EECS 349, Homework 8 Marc Gyongyosi

Problem 1:

a) If the dataset were to change after each round, AdaBoost's performance would be severely affected. AdaBoost depends on the dataset not varying from run to run as it tries to focus on the hardest and previously misclassified examples during each new run. If some of the examples change, then there will be no way to guarantee a successful boosting with AdaBoost. In general, however, I would assume that if q <<1 (i.e. q is very small and very close to 0), then it might still be possible to get O.K. results.

b) I guess there is a number of different ways how we could deal with the changing data. One of them would look the following:

As a central part of my modification would be a check that makes sure that any given training sample that is used during each training run has not changed its label from the previous to the current run. If it has, then I would discard that sample by setting D(that sample) to 0.

I conjecture that this would work well as long as q is small and the number of samples that change from run to run is smaller than those dominating during each run.

Mathematically, this could be represented by adding a function "m" which is 1 if the sample hasn't changed and 0 if it has.

$$m\big(y_i(t)\big) = \begin{cases} 1 \; if \; y_i(t) = y_i(t=0) \\ 0 \; \; if \; y_i(t) \; \neq \; y_i(t=0) \end{cases}$$

$$D_{t+1}(i) = \frac{D_t(i)}{Z(t)} * \begin{cases} e^{-a*t} * m\big(y_i(t)\big) \; if \; y_i = h_t(x_i) \\ e^{a*t} * m\big(y_i(t)\big) \; if \; y_i \neq h_t(x_i) \end{cases}$$

c) This modification would work significantly better than the unmodified one, because that way the changing or incorrect samples wouldn't affect the current training runs and the selection of the right samples for the next ones. Since m is comparing to y(t=0) it would let the samples that flipped back to the correct values after being incorrect be reconsidered in the following runs. Theoretically, if q is small enough, this should work regardless of what T is.

Problem 2:

a) In active learning, the learner actively influences what inputs are fed to itself. This means, that training samples are not randomly drawn from a training set, but rather, the algorithm determines which ranges/areas/examples it wants to draw from. For example, in binary decision surface learning this allows the learner to more closely look at the examples around the decision surface and not those that are far away. This renders the entire process more efficient and effective.

b) One would look at the set of hypotheses and pick the instances that cause the greatest conflict/disagreement between hypotheses in the version space. That way one is most likely to split the version space. This is called Query by Committee (QBC).

c) In the case of selective sampling, they present a polynomial fit and an exponential fit. The

exponential one has higher determination (0.995 vs 0.937). Thus for selective sampling, the error rate seems to decrease exponentially with the number of training samples. The function they quote is: error = e^(-0.0218*m + 0.071). In the case of random sampling, they try to also fit both an exponential and polynomial. The polynomial has better determination (0.987 vs 0.981). The function they quote for that is: error = (0.0514*m + -0.076)^(-1).

Problem 3:

HMMs vs DNN for speech recognition:
Using Hidden Markov Models it is effective and simple to model time-varying spectral vector sequences. This makes HMMs ideal for using them to model large vocabulary based speech. Thus, they are at the foundation of almost all modern Speech Recognition engines. GMMs provide the means to determine the relationships between HMM states and the acoustic input. Thus, these systems are called GMM-HMM speech recognition systems.

GMMs, however, have an important disadvantage: they are inefficient when looking at data that lies on or near a non-linear manifold in the data space. Speech itself is produced by modulating only few parameters of a dynamical system which means that the actual system is of much lower dimensionality than might be assumed initially. DNNs are better at modeling this kind of data and with recent advancements in training them with many hidden and non-hidden, linear and non-linear layers they can provide advantages for dissecting and analyzing acoustic models to use them to estimate HMM states.

Traditional approach: As early as 1980, researchers have combined Artificial Neural Nets (ANN) and Hidden Markov Models (HMM) to achieve automatic speech recognition. The most relevant ones were those that used the ANNs to estimate the HMM state-posterior probabilities. This has later developed into GMM-HMM, which – with the advent of GPUs – has become one of the most dominant approaches in the past 2 decades.

Proposed approach: In this paper, the authors propose a novel approach that is a hybrid between a pre-trained, deep neural network and a context-dependent Hidden Markov Model. The pre-training is accomplished using a deep-belief network pre-training algorithm. This deep belief network is not used after pre-training is complete; only its recognition weights are used afterwards in conjunction with the context dependent Hidden Markov Models.

New one better? How? Experimental evidence: Previous approaches have traditionally been very limited in their ability to do context-based speech recognition. Also, using only backpropagation made it difficult to train and exploit more than two hidden layers in the ANN. In addition, the previously used context based models didn't take advantage of all of the numerous effective techniques developed for GMM-HMMs.

In terms of actual performance, the proposed CD-DNN-HMM achieves better results (in terms of accuracy) than CD-GMM-HMMs. However, the main caveat is that training the CD-DNN-HMM is significantly more computationally expensive compared to training CD-GMM-HMMs. The reason is that training CD-DNN-HMMs cannot be parallelized as well as the training algorithms on the CD-GMM-HMMs (at least not yet). Nevertheless, in terms of decoding, CD-DNN-HMM seem to be very efficient so the testing and real-time operation should not be a problem for actual applications. The authors present experimental evidence to support these claims. Most importantly, they compare the performance of CD-GMM-HMMs with that of CD-DNN-HMMs on a business data set provided by the search engine Bing. In table VI it can be seen that CD-DNN-HMMs outperform any of the CD-GMM-

HMM setups by about 1.5%. Also, as the number of hidden layers increases, the performance seems to improve, however the gain becomes marginal as the number of hidden layers grows beyond 5. Also, the authors examine whether or not the pre-training (the biggest caveat to the CD-DNN-HMMs) is really necessary. They were able to observe a significant increase in performance for 2 layers. When only 1 layer was used, they weren't able to see this increase in performance.

Problem 4:

Thesis:
The main thesis behind this paper is that in the area of classifiers, recent progress has been rather illusional/marginal and that the minimal performance increases shown in very complex classifiers such as SVMs and Neural Nets are not big enough when compared to very simplistic approaches to justify the increased complexity.

Evidence:
One of the examples that the authors present is that simple classifiers' major limitations – such as higher dimensional data sets (e.g. checkerboard patterns, intertwined spirals, etc.) - are not very applicable to actual real world applications and thus the limitations are not that relevant. They compare Fischer's LDA with the best performing classifier they were able to find in literature by running them on 10 data sets (table 1). In most cases, the error rate of the LDA classifier was only slightly higher than that of the best one from literature.

Another example the authors go through relates to the decrease in improvement consistent with an increase in classifier complexity on the sonar data from UCI. The authors set up a neural net with a variable number of hidden nodes and then calculate the error rate for each. As can be seen in the figure there is no clear increase in performance with the increase in number of hidden nodes.

Another point the authors bring up is in the context of how training and testing data is selected; especially when it comes to data about people and population. They show how misclassification rate changes as time goes on, simply because the population changes. They mention how customers' tastes change over time and product cycles.

In addition to these examples, the authors also present 3 less obvious reasons why things might go wrong with classifiers. First they mention errors in class labels, then arbitrariness in the class definitions (e.g. changing/randomly selected class thresholds), and finally the measure that is optimized for (e.g. likelihood, proximity, distance, etc.).

I think the author brings up valid points when looking at simple classifiers. In simple cases, I agree that one should go with Occam's Razor and try to use the most simplistic (working!) solution possible. However, I do believe that when it comes to more complicated problems such object or place recognition in computer vision, or also speech recognition and natural language processing tasks there is a substantial need for the more complicated models and frameworks; especially with machines operating in more and more unstructured and most likely also highly complex environments. So in a nutshell, yes, I can see how people seem to over complicate simple problems, but as the problems themselves become more complicated (and that's where I disagree with the author) I think highly complex approaches are appropriate.