# Direct Loss Minimization for Structured Prediction

**David McAllester** MCALLESTER@TTIC.EDU
**Tamir Hazan** TAMIR@TTIC.EDU
**Joseph Keshet** JKESHET@TTIC.EDU
TTI Chicago
6045 S. Kenwood Ave, Chicago IL. 60637

## Abstract

In discriminative machine learning one is interested in training a system to optimize a certain desired measure of performance such as the BLEU score in machine translation or the intersection-over-union score in the PASCAL segmentation evaluation. We propose here a perceptron-like learning method based on computing a difference of feature vectors between two inferred output values where at least one of the outputs is inferred by loss-adjusted inference. The main contribution of this paper is a theorem directly relating updates of this form to the gradient of the given loss function with respect to the system parameters. This provides a theoretical foundation for certain training methods which have already gained widespread use in machine translation. Empirical results on phonetic alignment are also given here surpassing all previously reported results on this problem.

## 1. Introduction

Many modern software systems compute a result as the solution, or approximate solution, to an optimization problem. For example, modern machine translation systems convert an input word string into an output word string in a different language by approximately optimizing a score defined on the input-output pair. Optimization underlies the leading approaches in a wide variety of computational problems including problems in computational linguistics, computer vision, genome annotation, advertisement placement, and speech recognition.

Appearing in *Proceedings of the 26$^{th}$ International Conference on Machine Learning*, Haifa, Israel, 2010. 

In many optimization-based software systems, such as systems for machine translation, one must design the objective function as well as the optimization algorithm. Here we consider a parameterized objective function and the problem of setting the parameters of the objective in such a way that the resulting optimization-driven software system performs well. For example, in a machine translation system we want the objective function to be such that optimizing that objective yields accurate translations.

The abstract problem considered here is easily formalized. Let us denote by $\mathcal{X}$ the abstract set of possible inputs and denote by $\mathcal{Y}$ the set of possible outputs. For example, we might be given a large corpus of translation pairs $(x, y)$ where $x \in \mathcal{X}$ is an English sentence and $y \in \mathcal{Y}$ is a French sentence. We assume an objective function $s_w : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ parameterized by a vector $w \in \mathbb{R}^d$ such that for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ we have a score $s_w(x, y)$. The parameter setting $w$ determines a mapping from input $x$ to output $y_w(x) \in \mathcal{Y}$ is defined as follows:

$$y_w(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} s_w(x, y). \tag{1}$$

In our machine translation example we want to set the parameter vector $w$ so that the system produces good translations. Finding good parameter settings for the parameters of an objective function is also important in applications in computer vision, natural language processing, genomics, and speech recognition.

In a particular application, such as machine translation, one typically has access to a corpus of "training pairs" of the form $(x, y)$ where $x$ is an input and $y$ is a desirable output for input $x$. In machine translation we would have that $(x, y)$ is a translation pair where $x$ is, say, an English sentence and $y$ a French translation of $x$. We can formalize what we mean by a "good" machine translation system by assuming that there exists some unknown probability distribution $\rho$ over translation pairs and a loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ which

gives a cost $L(y, \hat{y}) \geq 0$ for producing output $\hat{y}$ when the desired output is $y$. We then want to set $w$ so as to minimize the expected loss.

$$w^* = \operatorname*{argmin}_w \mathbb{E}_{(x,y)\sim\rho} \left[ L(y, y_w(x)) \right] \tag{2}$$

In machine learning terminology we refer to (1) as inference and (2) as training.

Unfortunately the objective function given by (2) is typically non-convex and, furthermore, there are typically no known algorithms with good approximation guarantees to (2). However, it is nonetheless common to replace the objective in (2) with some form of convex relaxation. For example, several forms of convex relaxations are used in corresponding forms of structured support vector machines (structured SVMs) (Taskar et al., 2003; Tsochantaridis et al., 2005) as described in section 2. But is should be noted that replacing the objective in (2) with a convex relaxation leads to inconsistency — the optimum of the relaxation is different from the optimum of (2). We are unaware of any approximation guarantees for the relaxations of (2) to the various forms of structured hinge loss.

An obvious alternative to using a convex relaxation is to simply perform gradient descent directly on the objective in (2). In a particular application area, such as machine translation or computer vision, it seems possible that the local minima problem of non-convex optimization might be less serious than the inconsistencies introduced by a convex relaxation.

Unfortunately, direct gradient descent on (2) is conceptually puzzling in the case where the output space $\mathcal{Y}$ is discrete. In this case the output $y_w(x)$ is not a differentiable function of $w$. As one smoothly changes $w$ the output $y_w(x)$ jumps discontinuously between discrete output values. So one cannot write $\nabla_w \mathbb{E}\left[L(y, y_w(x))\right]$ as $\mathbb{E}\left[\nabla_w L(y, y_w(x))\right]$. However, when the input space $\mathcal{X}$ is continuous the gradient $\nabla_w \mathbb{E}\left[L(y, y_w(x))\right]$ can exist even when the output space $\mathcal{Y}$ is discrete. The main results of this paper is a perceptron-like method of performing direct gradient descent on (2) in the case where the output space is discrete but the input space is continuous.

After formulating our method we discovered that closely related methods have recently become popular for training machine translation systems (Liang et al., 2006; Chiang et al., 2009). Although machine translation has discrete inputs as well as discrete outputs, the training method we propose can still be used, although without theoretical guarantees.

We also present empirical results on the use of this method in a phoneme alignment task where it achieves the best known results on this problem.

## 2. Perceptron-Like Training Methods

Perceptron-like training methods are generally formulated for the case where the scoring function is linear in $w$. In other words, we assume that the scoring function can be written as follows where $\boldsymbol{\phi} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$ is called a *feature map*.

$$s_w(x, y) = w^\top \boldsymbol{\phi}(x, y) \tag{3}$$

Because the feature map $\boldsymbol{\phi}$ can itself be nonlinear, and the feature vector $\boldsymbol{\phi}(x, y)$ can be very high dimensional, objective functions of the form (3) are highly expressive and are widely used in the applications mentioned above.

Here we will formulate perceptron-like training in the data-rich regime where we have access to an unbounded sequence $(x_1, y_1)$, $(x_2, y_2)$, $(x_3, y_3)$, ... where each $(x_t, y_t)$ is drawn IID from the distribution $\rho$. In the basic multiclass perceptron algorithm (Collins, 2004) one constructs a sequence of parameter settings $w^0$, $w^1$, $w^2$, ... where $w^0 = 0$ and $w^{t+1}$ is defined as follows.

$$w^{t+1} = w^t + \boldsymbol{\phi}(x_t, y_t) - \boldsymbol{\phi}(x_t, y_{w^t}(x_t)) \tag{4}$$

Note that if $y_w(x_t) = y_t$ then no update is made and we have $w^{t+1} = w^t$. If $y_w(x_t) \neq y_t$ then the update changes the parameter vector in a way that favors $y_t$ over $y_w(x_t)$. If there exists $w$ that always well-separates the desired output[1] then the perceptron update rule will eventually lead to a parameter setting with zero loss assuming L(y,y)=0. Note, however, that the basic perceptron update does not involve the loss function $L$. Hence it cannot be expected to optimize the training objective (2) when perfect performance cannot be achieved.

A loss-sensitive perceptron-like algorithm can be derived from the structured hinge loss of a margin-scaled structured SVM (Tsochantaridis et al., 2005). The optimization problem for margin-scaled structured hinge loss can be defined as follows.

$$w^* = \operatorname*{argmin}_w \mathbb{E}\left[ \max_{\tilde{y}\in\mathcal{Y}} \left( L(y, \tilde{y}) - w^\top \left( \boldsymbol{\phi}(x, y) - \boldsymbol{\phi}(x, \tilde{y}) \right) \right) \right] \tag{5}$$

This is a convex relaxation of (2). To see that it is convex in $w$ it suffices to note that it is a maximum of functions each of which is linear in $w$ (and a maximum of convex functions is convex). To see that it is a relaxation of (an upper bound on) $L(y, y_w(x))$ we note

[1] There exists $w$, with $||w|| = 1$, and $\gamma > 0$ such that with probability one over the draw of $(x, y)$ we have $w^\top \boldsymbol{\phi}(x, y) - w^\top \boldsymbol{\phi}(x, \tilde{y}) \geq \gamma$ for all $\tilde{y} \in \mathcal{Y}$ with $\tilde{y} \neq y$.

the following.

$$\begin{aligned} w^\top \boldsymbol{\phi}(x,y) &\leq w^\top \boldsymbol{\phi}(x, y_w(x)) \\ L(y, y_w(x)) &\leq L(y, y_w(x)) - w^\top (\boldsymbol{\phi}(x,y) - \boldsymbol{\phi}(y, y_w(x))) \\ &\leq \max_{\tilde{y} \in \mathcal{Y}} L(y, \tilde{y}) - w^\top \left(\boldsymbol{\phi}(x,y) - \boldsymbol{\phi}(x, \tilde{y})\right) \end{aligned}$$

As noted in the introduction, we are unaware of any approximation guarantees for this relaxation. At least one other convex relaxation is often considered, the so-called slack-scaled hinge loss, which was also introduced in (Tsochantaridis et al., 2005). Here however, we consider only the margin-scaled version as this version is most closely related to the direct loss optimization considered here.

We can optimize (5) with stochastic sub-gradient descent. To do this we compute a sub-gradient of the objective in (5) by first computing the value of $\tilde{y}$ which achieves the maximum.

$$\begin{aligned} y^t_{hinge} &= \operatorname*{argmax}_{\tilde{y} \in \mathcal{Y}} L(y_t, \tilde{y}) - w^\top \left(\boldsymbol{\phi}(x_t, y_t) - \boldsymbol{\phi}(x_t, \tilde{y})\right) \\ &= \operatorname*{argmax}_{\tilde{y} \in \mathcal{Y}} w^\top \boldsymbol{\phi}(x_t, \tilde{y}) + L(y_t, \tilde{y}) \end{aligned} \quad (6)$$

The sub-gradient is then taken to be the gradient of the objective in (5) with respect to the parameter $w$ where $\tilde{y}$ is held fixed at $y^t_{\text{hinge}}$. This yields the following perceptron-like update rule where the update direction is the negative of the sub-gradient of the loss and $\eta^t$ is a learning rate.

$$w^{t+1} = w^t + \eta^t \left(\boldsymbol{\phi}(x_t, y_t) - \boldsymbol{\phi}(x_t, y^t_{\text{hinge}})\right) \quad (7)$$

It is useful to compare (7) with (4). The main difference is that in (7) the label $y^t_{\text{hinge}}$ is defined by (6) while in (4) we have that $y_w$ is defined by (1). Equation (6) defines a form of *loss-adjusted* inference. So (7) is at least influenced to some extent by the loss function $L$.

The algorithm proposed here uses the following perceptron-like update rule where $\eta^t$ is a time-varying learning rate and $\epsilon^t$ is a time-varying accuracy parameter.

$$w^{t+1} = w^t + \eta^t \left(\boldsymbol{\phi}(x_t, y^t_{\text{direct}}) - \boldsymbol{\phi}(x_t, y_{w^t}(x_t))\right) \quad (8)$$

$$y^t_{\text{direct}} = \operatorname*{argmax}_{\tilde{y} \in \mathcal{Y}} (w^t)^\top \boldsymbol{\phi}(x_t, \tilde{y}) - \epsilon^t L(y, \tilde{y}) \quad (9)$$

The most significant difference between (8) and (7) is that (8) is a difference of feature vectors between two *inferred* output values. The feature vector of the given desired output $y_t$ is not used in (8). It is also useful to compare the two forms of loss-adjusted inference defined by (9) and (6). Note that (9) subtracts the loss while (6) adds the loss. At a theoretical level this difference is not significant — our update can be formulated in terms of either form of loss-adjustment. However, we have found that computing a loss-adjusted label by subtracting loss works better in practice. Intuitively this is because subtracting loss causes loss-adjusted inference to operate is a desirable region of the search space rather than an undesirable region. In the update (7) we view $y_t$ as the desirable value and $y^t_{\text{hinge}}$ as the undesirable value. In the update (8) we view $y^t_{\text{direct}}$ as the desirable value and $y_{w^t}(x_t)$ as the undesirable value. The update (8) can be intuitively motivated by saying that we wish to change the parameter settings in a way that moves the output values toward values of lower loss. The parameter $\epsilon^t$ is an accuracy parameter which is motivated by the main results of this paper. The main result directly relates (8) to the gradient of the objective in (2).

## 3. The Loss Gradient Theorem

We are now ready for the main result of this paper.

**Theorem 1.** *For a finite set $\mathcal{Y}$ of possible output values, and under a mild general position assumption stated below, we have the following.*

$$\begin{aligned} &\nabla_w \mathbb{E}_{(x,y)\sim\rho} \left[L(y, y_w(x))\right] \\ &= \lim_{\epsilon \to 0} \frac{1}{\epsilon} \mathbb{E} \left[\boldsymbol{\phi}(x, y_w(x)) - \boldsymbol{\phi}(x, y_w(x, y, \epsilon))\right] \end{aligned} \quad (10)$$

*where*

$$y_w(x, y, \epsilon) = \operatorname*{argmax}_{\tilde{y} \in \mathcal{Y}} w^\top \boldsymbol{\phi}(x, \tilde{y}) - \epsilon \cdot L(y, \tilde{y}). \quad (11)$$

One should note that the update equation (8) directly corresponds to a stochastic gradient step where we are estimating the gradient by a stochastic sample of the right hand side of (10). To make the update we select a finite $\epsilon$, take a stochastic sample of the pair $(x, y)$, and perform an update in the negative gradient direction, i.e., the negative of the direction of the feature difference given in theorem 1.

The intuition behind this theorem is that the mapping $y_w : \mathcal{X} \to \mathcal{Y}$ partitions the set $\mathcal{X}$ into a finite number of cells with one cell for each possible value of $y_w(x)$. The loss is determined by this cell structure. When $\mathcal{X}$ is a continuous space, and $w$ is in general position (defined below) the loss is a differentiable function of $w$. Furthermore, the gradient of the loss with respect to $w$ is determined by a certain integral over the boundary of the cell structure where the boundary of the cell structure is the set of inputs $x$ where two or more possible outputs both achieve a maximal score. In fact

both of the quantities in theorem 1 can be expressed as integrals over the boundary of the cell structure. For a finite $\epsilon$ each integral involves a product of a width of a boundary layer and a quantity being integrated over the boundary. In the integral for the left hand side of (10) the width of the boundary layer is determined by a difference of feature vectors and the quantity being integrated is a difference of loss values. In integral for the right hand side of (10) the width of the boundary layer is given by a difference of loss values and quantity being integrated is a difference of feature vectors. In general position the value of two integrals is the same.

The proof of the theorem is rather technical and is given in an appendix. In addition to the assumption that $\mathcal{Y}$ is finite we assume that $w$ and $\rho$ are in general position by which we mean that for all $v, u, s \in \mathcal{Y}$ we have that the distribution on the three random variables $w^\top \boldsymbol{\phi}(x, v)$, $w^\top \boldsymbol{\phi}(x, u)$, $w^\top \boldsymbol{\phi}(x, s)$ conditioned on $y = v$ and $y_w(x) = u$ defines a bounded density on $\mathbb{R}^3$.

## 4. Approximate Inference

In many applications the inference problem (1) is intractable. Most commonly we have some form of graphical model. In this case the score $w^\top \Phi(x, y)$ is defined as the negative energy of a Markov random field (MRF) where $x$ and $y$ are assignments of values to nodes of the field. Finding a lowest energy value for $y$ in (1) in a general graphical model is NP-hard.

A common approach to an intractable optimization problem is to define a convex relaxation of the objective function. In the case of graphical models this can be done by defining a relaxation of a marginal polytope (Wainwright & Jordan, 2008). The details of the relaxation are not important here. At a very abstract level the resulting approximate inference problem can be defined as follows where the set $R$ is a relaxation of the set $\mathcal{Y}$.

$$r_w(x) = \operatorname*{argmax}_{r \in \mathcal{R}} w^\top \boldsymbol{\phi}(x, r) \tag{12}$$

We assume that for $y \in \mathcal{Y}$ and $r \in \mathcal{R}$ we can assign a loss $L(y, r)$. In the case of a relaxation of the marginal polytope of a graphical model we can take $L(y, r)$ to be the expectation over a random rounding of $r$ to $\tilde{y}$ of $L(y, \tilde{y})$. For many loss functions, such as weighted Hamming loss, one can compute $L(y, r)$ efficiently. The training problem is then defined by the following equation.

$$w^* = \operatorname*{argmin}_w \mathbb{E}\left[L(y, r_w(x))\right] \tag{13}$$

Note that (13) directly optimizes the performance of the approximate inference algorithm. The parameter setting optimizing approximate inference might be significantly different from the parameter setting optimizing the loss under exact inference.

The proof of theorem 1 immediately generalizes to (13) provided that $R$ is a finite set, such as the set of vertices of a relaxation of the marginal polytope. So we immediately get the following generalization of theorem 1.

$$\nabla_w \mathbb{E}_{(x,y)\sim\rho}\left[L(y, r_w(x))\right]$$
$$= \lim_{\epsilon \longrightarrow 0} \frac{1}{\epsilon} \mathbb{E}\left[\boldsymbol{\phi}(x, r_w(x)) - \boldsymbol{\phi}(x, r_w(x, y, \epsilon))\right]$$

where

$$r_w(x, y, \epsilon) = \operatorname*{argmax}_{\tilde{r} \in \mathcal{R}} w^\top \boldsymbol{\phi}(x, \tilde{r}) - \epsilon \cdot L(y, \tilde{r})$$

This suggests the following perceptron-like update equation for optimizing the performance of approximate inference.

$$w^{t+1} = w^t + \eta^t \left(\boldsymbol{\phi}(x_t, r_{w^t}(x_t, y_t, \epsilon^t)) - \boldsymbol{\phi}(x_t, r_{w^t}(x_t))\right) \tag{14}$$

## 5. Latent Structure

In many applications it is useful to introduce hidden information into the inference optimization problem. For example, in machine translation we might want to construct parse trees for the both the input and output sentence. In this case the inference equation can be written as follows where $h$ is the hidden information.

$$y_w(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} \max_{h \in \mathcal{H}} w^\top \boldsymbol{\phi}(x, y, h) \tag{15}$$

In this case we can take the training problem to again be defined by (2) but where $y_w(x)$ is defined by (15).

Latent information can be handled by the equations of section 4 but where $R$ is reinterpreted as the set of pairs $(y, h)$ with $y \in \mathcal{Y}$ and $h \in \mathcal{H}$. In this case $L(y, r)$ has the form $L(y, (y', h))$ which we can take to be equal to $L(y, y')$.

## 6. Experiments

In this section we present empirical results on the task of phoneme-to-speech alignment. Phoneme-to-speech alignment is used as a tool in developing speech recognition and text-to-speech systems. In the phoneme alignment problem each input $\overline{x}_t$ is a pair $((p_1^t, \ldots, p_K^t)(s_1^t, \ldots, s_T^t))$ where $p_i^t$ is a phoneme name, with $p_i^t \in \mathcal{P}$ where $\mathcal{P}$ is a finite set of phoneme names, and where $s_j^t \in \mathbb{R}^d$ is a mel-frequency cepstral coefficient (MFCC) vector. The lengths $K$ and $T$ can be different for different inputs although typically we have

*Table 1.* Percentage of correctly positioned phoneme boundaries, given a predefined tolerance on the TIMIT corpus. Results are reported on the whole TIMIT test-set (1344 utterances).

| | $t \leq 10$ms | $t \leq 20$ms | $t \leq 30$ms | $t \leq 40$ms |
|---|---|---|---|---|
| Hosom (2009) | 79.30 | 93.36 | 96.74 | 98.22 |
| Keshet (2007) | 80.0 | 92.3 | 96.4 | 98.2 |
| Direct loss min. $\tau-$alignment | **86.01** | 94.08 | 97.08 | 98.44 |
| Direct loss min. $\tau-$insensitive | 85.72 | **94.21** | **97.21** | **98.60** |

$T$ significantly larger than $K$. The goal is to generate an alignment between the two sequences in the input. Sometimes this task is called *forced-alignment* because one is forced to interpret the given acoustic signal as the given phoneme sequence. The output $\overline{y}_t$ is a sequence $(y_1^t, \ldots, y_K^t)$, where $1 \leq y_i^t \leq T$ is an integer giving the start frame in the acoustic sequence of the $i$-th phoneme in the phoneme sequence. Hence the $i$-th phoneme starts at frame $y_i^t$ and ends at frame $y_{i+1}^t - 1$.

Two types of loss functions are used to quantitatively assess alignments. The first loss is called the $\tau$-*alignment loss* and it is defined as

$$L^{\tau\text{-alignment}}(\bar{y}, \bar{y}') = \frac{1}{|\bar{y}|} \left|\{k : |y_k - y'_k| > \tau\}\right|. \quad (16)$$

In words, this loss measures the average number of times the absolute difference between the predicted alignment sequence and the manual alignment sequence is greater than $\tau$. This loss with different values of $\tau$ was used to measure the performance of the learned alignment function in (Brugnara et al., 1993; Toledano et al., 2003; Hosom, 1998). The second loss, called $\tau$-*insensitive loss* was proposed in (Keshet et al., 2007) as is defined as follows.

$$L^{\tau\text{-insensitive}}(\bar{y}, \bar{y}') = \frac{1}{|\bar{y}|} \max\left\{|y_k - y'_k| - \tau, 0\right\} \quad (17)$$

This loss measures the average disagreement between all the boundaries of the desired alignment sequence and the boundaries of predicted alignment sequence where a disagreement of less than $\tau$ is ignored. Note that $\tau$-insensitive loss is continuous and convex while $\tau$-alignment is discontinuous and non-convex.

Our experiments are on the TIMIT speech corpus for which there are published benchmark results. The corpus contains aligned utterances each of which is a pair $(\overline{x}, \overline{y})$ where $\overline{x}$ is a pair of a phonetic sequence and an acoustic sequence and $\overline{y}$ is a desired alignment. We divided the training portion of TIMIT (excluding the SA1 and SA2 utterances) into three disjoint parts containing 1500, 1796, and 100 utterances, respectively. The first part of the training set was used to train a phoneme frame-based classifier, which given a speech frame and a phoneme, outputs the confident that the phoneme was uttered in that frame. The phoneme frame-based classifier is then used as part of a seven dimensional feature map $\boldsymbol{\phi}(\overline{x}, \overline{y}) = \boldsymbol{\phi}((\bar{\mathbf{s}}, \bar{p}), \bar{y})$ as described in (Keshet et al., 2007). The feature set used to train the phoneme classifier consisted of the Mel-Frequency Cepstral Coefficient (MFCC) and the log-energy along with their first and second derivatives ($\Delta$+$\Delta\Delta$) as described in (Keshet et al., 2007). The classifier used a Gaussian kernel with $\sigma^2 = 19$ and a trade-off parameter $C = 5.0$. The complete set of 61 TIMIT phoneme symbols were mapped into 39 phoneme symbols as proposed by (Lee & Hon, 1989), and was used throughout the training process.

The seven dimensional weight vector $w$ was trained on the second set of 1796 aligned utterances. We trained twice, once for $\tau$-sensitive loss and once for $\tau$-insensitive loss, with $\tau = 10$ millisecond in both cases. Training was done by first setting $w^0 = 0$ and then repeatedly selecting one of the 1796 training pairs at random and performing the update (8) with $\eta^t = 1$ and $\epsilon^t$ set to a fixed value $\epsilon$. It should be noted that if $w^0 = 0$ and $\epsilon^t$ and $\eta^t$ are both held constant at $\epsilon$ and $\eta$ respectively, then the direction of $w^t$ is independent of the choice of $\eta$. These updates are repeated until the performance of $w^t$ on the third data set (the hold-out set) begins to degrade. This gives a form of regularization known as early stopping. This was repeated for various values of $\epsilon$ and a value of $\epsilon$ was selected based on the resulting performance on the 100 hold-out pairs. We selected $\epsilon = 1.1$ for both loss functions.

We scored the performance of our system using $\tau$-sensitive loss on the TIMIT test set of 1344 utterances and with $\tau$ set to each of 10, 20, 30 and 40 milliseconds. As should be expected, for $\tau$ equal to 10 milliseconds the best performance is achieved when the loss used in training matches the loss used in test. Larger values of $\tau$ correspond to a loss function that was not used in training. The results are given in Table 1. We compared our results with (Hosom, 1998), which is an HMM/ANN-based system, and with (Keshet et al., 2007), which is based on structured SVM training trained for $\tau$-insensitive loss. Both systems are considered to be state-of-the-art results on this corpus. As can be seen, our algorithm outperforms the current state-of-the-art results in every tolerance value. Also, as might be expected, the $\tau$-insensitive loss seems more robust to the use of a $\tau$ value at test time that is larger

than the $\tau$ value used in training.

## 7. Open Problems and Discussion

The main result of this paper is the loss gradient theorem of section 3 which is proved in the appendix. This theorem provides a theoretical foundation for perceptron-like training methods with updates computed as a difference between the feature vectors of two different inferred outputs where at least one of those outputs is inferred with loss-adjusted inference. Perceptron-like training methods using feature differences between two inferred outputs have already been shown to be successful for machine translation but theoretical justification has been lacking. We also show the value of these training methods in a phonetic alignment problem.

Although we did not give an asymptotic convergence results it should be straightforward to show that under the update given by (8) we have that $w^t$ converges to a local optimum of the objective provided that both $\eta^t$ and $\epsilon^t$ go to zero while $\sum_t \eta^t \epsilon^t$ goes to infinity. For example one could take $\eta^t = \epsilon^t = 1/\sqrt{t}$.

An open problem is how to properly incorporate regularization in the case where only a finite corpus of training data is available. In our phoneme alignment experiments we trained only a seven dimensional weight vector and early stopping was used as regularization. It should be noted that naive regularization with a norm of $w$, such as regularizing with $\lambda||w||^2$, is nonsensical as the loss $\mathbb{E}\left[L(y, y_w(x))\right]$ is insensitive to the norm of $w$. Regularization is typically done with a surrogate loss function such as hinge loss. Regularization remains an open theoretical issue for direct gradient descent on a desired loss function on a finite training sample. Early stopping may be a viable approach in practice.

Many practical computational problems in areas such as computational linguistics, computer vision, speech recognition, robotics, genomics, and marketing seem best handled by some form of score optimization. In all such applications we have two optimization problems. Inference is an optimization problem (approximately) solved during the operation of the fielded software system. Training involves optimizing the parameters of the scoring function to achieve good performance of the fielded system. We have provided a theoretical foundation for a certain perceptron-like training algorithm by showing that it can be viewed as direct stochastic gradient descent on the loss of the inference system. The main point of this training method is to incorporate domain-specific loss functions, such as the BLEU score in machine translation, directly into the training process with a clear theoretical foundation. Hopefully the theoretical framework provided here will prove helpful in the continued development of improved training methods.

## References


Brugnara, F., Falavigna, D., and Omologo, M. Automatic segmentation and labeling of speech based on hidden markov models. *Speech Communication*, 12: 357–370, 1993.

Chiang, D., Knight, K., and Wang, W. 11,001 new features for statistical machine translation. In *Proc. NAACL, 2009*, 2009.

Collins, M. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Conference on Empirical Methods in Natural Language Processing*, 2002.

Collins, M. Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. In Bunt, Harry, Carroll, John, and Satta, Giorgio (eds.), *New Developments in Parsing Technology*. Kluwer, 2004. Revised version of the paper that appeared at IWPT 2001.

Hosom, J.-P. Speaker-independent phoneme alignment using transition-dependent states. *Speech Communication*, 51:352–368, 1998.

Keshet, J., Shalev-Shwartz, S., Singer, Y., and Chazan, D. A large margin algorithm for speech and audio segmentation. *IEEE Trans. on Audio, Speech and Language Processing*, Nov. 2007.

Lee, K.-F. and Hon, H.-W. Speaker independent phone recognition using hidden markov models. *IEEE Trans. Acoustic, Speech and Signal Proc.*, 37(2): 1641–1648, 1989.

Liang, P., Bouchard-Ct, A., Klein, D., and Taskar, B. An end-to-end discriminative approach to machine translation. In *International Conference on Computational Linguistics and Association for Computational Linguistics (COLING/ACL)*, 2006.

Taskar, B., Guestrin, C., and Koller, D. Max-margin markov networks. In *Advances in Neural Information Processing Systems 17*, 2003.

Toledano, D.T., Gomez, L.A.H., and Grande, L.V. Automatic phoneme segmentation. *IEEE Trans. Speech and Audio Proc.*, 11(6):617–625, 2003.

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

Wainwright, M. J. and Jordan, M. I. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, December 2008.

## A. Proof of The Loss Gradient Theorem

We will consider a form of the loss gradient theorem appropriate for sections 4 and 5. We assume a input set $\mathcal{X}$, a label set $\mathcal{Y}$, and a probability distribution or density (measure) $\rho$ on $\mathcal{X} \times \mathcal{Y}$. We also assume an output space $\mathcal{Z}$, a feature map $\boldsymbol{\phi} : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}^d$, and a loss function $L : \mathcal{Y} \times \mathcal{Z} \to \mathbb{R}$. Our objective is to set $w$ as follows where here, and in the remainder of this section, all expectations are taken over a random draw of $(x, y)$ from $\rho$.

$$w^* = \operatorname*{argmin}_w \mathbb{E}\left[L(y, z_w(x))\right] \tag{18}$$

$$z_w(x) = \operatorname*{argmax}_{z \in \mathcal{Z}} w^\top \boldsymbol{\phi}(x, z) \tag{19}$$

In our proof of the loss gradient theorem we make two assumptions. First, we assume that the label set $\mathcal{Y}$ and the output space $\mathcal{Z}$ are both finite. Second we assume that $w$ and $\rho$ are jointly in "general position" where by this we mean that for any $v \in \mathcal{Y}$ and $z, s, r \in \mathcal{Z}$ the probability measure on the three random variables $w^\top \Phi(x, z)$, $w^\top \Phi(x, s)$ and $w^\top \boldsymbol{\phi}(x, w)$ conditioned on $y = v$ and $z_w(x) = z$ forms a bounded density on $\mathbb{R}^3$. Note that this essentially requires that $\mathcal{X}$ is a continuous space. Also, we must have $w \neq 0$. But nonzero values of $w$ can also fail to be in general position. Under the finiteness and general position assumptions we prove the following.

$$\nabla_w \, \mathbb{E}\left[L(y, z_w(x))\right]$$

$$= \lim_{\epsilon \to 0} \frac{1}{\epsilon} \mathbb{E}\left[\boldsymbol{\phi}(x, z_w(x)) - \boldsymbol{\phi}(x, z_w(x, y, \epsilon))\right]$$

$$z_w(x, y, \epsilon) = \operatorname*{argmax}_{z \in \mathcal{Z}} \; w^\top \Phi(x, z) - \epsilon L(y, z)$$

We will write $f(\epsilon) \sim g(\epsilon)$ to mean that in the limit as $\epsilon$ approaches zero *from above* we have that the ratio $f(\epsilon)/g(\epsilon)$ approaches one. To prove the loss gradient theorem it suffices to prove that for any $\Delta w \in \mathbb{R}^d$ the following two quantities are asymptotically equivalent.

$$\mathbb{E}\left[L(y, o_{w+\epsilon\Delta w}(x)) - L(y, z_w(x))\right] \tag{20}$$

$$(\Delta w)^\top \mathbb{E}\left[\boldsymbol{\phi}(x, z_w(x)) - \boldsymbol{\phi}(x, z_w(x, y, \epsilon))\right] \tag{21}$$

We will now find the following notation useful.

$$\begin{aligned} \Delta\Phi_{z,s}(x) &= \Phi(x, z) - \Phi(x, s) \\ \Delta L_{z,s}(y) &= L(y, z) - L(y, s) \end{aligned}$$

We first note that the directional gradient value (20) can be written as follows.

$$\sum_{v \in \mathcal{Y},\, z,s \in \mathcal{Z}} \mathbb{E}\left[\begin{array}{l} I\left[\begin{array}{rcl} y & = & v \\ z_w(x) & = & z \\ z_{w+\epsilon\Delta w}(x) & = & s \end{array}\right] \\ \\ \Delta L_{s,z}(v) \end{array}\right] \tag{22}$$

By the general position assumption this quantity is asymptotically equivalent to $\epsilon f(v, z, s)$ for some function $f(v, z, s)$ which can be calculated with an appropriate integral over the density on $\mathbb{R}^3$ specified in the assumption. In short, the general position assumption implies that the probability that there is an output value $s$ whose score is $\epsilon$-near the best scoring output declines linearly in $\epsilon$. By a similar observation, the probability that there are two output labels which are both simultaneously $\epsilon$-near the best scoring output label declines as $\epsilon^2$. So up to asymptotic equivalence we can ignore the case where two or more competing outputs are $\epsilon$-close. We then have that the quantity (22) is asymptotically equivalent to the following.

$$\sum_{v \in \mathcal{Y},\, z,s \in \mathcal{Z}} \mathbb{E}\left[\begin{array}{l} I\left[\begin{array}{l} y = v \\ z_w(x) = z \\ w^\top \Delta\boldsymbol{\phi}_{z,s}(x) \\ \in (0, \epsilon(\Delta w)^\top \Delta\boldsymbol{\phi}_{s,z}(x) \end{array}\right] \\ \\ \Delta L_{s,z}(v) \end{array}\right] \tag{23}$$

At an intuitive level, the quantity (23) can be viewed as the value of an integral over the decision boundary between output $z$ and output $s$. This integral can be expressed in terms of the two dimensional density for the random variables $w^\top \Phi(x, z)$ and $w^\top \Phi(x, s)$. There is a boundary layer along this decision boundary of width $\epsilon(\Delta w)^\top \Delta\boldsymbol{\phi}_{s,z}(x)$. Note that the width of the boundary layer is determined by the two random variables under consideration. The general position assumption implies that this integral can be calculated from a continuous density on $\mathbb{R}^2$ and the width of the boundary layer becomes a factor in the integral. We then have that (23) is asymptotically equivalent to the following.

$$\sum_{v \in \mathcal{Y},\, z,s \in \mathcal{Z}} \mathbb{E}\left[\begin{array}{l} I\left[\begin{array}{l} y = v \\ z_w(x) = z \\ (\Delta w)^\top \Delta\boldsymbol{\phi}_{s,z}(x) \geq 0 \\ w^\top \Delta\boldsymbol{\phi}_{z,s}(x) \in (0, \epsilon) \end{array}\right] \\ \\ (\Delta w)^\top \Delta\boldsymbol{\phi}_{s,z}(x) \;\; \Delta L_{s,z}(v) \end{array}\right] \tag{24}$$

Now define $E(v, z, s)$ to be the expectation inside the sum in (24) and consider the quantity $E(v, z, s) + E(v, s, z)$. Reversing $z$ and $s$ flips the sign of both $\Delta\boldsymbol{\phi}_{z,s}(x)$ and of $\Delta_{z,s}(x)$. So the only difference between the term $E(v, z, w)$ and $E(v, w, z)$ is that the first requires that $(\Delta w)^\top \Delta\boldsymbol{\phi}_{s,z}(x)$ is positive while the second term requires that this same quantity is negative. We then get than $E(v, z, s) + E(v, s, z)$ is the following quantity.

$$\mathbb{E}\left[\begin{array}{l} I\left[\begin{array}{l} y = v \\ z_w(x) = z \\ w^\top \Delta\boldsymbol{\phi}_{z,s}(x) \in (0, \epsilon) \end{array}\right] \\ \\ (\Delta w)^\top \Delta\boldsymbol{\phi}_{s,z}(x) \;\; \Delta L_{s,z}(v) \end{array}\right] \tag{25}$$

This gives that the directional gradient quantity (20) is asymptotically equivalent to the following.

$$\frac{1}{2} \sum_{v \in \mathcal{Y},\ z,s \in \mathcal{Z}} \mathbb{E}\left[\begin{array}{l} I\left[\begin{array}{l} y = v \\ z_w(x) = z \\ w^\top \Delta\boldsymbol{\phi}_{z,s}(x) \in (0, \epsilon) \end{array}\right] \\ \\ (\Delta w)^\top \Delta\boldsymbol{\phi}_{s,z}(x) \;\; \Delta L_{s,z}(v) \end{array}\right] \tag{26}$$

We now consider the difference of feature vector quantity (21). By similar arguments we get that (21) is asymptotically equivalent to the following sequence of quantities.

$$\sum_{v \in \mathcal{Y},\ z,s \in \mathcal{Z}} \mathbb{E}\left[\begin{array}{l} I\left[\begin{array}{rcl} y & = & v \\ z_w(x) & = & z \\ z_w(x, y, \epsilon) & = & s \end{array}\right] \\ \\ (\Delta w)^\top \Delta\boldsymbol{\phi}_{z,s}(v) \end{array}\right] \tag{27}$$

$$\sum_{v \in \mathcal{Y},\ z,s \in \mathcal{Z}} \mathbb{E}\left[\begin{array}{l} I\left[\begin{array}{l} y = v \\ z_w(x) = z \\ w^\top \Delta\boldsymbol{\phi}_{z,s}(x) \\ \in (0, \epsilon \Delta L_{z,s}(y)) \end{array}\right] \\ \\ (\Delta w)^\top \Delta\boldsymbol{\phi}_{z,s}(v) \end{array}\right] \tag{28}$$

$$\sum_{v \in \mathcal{Y},\ z,s \in \mathcal{Z}} \mathbb{E}\left[\begin{array}{l} I\left[\begin{array}{l} y = v \\ z_w(x) = z \\ \Delta L_{z,s}(y) \geq 0 \\ w^\top \Delta\boldsymbol{\phi}_{z,s}(x) \in (0, \epsilon) \end{array}\right] \\ \\ \Delta L_{z,s}(y) \;\; (\Delta w)^\top \Delta\boldsymbol{\phi}_{z,s}(v) \end{array}\right] \tag{29}$$

$$\frac{1}{2} \sum_{v \in \mathcal{Y},\ z,s \in \mathcal{Z}} \mathbb{E}\left[\begin{array}{l} I\left[\begin{array}{l} y = v \\ z_w(x) = z \\ w^\top \Delta\boldsymbol{\phi}_{z,s}(x) \in (0, \epsilon) \end{array}\right] \\ \\ \Delta L_{z,s}(y) \;\; (\Delta w)^\top \Delta\boldsymbol{\phi}_{z,s}(v) \end{array}\right] \tag{30}$$

The quantity (30) is equal to the quantity (26) and we have now shown that (20) is asymptotically equal to (21).