# EFFICIENT ONLINE LEARNING WITH INDIVIDUAL LEARNING-RATES FOR PHONEME SEQUENCE RECOGNITION

*Koby Crammer*

Department of Electrical Enginering, The Technion
Haifa, Israel
koby@ee.technion.ac.il

## ABSTRACT

We describe a fast and efficient online algorithm for phoneme sequence speech recognition. Our method is using a discriminative training to update the model parameters one utterance at a time. The algorithm is based on recent advances in confidence-weighted learning and it maintains one learning rate *per* feature. The algorithm is evaluated using the TIMIT database and was found to achieve the lowest phoneme error rate compared to other discriminative and generative models with the same expressive power. Additionally, our algorithm converges in less iterations over the training set compared with other online methods.

***Index Terms***— Online learning, confidence weighted, large margin, discriminative training, speech recognition.

## 1. INTRODUCTION

Hidden Markov Models (HMMs) have been a primary tool for building automatic speech recognition (ASR) systems for more than fifteen years [1]. Their parameters were often estimated to best model the likelihood of jointly the acoustic signal and underlying phonemes, aka generative training. Recently (e.g. [2, 3, 4, 5]), researchers have proposed to use the discriminative approach for building ASR systems, by either proposing alternative parameter estimation techniques (e.g. [6]), or also by replacing the joint model (e.g. [2, 7]) with a conditional model of the phonemes *given* the acoustic signal (such as conditional random fields, CRF [2]).

Recently [8, 9] the notion of confidence weighted (CW) online learning was introduced. CW learning explicitly models classifier weight uncertainty using a multivariate Gaussian distribution over weight vectors. The learner makes online updates based on its confidence in the current parameters, making larger changes in the weights of uncertain features. Empirical evaluation has demonstrated the advantages of this approach for a number of binary natural language processing (NLP) problems.

In this paper, we develop and evaluate new confidence-weighted learning algorithms for ASR. As other online algorithms, our method process one utterance at a time. However, as opposed to many previous algorithms, our algorithms maintain one learning rate per feature, allowing it to focus in the most uncertain features. We found that our algorithms yields significantly lower phoneme error rate compared with models with the same number of parameters and converges faster than other online and batch methods.

The author is a Horev Fellow, supported by the Taub Foundations. This work was also supported by German-Israeli Foundation grant GIF-2209-1912.

## 2. METHODS

### 2.1. Problem Setting

In automatic speech recognition, we are given an acoustic representation of a speech utterance and wish to predict the sequence of phonemes spoken. We represent the acoustic signal $\vec{\boldsymbol{a}}$ as a sequence of feature-vectors $\vec{\boldsymbol{a}} = (\boldsymbol{a}_1 \ldots \boldsymbol{a}_T)$, where $T$ denotes the length of the utterance (in appropriate time units) and the features belong to a finite-dimension vector space $\boldsymbol{a}_t \in \mathbb{R}^d$. Each utterance is a realization of a sequence of phonemes, $\vec{p} = (p_1 \ldots p_T)$, where each phoneme belongs to a finite set of possible phonemes $p \in P$ of size $|P|$. Since different phonemes have different typical lengths, for example, vowels are often longer than consonants, we often have consecutive sub-sequences with the same phoneme, i.e. $p_t = p$ for some $p \in P$ and $t = \tau \ldots \tau + \nu$ for some non-negative integers $\tau, \nu$. Our goal is to build a mapping from signals $\vec{\boldsymbol{a}}$ (for all possible lengths $T$) to the corresponding phoneme $\vec{p}$, such that $\vec{p}$ capture the spoken utterance.

### 2.2. Model and Inference

Our score model is based on a linear score $s(\vec{\boldsymbol{a}}, \vec{p}; \boldsymbol{w})$ for each acoustic signal $\vec{\boldsymbol{a}}$ and its corresponding transcription $\vec{p}$, defined as

$$s(\vec{\boldsymbol{a}}, \vec{p}; \boldsymbol{w}) = \boldsymbol{w} \cdot \phi\left(\vec{\boldsymbol{a}}, \vec{p}\right)$$

where $\phi\left(\vec{\boldsymbol{a}}, \vec{p}\right) \in \mathbb{R}^D$ is a feature-vector representation of the acoustic-utterance and phonemic-sequence pair, and $\boldsymbol{w} \in \mathbb{R}^D$ is a vector of feature weights.

Given the weight vector $\boldsymbol{w}$ and an acoustic input $\vec{\boldsymbol{a}}$ we predict the sequence of phonemes with the highest score,

$$f(\vec{\boldsymbol{a}}; \boldsymbol{w}) = \arg\max_{\vec{p}} s(\vec{\boldsymbol{a}}, \vec{p}; \boldsymbol{w}) = \arg\max_{\vec{p}} \left(\boldsymbol{w} \cdot \phi\left(\vec{\boldsymbol{a}}, \vec{p}\right)\right) \ , \quad (1)$$

where we search among all $|P|^T$ possible sequences of phonemes of length $T$. Even for moderate number of possible phonemes and relatively short sequences the number of possible sequences is huge and the search becomes intractable.

We thus restrict the feature function $\phi(\vec{\boldsymbol{a}}, \vec{p})$ to be a sum of *local* functions, each is only a function of two neighbor phonemes, that is,

$$\phi(\vec{\boldsymbol{a}}, \vec{p}) = \sum_{t=1}^{T-1} \phi(\vec{\boldsymbol{a}}, p_t, p_{t+1}) \ . \quad (2)$$

This restriction over feature functions induces a factorization of the score function,

$$s(\vec{\boldsymbol{a}}, \vec{p}; \boldsymbol{w}) = \boldsymbol{w} \cdot \left\{ \sum_{t=1}^{T-1} \phi\left(\vec{\boldsymbol{a}}, p_t, p_{t+1}\right) \right\} = \sum_{t=1}^{T-1} \left(\boldsymbol{w} \cdot \phi\left(\vec{\boldsymbol{a}}, p_t, p_{t+1}\right)\right) ,$$

which can be exploited for an efficient inference using the Viterbi algorithm in time linear in the length of the sequence $T$ and quadratic in the number of possible phonemes $|P|^2$.

Recently [8, 9], the framework of online confidence weighted (CW) learning for binary classification was introduced. CW learning captures the notion of confidence in a linear classifier by maintaining a Gaussian distribution over the weights with mean $\boldsymbol{\mu} \in \mathbb{R}^D$ and covariance matrix $\Sigma \in \mathbb{R}^{D\times D}$. The values $\mu_k$ and $\Sigma_{k,k}$, respectively, encode the learner's knowledge and confidence in the weight associated with feature $k$: the smaller $\Sigma_{k,k}$, the more confidence the learner has in the mean weight value $\mu_k$. Covariance terms $\Sigma_{k,r}$ capture interactions between weights.

Conceptually, given an acoustic input $\vec{\boldsymbol{a}}$, a Gibbs classifier draws a weight vector $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ and predicts the label according to (1). In practice, however, it can be easier to simply use the averaged weight vector $\boldsymbol{\mu} = \mathrm{E}[\boldsymbol{w}]$ and use the covariance information *only* to estimate the parameters of $\boldsymbol{\mu}$ and not for prediction.

## 2.3. Learning

We work in the online framework, where learning is performed in rounds or iterations. On each round the learner receives an acoustic input of an utterance $\vec{\boldsymbol{a}}_i$ and makes a prediction $\hat{\vec{p}}_i$ according to (1) using its current parameters. It then receives the true phoneme sequence $\vec{p}_i$ and suffers a loss $\ell\left(\vec{p}_i, \hat{\vec{p}}_i\right)$. In our context the loss function is either the number of erroneous frames, $|\{t : \hat{p}_{i,t} \neq p_{i,t}\}|$ or the Levenshtein edit distance between the correct sequence and the predicted one, called phoneme error. (Where $p_{i,t}$ is the phoneme in the t$th$ location in the i$th$ sequence.) The learner uses the current pair $(\vec{\boldsymbol{a}}_i, \vec{p}_i)$ to modify its prediction rule. Its goal is to minimize the cumulative loss, $\sum_t \ell\left(\vec{p}_i, \hat{\vec{p}}_i\right)$.

To complete the description of our algorithm, it remains to show how we modify the parameters $\boldsymbol{\mu}_i$ and $\Sigma_i$ given the i$th$ pair $(\vec{\boldsymbol{a}}_i, \vec{p}_i)$ to obtain $\boldsymbol{\mu}_{i+1}$ and $\Sigma_{i+1}$. We set $\boldsymbol{\mu}_{i+1}$ and $\Sigma_{i+1}$ to be the solution of the following objective function,

$$\mathcal{O}(\boldsymbol{\mu}, \Sigma) = \mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}, \Sigma) \| \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)\right) + \frac{\mathcal{L}_i^2 + \boldsymbol{\delta}_i^\top \Sigma \boldsymbol{\delta}_i}{2C} \quad (3)$$

where

$$\boldsymbol{\delta}_i = \phi(\vec{\boldsymbol{a}}_i, \vec{p}_i) - \phi\left(\vec{\boldsymbol{a}}_i, \hat{\vec{p}}_i\right), \quad (4)$$

is the difference between the feature vector of the correct phoneme sequence and the feature vector of the predicted phone sequence. Since $\boldsymbol{\delta}_i \in \mathbb{R}^D$ and $\Sigma_i \in \mathbb{R}^{D\times D}$ the quantity $\boldsymbol{\delta}_i^\top \Sigma \boldsymbol{\delta}_i$ is a scalar value equals the variance of the score difference of the correct and predicted phone sequences. We also define,

$$\mathcal{L}_i = \mathcal{L}_i(\boldsymbol{\mu}, \boldsymbol{\delta}_i) = \max\left\{0, \ell\left(\vec{p}_i, \hat{\vec{p}}_i\right) - \boldsymbol{\mu} \cdot \boldsymbol{\delta}_i\right\}. \quad (5)$$

In words, $\mathcal{L}_i$ is a function of the mean parameters $\boldsymbol{\mu}$ and the vector $\boldsymbol{\delta}_i$. It equals to the large-margin hinge loss [10] suffered when comparing the score of the correct phoneme sequence and the predicted phoneme sequence. If $\boldsymbol{\mu}_i \cdot \phi(\vec{\boldsymbol{a}}_i, \vec{p}_i) < \boldsymbol{\mu}_i \cdot \phi\left(\vec{\boldsymbol{a}}_i, \hat{\vec{p}}_i\right) + \ell\left(\vec{p}_i, \hat{\vec{p}}_i\right)$ then $\mathcal{L}_i$ is proportional to the difference $\ell\left(\vec{p}_i, \hat{\vec{p}}_i\right) + \boldsymbol{\mu}_i \cdot \phi(\vec{\boldsymbol{a}}_i, \vec{p}_i) - \boldsymbol{\mu}_i \cdot \phi\left(\vec{\boldsymbol{a}}_i, \hat{\vec{p}}_i\right)$, and otherwise $\mathcal{L}_i = 0$.

The constant $C > 0$ is a tradeoff parameter used to balance between three desires. First, we want the modified set of parameters to be as close as possible to the current set of parameters, since the current parameters captures all of present knowledge from previous examples. Second, we want to reduce the uncertainty about our prediction. Third, we want the new set of parameters to do well on the current example. In particular, we use the predicted sequence of phonemes $\hat{\vec{p}}_i = \arg\max_{\vec{p}} \boldsymbol{\mu}_i \cdot \phi(\vec{\boldsymbol{a}}_i, \vec{p}_i)$ and modify the parameters $\boldsymbol{\mu}_i$ only if the score of the correct phoneme sequence $\vec{p}_i$ is not greater than the score of the predicted sequence $\hat{\vec{p}}_i$ by a difference of at least $\ell\left(\vec{p}_i, \hat{\vec{p}}_i\right)$, that is if,

$$\boldsymbol{\mu}_i \cdot \phi(\vec{\boldsymbol{a}}_i, \vec{p}_i) < \boldsymbol{\mu}_i \cdot \phi\left(\vec{\boldsymbol{a}}_i, \hat{\vec{p}}_i\right) + \ell\left(\vec{p}_i, \hat{\vec{p}}_i\right). \quad (6)$$

In other words, we not only want the score of the correct phoneme sequence to be the highest, but we also want that the difference between the score of the correct phoneme sequence and the (second) best phoneme sequence to be proportional the loss suffered when predicting the later instead of the former. The higher the value of the loss, the higher the gap we enforce.

To solve the optimization problem of (3) we note that it is convex in both $\boldsymbol{\mu}$ and $\Sigma$ and it is a sum of terms, each is a function of either $\boldsymbol{\mu}$ or $\Sigma$ (but not both), including the first term - the KL divergence. We thus first compute the derivative of (3), set it to zero and solve for $\boldsymbol{\mu}$ to obtain,

$$\boldsymbol{\mu}_{i+1} \leftarrow \boldsymbol{\mu}_i + \mathcal{L}_i \, \beta_i \, \Sigma_i \boldsymbol{\delta}_i, \quad (7)$$

where

$$\beta_i \leftarrow \frac{1}{\boldsymbol{\delta}_i^\top \Sigma_i \boldsymbol{\delta}_i + C}. \quad (8)$$

Note that $\mathcal{L}_i$ and $\beta_i$ are scalars, and their product is the *global* effective learning rate. The product of the matrix $\Sigma_i$ and the vector $\boldsymbol{\delta}_i$ give the effective direction of the update, which can be thought of as individual learning rate per features (see also Sec. 2.4).

Finally, we also compute the derivative of (3) with respect to $\Sigma$, set it to zero and solve for $\Sigma$ to obtain,

$$\Sigma_{i+1}^{-1} = \Sigma_i^{-1} + \frac{\boldsymbol{\delta}_i \boldsymbol{\delta}_i^\top}{C}. \quad (9)$$

Using the Woodbury [11] identity we can also rewrite the update for $\Sigma$ in non-inverted form:

$$\Sigma_{i+1} = \Sigma_i - \beta_i \left(\Sigma_i \boldsymbol{\delta}_i \boldsymbol{\delta}_i^\top \Sigma_i\right), \quad (10)$$

where $\beta_i$ was defined above in (8). We note in passing that both the update of $\Sigma$ (10) and the update of its inverse (9) are modified by adding an (rank-one) outer-product symmetric matrix, and thus the covariance parameters remains both symmetric and positive semi-definite. In the former case we add a quantity proportional to the outer-product of $\Sigma_i \boldsymbol{\delta}_i$ with itself, and in the later case, the outer product of $\boldsymbol{\delta}_i$ with itself.

To conclude, the algorithm maintains a pair of a vector $\boldsymbol{\mu}_i$ and a matrix $\Sigma_i$. On the i$th$ iteration it receives an pair of acoustic representation and a corresponding phoneme sequence $(\vec{\boldsymbol{a}}_i, \vec{p}_i)$. It uses the mean parameters to make a prediction $\hat{\vec{p}}_i = f(\vec{\boldsymbol{a}}_i, \boldsymbol{\mu}_i)$ (1) and suffers loss $\ell\left(\vec{p}_i, \hat{\vec{p}}_i\right)$. If the difference between the score of the correct sequence $\vec{p}_i$ and the predicted phoneme-sequence $\hat{\vec{p}}_i$ is not greater than the loss it updates the mean parameters using (7),(8). Otherwise the mean parameters are not modified, that is $\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i$. It updates the covariance matrix using either (9) or (10),(8). We initialize $\boldsymbol{\mu}_1 = \mathbf{0}$ and $\Sigma = I \in \mathbb{R}^{D\times D}$ (identity matrix).

|  | Frame Error Rate | Phone Error Rate | Ins. | Del. | Sub. |
|---|---|---|---|---|---|
| HMM[6] | 39.3 | 42.0 | - | - | - |
| HMM[7] | - | 40.9 | 3.6 | 10.5 | 26.8 |
| KSBSC[7] | - | 45.1 | 5.9 | 10.8 | 28.4 |
| CSS[6] | **28.3** | 33.5 | - | - | - |
| PA | 30.0 | 33.4 | 6.3 | 6.1 | 18.8 |
| DROP | 29.2 | **31.1** | 4.2 | 7.5 | 18.1 |
| PROJECT | 30.0 | 31.7 | 4.5 | 7.4 | 18.3 |

**Table 1**. Frame error rates, phone error rates, insertions, deletions and substitutions on the TIMIT test set for seven methods: two different HMMs models (taken from [6, 7]), three single learning-rate learning methods (two reported in [6, 7] and PA) and the two methods described in this paper.

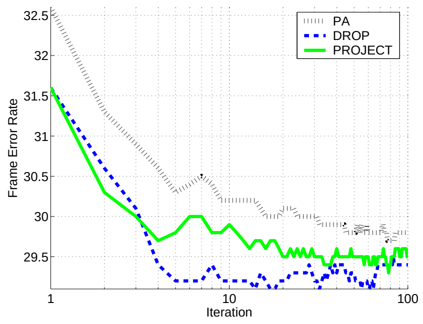


**Fig. 1**. Frame error rate against iteration for three algorithms.

### 2.4. Diagonalization

The above derivation yields a full covariance matrix, since it is a sum of rank-one matrices. For the purpose of ASR full matrices are not practical since the the number of features the function $\phi$ employ (its dimension) is in the order of hundreds. We thus restrict ourself to diagonal matrices in one of two ways. One option is to update $\Sigma_{i+1}$ directly using (10) and then *drop* all the non-diagonal elements. We refer to this method as drop. A second option is to compute the inverse of the covariance $\Sigma_{i+1}^{-1}$ using (9), then drop its non-diagonal elements and compute the inverse of the resulting diagonal matrix. We refer to this method as project since it is equivalent to a projection of the full matrix onto the set of diagonal matrices using the KL divergence. Below we report the results for both updates.

### 2.5. Averaging

Given a set of $N$ pairs $\{(\vec{\boldsymbol{a}}_i, \vec{p}_i)\}_1^N$, we iterate over its elements (possibly $M$ epochs) and employ the update rule on each pair. One common procedure to evaluate the algorithm is using the latest parameters $\boldsymbol{\mu}_{NM}$ to classify unseen utterances (test set). Other alternatives were described and analyzed in [12]. In particular, one option is to use the average of $\boldsymbol{\mu}_1 \ldots \boldsymbol{\mu}_{MN}$ rather than just its last element. A theoretical analysis shows that with i.i.d. assumptions over the data, the average estimate is optimal, in the sense that, with high probability the loss suffered over new speech utterances will be small.

## 3. EXPERIMENTS

We evaluate the performance of our proposed methods using the TIMIT [13] speech corpus. We followed standard partition [6] of the corpus into a training, test and validation set of $3,696$, $400$ and $192$ utterances respectively ($1.1M$,$120K$,$57K$ frames). We followed previous processing of the data [6] and computed a 39-dimensional Mel-frequency cepstrum coefficients with their first and second derivatives. We mapped TIMIT set of 61 phonemes into 48.

The features $\phi(\vec{\boldsymbol{a}}, \vec{p})$ (see (2)) are of one of two types. Either $\phi_n(\vec{\boldsymbol{a}}_t, p)$ (node features) for all $t = 1 \ldots T$ and some phoneme $p$ or $\phi_e(p_t, p_{t+1})$ (edge features) for all $t = 1 \ldots T-1$. The first type captures the relation between a possible phoneme and the corresponding acoustic input, and intuitively is the equivalent of the emission probabilities of HMMs. The second type capture the dependency between two adjacent phonemes and intuitively is the equivalent of the transition probabilities in HMMs. In particular, for the first type we used the 39 acoustic features, their square and the products of all unordered pairs. For the second type we used a single fixed feature.

We evaluated three algorithms, drop and project described above and the passive-aggressive algorithm [14, 7] (PA) which we describe now. The algorithm is an online algorithm that maintains a weight vector $\boldsymbol{w} \in \mathbb{R}^D$ and uses the same prediction function as in (1). To update its parameters it first computes the phoneme-sequence with the highest score $\hat{\vec{p}}_i$ and sets $\boldsymbol{w}_{i+1}$ to be $\boldsymbol{w}_i + \alpha_i \boldsymbol{\delta}_i$ (defined in (4)) where

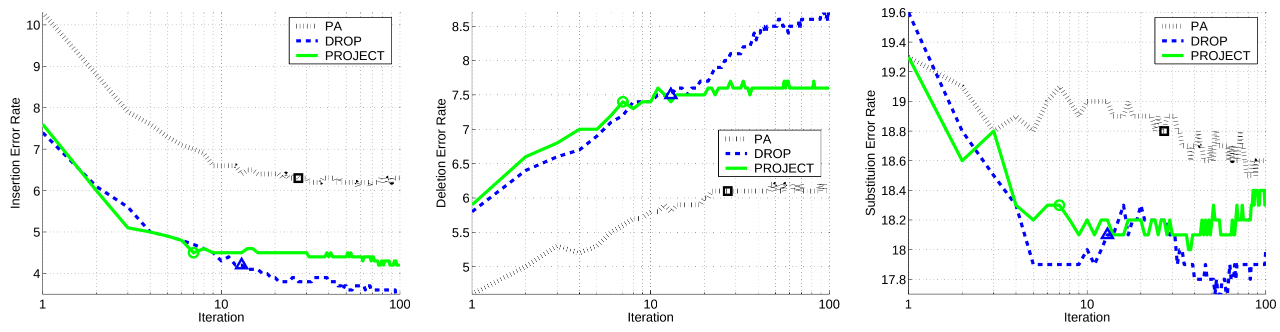
$$\alpha_i = \min\left\{ C, \frac{\max\left\{0, \ell\left(\vec{p}_i, \hat{\vec{p}}_i\right) - \boldsymbol{w}_i \cdot \boldsymbol{\delta}_i\right\}}{\|\boldsymbol{\delta}_i\|^2} \right\} ,$$

(see [14, sec. 8] and [7, sec. 3] for details). The PA algorithm in conjunction with our choice of features is denoted by PA (as opposed to the PA algorithm employed with kernels mentioned below). All algorithms used averaging and were run for 100 epochs with 8 different values for the parameter $C$. For each algorithm we picked the best value of $C$ and epoch using the validation set and report the results on the test set. We also quote the results of two HMM systems taken from [7] and [6] (single state which corresponds to out setting), the Kernel-based recognizer trained with a PA algorithm [7] denoted by the initials of the authors KSBSC, and CD-HMM [6] denoted similarly by CSS.

Following [6] we evaluated the algorithms using two metrics: the frame error rate, which is the fraction of misclassified frames, and the phoneme error rate, computed using the Levenshtein distance. The later reflects better our ultimate goal of speech recognition. In both cases we followed a common practice [15] and mapped the 48 states into 39 categories.

Table 1 summaries our results. The first column shows the frame error rate and the second the phone error rate which is the sum of the fraction of insertions (Ins.), deletions (Del.) and substitutions (Sub.) as computed by the Levenshtein distance. The results show that the CD-HMMs trained with an online method have the smallest frame error rate, although drop is not far beyond. However, drop and project have a significantly lower phoneme error rate compared to all other methods. Furthermore, the phone error rate of the drop algorithm is even better than the performance of CD-HMMS with two mixture components, although for prediction it uses much less parameters.

Interestingly, HMM has the lowest insertion rate of 3.6 while drop is not far beyond with a value of 4.2, and PA has the lowest deletions rate of 6.1 compared with 7.4 of drop. However, drop compensate these gaps with a relatively low substitution rate of 18.1.



(a) Insertion (b) Deletion (c) Substitution

**Fig. 2**. The three components of the Levenshtein distance over the test set against the number of iterations for three algorithms: PA, DROP and PROJECT. The marker indicates the best iteration using the frame error rate on the development set.

Fig. 1 shows the frame error rate of three algorithms against the number of iterations. We observe that `drop` and `project` achieves their best performance after about 10 iterations, while `PA` did not converge even after 100 iterations. Similar phenomena was reported by [6, Figure. 3].

To better understand the performance of the algorithms for phoneme sequence classification we plot in Fig. 2 insertion, deletion and substitution rate the of three algorithms against the number of iterations. The marker indicates the iteration chosen according the frame error rate on the validation data. Interestingly, both the insertion rate and substitution rate decrease with more epochs over the training data, while the deletion rate *increases* with more epochs. Furthermore, for `project` all three measures seems to converge after about 10 epochs. Also as `drop` makes more passes it continues to reduce the insertion error rate for the cost of further increasing deletion error rate. Finally, `PA` converges both for insertion and deletion after about 10 epochs, yet its substitution error does not seems to have a monotone behaviour. We are still in investigating the cause of these observations.

Finally (not shown), we found empirically that `project` was more sensitive to the value of the parameter $C$, compared to `drop` and `PA`, which showed a close to constant performance across large range of values.

## 4. DISCUSSION

Our choice of model, trained either by `drop`, `project` or `PA`, uses the same information as a single mixture component CD-HMM. All use a single parameter to model phoneme transition and first and second order functions of the 39-MFCC coefficients to model observation-phoneme relation. Still, the difference in performance between all training methods is remarkable. Standard discriminative online methods (which uses a single learning rate) obtain lower error rates than traditional methods which optimize the likelihood and are faster to train. Per-feature learning rate methods perform even better in terms of phoneme error rate and converge even faster. We plan to lift our methods to train models equivalent to HMMs with more than one mixture when modeling each phoneme.

## 5. REFERENCES

[1] L. Rabiner and B.H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall PTR, 1993.

[2] A. Gunawardana, M. Mahajan, A. Acero, and J.C. Platt, “Hidden conditional random fields for phone classification,” in *INTERSPEECH*, 2005, pp. 1117–1120.

[3] E. McDermott, T.J. Hazen, J. Le Roux, A. Nakamura, and S. Katagiri, “Discriminative training for large vocabulary speech recognition using minimum classification error,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 1, 2007.

[4] F. Sha and L. Saul, “Comparison of large margin training to other discriminative methods for phonetic recognition by hidden markov models,” in *ICASSP*, 2007.

[5] D. Yu, L. Deng, X. He, and A. Acero, “Large-margin minimum classification error training for large-scale speech recognition tasks,” in *ICASSP*, 2007.

[6] C.C. Cheng, F. Sha, and L. Saul, “A fast online algorithm for large margin training of continuous-density hidden markov models,” in *INTERSPEECH*, 2009.

[7] J. Keshet, S. Shalev-Shwartz, S. Bengio, Y. Singer, and D. Chazan, “Discriminative kernel-based phoneme sequence recognition,” in *INTERSPEECH*, 2006.

[8] K. Crammer, A. Kulesza, and M. Dredze, “Adaptive regularization of weight vectors,” in *NIPS 23*, 2009.

[9] K. Crammer, M. Dredze, and A. Kulesza, “Multi-class confidence weighted algorithms,” in *EMNLP*, 2009.

[10] B. Taskar, C. Guestrin, and D. Koller, “Max-margin markov networks,” in *NIPS*. 2003, MIT Press.

[11] Kaare Brandt Petersen and Michael Syskind Pedersen, *The Matrix Cookbook*, 2007.

[12] N. Cesa-Bianchi, A. Conconi, and C. Gentile, “On the generalization ability of on-line learning algorithms,” *IEEE Tran. on Information Theory*, vol. 50, no. 9, pp. 2050–2057, 2004.

[13] L. Lamel, R. Kassel, and S. Seneff, “Speech database development: design and analysis of the acoustic-phonetic corpus,” in *Proc. DARPA Speech Recognition Workshop*, 1986.

[14] K. Crammer, O. Dekel, J. Keshet, S.Shalev-Shwartz, and Y. Singer, “Online passive-aggressive algorithms,” *JMLR*, vol. 7, pp. 551–585, 2006.

[15] K-F Lee and H-W Hon, “Speaker-independent phone recognition using hidden markov models,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 2, pp. 1641–1648, 1989.