshold are zeroed out. In this case, we are constructing the approximation to the decision boundary (as opposed to a signal), and are selecting wavelets at the appropriate location and scale to make the approximation iteratively more exact with regard to the “optimum” decision boundary. The analogy is more complete if we compare the zeroed out wavelet coefficients with components of the decision boundary that are not selected for use. One important distinction is that the LLD's are not constrained to locations on a grid, as in the DWT, but are more analogous to selecting wavelets for a signal approximation from the set of wavelets in a CWT.

At each scale, Parzen window smoothing will be used to locate peaks in density estimates separately for each class. The initial Parzen window bandwidth will be based on the total variance of the data. Seed tokens at each scale will be chosen from the set of training tokens at local peaks in the density estimates (separate for each class). At each scale, seed tokens will be pairs of tokens at local density peaks which are nearest neighbors. At coarser scales, since the density estimates are generated separately for each class, seed tokens will be closer to class data cluster centers of gravity, in comparison to seed tokens at finer scales which will be closer to the between class margin regions. As LLD's are grown in this multi-scale framework, the tokens added to the local training datasets will be those which are chosen by a measure biasing the choice towards close proximity to the current cluster for each class, and higher values of the density estimate. For example, for each candidate seed token pair, the selection measure might be the product of the within-class densities for each token and the inverse distance between the pair of tokens. This method of selecting the centers of the neighborhoods for choosing training tokens from each class is oriented towards a combination of densely clustered training tokens within a given class, and relatively close proximity to a dense cluster of training tokens from the opposite class. At the same time the scale factor in the LLD training algorithm (see [5]) will be set so that for coarser scales the sigmoid widths are wider, effectively integrating more of the training data in the final hyperplane orientation. The third parameter to be scale dependent will be the error used to halt the growth of the LLD, so that at coarser scales more training tokens will be incorporated. These modifications of the LLD training algorithm will bias the resultant LLD's trained with parameters for coarser scales to incorporate more training tokens and “cover” more of the feature space than LLD's trained with increasingly smaller Parzen window bandwidths, steeper sigmoid functions, and lower error thresholds. The intent is to encourage a decomposition resulting in multiple LLD's at each scale. Each layer of the multi-scale LLDN will represent a decomposition at a given scale.

For the purpose of drift adaptation, we want to encourage coverage of the training dataset at each scale by using a two stage process. In the first stage the Parzen window smoothing will generate a map of training data density at that scale. When a sufficient percentage of the data has been used in training data a set of LLD's covering that scale, the second stage will be triggered, which will focus not on data density but the density of error / classification confidence. While the scale-related parameters will remain the same, the Parzen window smoothing will now be applied to error / classification confidence, and the choice of seed tokens for the LLD's to be grown in stage two will be based on local peaks in the error density estimate at that scale.

At the coarsest scale, there is some possibility that a relatively simple decision boundary structure may be found, and that a classifier with minimal complexity may be generated. In most cases it will likely be necessary that the process of generates a more and more detailed decision boundary structure, and halts at the level of detail (scale) required for optimum generalization performance. This could be viewed as a method of regularization, explicitly using an ordered sequence of a smoothing parameter to allow development of necessary classifier complexity but no more than what is optimum. At the completion of each layer, cross-validation could be used to estimate generalization error, halting the process when generalization starts to degrade. Since each LLD is localized and "covers" a subset of training tokens, the cross-validation error rate can be estimated by bootstrap re-sampling of training data "covered" by each LLD, followed by re-training each LLD (ignoring any order sensitivity in the overall process). Alternatively the proper scale to halt network construction could be determined by bootstrap re-sampling and construction of a set of classifiers only up to the given scale. It is expected that using varying degrees of detail in construction of the overall decision boundary over the training dataset will provide a more optimized level of tuning of classifier complexity for generalization with regard to the varying densities and between-class margin widths of the training data at different regions in feature space.

If during the network construction process an acceptable level of generalization performance has not yet been achieved, it would be possible to add LLD's endlessly. In order to avoid this situation, the addition of LLD's to the current layer will be stopped when generalization performance starts to become asymptotic; this will be the indicator required to move to the construction of the next layer at the next finer scale. It is worth repeating that each layer is not like the layers in, say, a CNN, but rather simply a group of LLD's at each scale, and an organization which is focused on how drift adaptation is handled differentially over scale.

<u>The basic mechanism for updating multi-scale layers of LLD's:</u>

Practical implementations of the SVM algorithm typically use parameters to allow a “soft margin” which allow contributions from training tokens which cannot be placed precisely on one of the margin boundary surfaces, effectively increasing the amount of training data encoded in the decision boundary. This technique offers the potential tradeoff for lower generalization error with higher training error under the assumption that the support vectors discovered in the hard margin case are not noise-free, unvarying representatives of the margin boundaries. In this algorithm, the coarser scale components of the decision boundary will have wider / softer margins, which will give higher error on static training data, but hopefully allow better tracking of drift and yield higher accuracy when the finer scale components can settle at times when pattern drift might slow.

Since LLD's are composed of Gaussian RBF's for localization, and a sigmoid for a smooth transition in classification confidence across the margin region, they are well suited to the purpose of tracking drift. In this new system, coarse scale LLD's will be updated, both in location and hyperplane orientation, based on unsupervised classification confidence. This is in contrast, for example, with the “Concept Following” algorithm of Hadas et. al. [16], where updates to clusters are made using a hard threshold. If our assumption that the distributions for each class drift, to some degree, in concert is correct, then we can use the shifts and rotations of the coarser scale components to drag and twist the associated finer scale components as a way to dynamically update the entire classifier, as well as put the finer scale components in better position to use data falling into their (more compact) regions for their own updates. The updates specific to the finer scales will, in general, be at lower rates due to the lower frequency of occurrence of pertinent data falling into the effective regions of support of the finer components.

An example (vowel formants) was given earlier to make the case that the details of a decision boundary will vary somewhat independently from the coarser scale components. Here we are proposing to adjust the finer scale components based on the same deltas of the parameters of the coarser scale components – but those are the only default adjustments we can assume for the finer scale components, unless the dynamic relationships between components over scale have somehow been learned (that level of complexity is far beyond what is being proposed here). The next step, at each scale, is to do the online, unsupervised updates based solely on the ability of the network to generate a classification with some confidence.

Since LLD's are centered on a local origin and have a normal discriminant vector, it is easy to update their positions and orientations using a convex combination of updates to their neighboring coarser scale LLD's, weighted by proximity to each neighbor. A high level view of this system is of a series of layers of discriminant components, each being pushed and twisted by the previous layer while adapting on their own, and then pushing and twisting all of the layers following them, each layer adapting at slower and slower rates due to the effect of the increasing localization at each finer scale.

The basic concept for updating classifier boundaries, then, is that at each scale, updates to the position and orientation of each LLD are propagated to finer scale LLD's which are associated with (nearby in feature space) that LLD. Online training updates at the first (coarsest) scale are simply weighted by classification confidence. Updates to finer scale components will be a convex combination of the updates from neighboring components at the previous scale, weighted by proximity to each neighboring LLD, in combination with overall classification confidence.

In this architecture the finer scale units are not used to modulate the overall response of the classifier, but merely to either fill in details where more resolution is needed, or else to bolster the classification in regions of the decision boundary where there may be some conflict between neighboring hidden units (so there is no deep “credit assignment” problem here – this is effectively a single hidden layer network). This is not to dismiss the critical utility of the finer scale units, which are generally necessary to define the large margin decision boundary, albeit in a smoothed, piecewise linear fashion.

Note also the fact that as the entire network is updated incrementally, in an unsupervised fashion, there will generally be minimal conflict or need for cooperation between hidden units. This should be much harder to do with deeper networks with hyperplanes of infinite extent. It is interesting to note that CNN's do have hidden units which have minimal overlap / conflict with other hidden units due to spatial isolation.

It is anticipated that allowing updates to all hidden units which have any significant response to each new example will compensate for the fact that we will not be allowing new hidden units to be added to the network. During network construction and training using the static training dataset, new hidden units are added in order to focus on higher error generating regions, in a fashion similar to the AdaBoost algorithm, but are trained only on the local set of tokens chosen during the “growth” phase. In order to maintain structural stability of the network, which is fixed during training as “optimal” (for the training data), the addition of new hidden units will not be allowed during adaptation. Since in the unsupervised case there can be no errors generated during online classification, only classifications with higher or lower confidence, it would be risky to add hidden units based simply on boosting classification confidence (and not correcting error if labeled data were available). However it is possible to allow multiple, neighboring hidden units to be updated, using the classifier output label and confidence; in this way we allow some competitive / cooperative reinforcement of the presumed label of the latest example. This also allows LLD’s which are nearby the example in question to be updated with respect to their own localized view of the class for that example. This simply means that LLD’s sufficiently close to the example, and whose output agrees with the overall network output will be updated to reinforce that classification through the large margin / soft margin perceptron update, with the update in the opposite direction for LLD’s local to the example whose outputs disagree with the overall classifier output. Additional control over the update rates would also be possible, so that the update rates could be further modulated according to the scale of the individual LLD's (and not simply by frequency of "capture" of online data). This would provide one additional tuning parameter for this critical function of online updating.

Optimizing dynamic performance with drift and the concept of an ensemble of classifiers over scale:

It is also not necessary, and probably not desirable to use a set of dyadic scales (which may not even make sense here). This decomposition is meant to look more like a CWT than a DWT, in the sense of being overcomplete, and having a relatively fine change in scale moving from coarse to fine, in the hope that updates at each scale under conditions of drift are more effective. If each scale is more closely related to the preceeding one, it is anticipated that the initial shifts and rotations for the next finer scale components are closer to the true shifts and rotations necessary to track drift, since the scales do not change too significantly from one level to the next, and thus the dynamic data sets used to update the associated components do not change too dramatically from scale to scale either.

One of the benefits of this approach is that some classification accuracy can be maintained, albeit at generally lower confidence, while drift is occurring; this is in contrast to, say, an SVM which would proceed to make high confidence classifications (in error) as drift moves the true margin region away from the margin determined using the static training data. It is then also proposed to obtain classification over varying levels of scale; an ensemble of classifiers will be generated over the set of scales. This ensemble will consist of a classifier based on the weighted combination of only the coarsest scale components, followed by one using only components from the two coarsest scales, and so on. The output unit(s) which collect and weight data from each hidden unit at a particular range of scales can, of course, select any weights to associate with each hidden unit over the range of scales for which it is operating. This functionality is a potential unique to classifiers having a multi-scale architecture, and is intended to capture the highest confidence classifications based on how well the individual components have been updated over scale in the dynamic situation of adaptation.

References:

[1]: “A Survey on Concept Drift Adaptation”, Pechenizkiyl, Žliobaitė, Bifet, and Bouchachia
http://eprints.bournemouth.ac.uk/22491/1/ACM%20computing%20surveys.pdf

[2]: "Handling Concept Drift in Medical Applications: Importance, Challenges and Solutions”, Pechenizkiy and Žliobaitė, Department of Computer Science, Eindhoven University of Technology
http://www.win.tue.nl/~mpechen/talks/cbms2010_Tutorial3.pdf

[3]: “Vowel Acoustic Space Development in Children: A Synthesis of Acoustic and Anatomic Data”, Vorperian and Kent
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2597712/

[4]: Scattering section of the Data Processing and Classification group webpage,
The Computer Science Department of ENS (École Normale Supérieure de Paris)
http://www.di.ens.fr/data/scattering/

[5]: “A Network of Localized Linear Discriminants”, Martin S. Glassman, NIPS '91
https://papers.nips.cc/paper/525-a-network-of-localized-linear-discriminants

[6]: “Invariant Scattering Convolution Networks”, Joan Bruna and Stephane Mallat,
CMAP, Ecole Polytechnique, Palaiseau, France
http://www.cmap.polytechnique.fr/scattering/pami-final.pdf

[7]: “Scattered Data Models Using Radial Basis Functions”, Armin Iske
In: “Tutorials on Multiresolution in Geometric Modeling”

[8]: “Modelling and Rendering Large Volume Data with Gaussian Radial Basis Functions”,
Derek Juba and Amitabh Varshney
https://www.cs.umd.edu/gvil/papers/juba_UMDTR07.pdf

[9]: “Constructing 3d elliptical gaussians for irregular data”,
W. Hong, N. Neophytou, K. Mueller, and A. Kaufman
https://cvc.cs.stonybrook.edu/Publications/2009/HNMK09/

[10]: “Multiscale volume representation by a DoG wavelet”, S. Muraki
http://ieeexplore.ieee.org/document/468408/

[11]: “Multiresolution Linear Discriminant Analysis: Efficient Extraction Of Geometrical Structures In Images”, Sinisa Todorovic and Michael C. Nechyba
http://ieeexplore.ieee.org/document/1247141/

[12]: “Approximation by superposition of sigmoid and radial basis functions”,
Advances in Applied Mathematics, HN Mhaskar and CA Micchelli

[13]: “Multi-scale Kernel Methods For Classification”,
Nick Kingsbury, David B H Tay, M Palaniswami
http://www-sigproc.eng.cam.ac.uk/foswiki/pub/Main/NGK/Kingsbury_MLSP05.pdf

[14]: “A Multi-scale Approach to 3D Scattered Data Interpolation with Compactly Supported Basis Functions”, Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel
http://www.cs.jhu.edu/~misha/Fall05/Papers/ohtake03B.pdf

[15]: “The Cascade Correlation Learning Architecture”, Fahlman and Lebiere, NIPS '90
https://papers.nips.cc/paper/207-the-cascade-correlation-learning-architecture.pdf

[16]: “Using unsupervised incremental learning to cope with gradual concept drift”,
David Hadas, Galit Yovel, and Nathan Intrator
http://www.math.tau.ac.il/~nin/papers/HadasConnSci2011.pdf

[17]: “The acoustic characteristics of Greek vowels produced by adults and children”,

A. Sfakianaki, MA Thesis, Department of Linguistic Science, University of Reading, 1999

[18]: Determining the training window for small sample size classification with concept drift."
Zliobate I. And Kuncheva, L.,
Proc. of IEEE Int. Conf. on Data Mining Workshops. ICDMW. 447–452

[19]: Scattering Invariant Deep Networks for Classification, Stéphane Mallat,
IHES Ecole Polytechnique
http://helper.ipam.ucla.edu/publications/gss2012/gss2012_10668.pdf

[20]: "Class Counts in Future Unlabeled Samples (Detecting and dealing with concept drift), Jerome Friedman, Stanford University
http://statweb.stanford.edu/~jhf/talks/qc.pdf