## Research problem

**Develop a method to adapt a learned model (eg: classifier, generative model) to a changing (dynamic) data distribution**

**Literature**

* This is a well-researched problem called **domain adaptation** (mostly new research: 2006 – 2015)
* Other related areas are transfer learning, and incremental learning
* Some papers
  + Continuous Manifold Based Adaptation for Evolving Visual Domains (video scene detection under varying conditions) - <https://cs.stanford.edu/~jhoffman/papers/Hoffman_CVPR2014.pdf>
  + Classifier Adaptation at Prediction Time (a simple moving window technique for changing class distributions p(y)) - <http://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Royer_Classifier_Adaptation_at_2015_CVPR_paper.pdf>
  + Predicting the Future Behavior of a Time-Varying Probability Distribution (adapt to changing p(x), good for predicting future frames of a video stream) - <https://arxiv.org/abs/1406.5362>

**Practical use cases**

* Heavily used in NLP
  + Part-of-speech tagging in documents from different domains
  + Voice recognition etc. for different speakers/ acoustic conditions
* Vision
  + Object recognition under varying conditions (light, different scenes/ clutter)
  + Video scene detection under varying conditions (weather/ lighting)
* Anomaly detection
  + Need a correct estimation of the data distribution to accurately find anomalies. In cases where the underlying data distribution is dynamic (eg: network traffic), adaptation is required

**Scope of our work**

* Consider a technique that estimates a probability distribution over a dataset. Some possible methods are;
  + Simple non-parametric estimation
    - Empirical cumulative distribution function (ECDF)
    - Histograms
    - Kernel density estimation (KDE)
  + Parametric estimation
    - Hidden Markov models (HMM)
    - Gaussian mixture models (GMM) and other mixture models
  + More advanced (eg: deep learning) models
    - Auto-encoders
    - Generative Adversarial Networks (GAN)
* We will consider ECDF and KDE as they are simple, and more suitable for mathematical analysis.
* As the underlying distribution changes and new samples are generated, we must use them to change (adapt) the originally learned model.
* We assume that the underlying changes are continuous rather than one-off. For example, some characteristic of network traffic (eg: inter-packet arrival time) change continuously. Contrast with the change in characteristics of a voice signal due to a different speaker.
* We should be able to control the forgetting effect as the model adapts to the new distributions, and there should not be catastrophic forgetting (completely forget previous knowledge).
* Currently, we are considering generative models that estimate a distribution of the dataset P(X)
* Can we also use these techniques for discriminative models, such as neural network classifiers?
  + Assume that at test time examples are unlabeled (unsupervised domain adaptation)
  + Better if we can accommodate new classes emerging at test time
* How do we demonstrate performance and applicability of the developed technique?
  + Apply to an existing standard dataset, so we can compare with current work
  + Apply to our own dataset as a demonstration on a novel problem (eg: adapting to network traffic characteristics)

## Implementation

Software used: Python 3.6.0 (Anaconda 4.3.1 distribution), scikit-learn: machine learning package

**Distribution adaptation by moving buffer of samples**

* Create a random variable (single) with a probability distribution that is complex enough
  + We have used a Gaussian Mixture Model (GMM) (Weighted sum of Gaussian components)
* Make the distribution time varying
  + We have used a sinusoidal variation for mean and variance of each component as follows
* At each time point, generate some samples, and add it to a buffer (with fixed size). As more and more new samples are generated and added to the buffer, the older ones are pushed off it.
* At each time point (or at suitable time intervals), we use the samples in this buffer to re-learn the probability distribution.
* By adjusting the fixed buffer size, we can incorporate a memory factor (K) to this technique (eg: as in the equation below).
* Compute some distance metrics to check if the adaptation is acceptable (distance between current true distribution and adapted estimation of the distribution). We have used Kullback–Leibler divergence, Kolmogorov–Smirnov statistic, Earth mover's distance or the 1st Wasserstein distance, and mean squared error.
* Advantages
  + Very simple
* Disadvantages
  + Re-learning the distribution for adaptation is very inefficient, especially if the dataset (buffer size) is large
    - We are not using any information from the current model for the adaptation
  + Figuring out when to re-learn the distribution can be difficult (when has the distribution changed enough to warrant a re-learn)