Direct Loss Minimization for training Deep Neural Nets

Yang Song
Dept. of Physics, Tsinghua University
songyang12@mails.tsinghua.edu.cn

Alexander G. Schwing & Richard S. Zemel & Raquel Urtasun
Dept. of Computer Science, University of Toronto
{aschwing,zemel,urtasun}@cs.toronto.edu

Abstract

Supervised training of deep neural nets typically relies on minimizing cross-entropy. However, in many domains, we are interested in performing well on specific application-specific metrics. In this paper we proposed a direct loss minimization approach to train deep neural networks, taking into account the application-specific loss functions. This can be non-trivial, when these functions are non-smooth and non-decomposable. We demonstrate the effectiveness of our approach in the context of maximizing average precision for ranking problems. Towards this goal, we propose a dynamic programming algorithm that can efficiently compute the weight updates. Our approach proves superior to a variety of baselines in the context of action classification and object detection.

1 Introduction

Supervised neural network training involves computing the gradient of the loss function with respect to the parameters of the model, and therefore requires the loss function to be differentiable. Many interesting loss functions are, however, not differentiable with respect to the output of the network. Notable examples are functions based on discrete outputs, as is common in labeling and ranking problems. In many cases these losses are also *non-decomposable*, in that they cannot be expressed as simple sums over the output units of the network.

In the context of structured prediction problems, in which the output is multi-dimensional, researchers have developed max-margin training methods that are capable of coping with such loss functions. Standard learning in this paradigm involves changing the model parameters such that the ground truth output has higher score than any other output. This is typically encoded by a constraint, enforcing that the ground truth score should be higher than a selected, contrastive output. The latter is defined as the result of performing inference in a modified score functionwhich combines the model score and the task loss. This encodes the fact that we penalize high scoring configurations that are less good in terms of the task loss, *i.e.*, the metric that we care about for the application domain. Various efficient methods have been proposed for incorporating discrete and complicated loss functions into this approach (Yue et al., 2007; Volkovs & Zemel, 2009; Tarlow & Zemel, 2012; Mohapatra et al., 2014).

However, this form of learning does not directly optimize the target function. An alternative approach, typically used in deep neural networks, is to train with a surrogate loss such as cross-entropy (LeCun & Huang, 2005; Bengio et al., 2015), that can be easily optimized. At test time we evaluate using the task loss. The problem of this procedure is that for many application domains the task loss differs significantly from the surrogate loss.

In this paper we propose to train deep neural nets to directly optimize the task loss, for structured, non-decomposable loss functions. In their seminal work, McAllester et al. (2010) showed that a form of structured prediction learning computes the gradients of the task loss. This work is, however, limited to linear models. In this paper we extend it to the non-linear case, proving that the theoretical results hold in this more general setting. The result is a simple learning algorithm that can be applied to nonlinear, deep networks. We demonstrate the effectiveness of our approach in the context of optimizing the average precision (AP) in ranking tasks. This application is challenging as AP is neither decomposable nor smooth. Our experiments investigate the effectiveness of this algorithm, and the benefits of optimizing the task loss, in the context of action classification and object detection.

2 Direct Loss Minimization for neural networks

In this section we present a formulation for learning neural networks by minimizing the task loss. Towards this goal, our first main result is a theorem extending direct loss minimization to deep non-linear networks.

A neural network can be viewed as defining a composite scoring function $F(x, y, w)$, which depends on the input data $x \in \mathcal{X}$, some parameters $w \in \mathbb{R}^A$, and the output $y \in \mathcal{Y}$. In the case of multi-class classification, the output refers to one of $|\mathcal{Y}|$ classes, *i.e.*, $y \in \{1, \dots, |\mathcal{Y}|\}$. Inference is then performed by picking the output with maximal score, *i.e.*:

$$y_w = \arg\max_{\hat{y} \in \mathcal{Y}} F(x, \hat{y}, w).$$

Given a dataset of input-output pairs $\mathcal{D} = \{(x, y)\}$, a standard approach to learning is to optimize the parameters w of the scoring function F by minimizing cross-entropy. This is equivalent to maximizing the likelihood of the data, where the probability over each output configuration is given by the output of a softmax function in the last layer of the network.

However, in many practical applications, we want prediction to succeed in an application-specific metric. This metric is typically referred to as the *task loss*, $\Delta(y, y_w)$. In this paper we are thus interested in minimizing the task loss

$$w^* = \arg\min_w \mathbb{E}\left[\Delta(y, y_w)\right], \tag{1}$$

where $\mathbb{E}\left[\cdot\right]$ denotes an expectation taken over the given dataset. Solving this program is non-trivial, as many loss functions of interest are non-decomposable and non-smooth, and thus are not amenable to gradient-based methods. Examples of such metrics include average precision (AP) from information retrieval, intersection-over-union which is used in image labeling, and normalized discounted cumulative gain (NDCG) which is popular in ranking. In general many metrics are discrete, are not simple sums over the network outputs, and are not readily differentiable.

It is hence common to employ a surrogate error metric where one can directly compute the gradients with respect to the parameters, such as cross-entropy or hinge-loss, during training of a classifier, while reporting results using the more appropriate measures. In the context of structured prediction models, several approaches have been developed to be able to optimize the structured hinge loss with non-decomposable task losses, *e.g.*, Tarlow & Zemel (2012); Yue et al. (2007); Mohapatra et al. (2014). While these methods include the task loss in the objective, they are still not minimizing it. As a consequence, these surrogate losses are at best highly correlated with the desired metric. Finding efficient techniques to directly minimize the metric of choice is therefore desirable.

McAllester et al. (2010) showed that it is possible to asymptotically optimize the task-loss when the function F is linear in the parameters *i.e.*, $F(x, y, w) = w^\top \phi(x, y)$. This work has produced encouraging results. For example, McAllester et al. (2010) used this technique for phoneme-to-speech alignment on the TIMIT dataset optimizing the *τ-alignment loss* and the *τ-insensitive loss*, while Keshet et al. (2011) illustrated applicability of the method to hidden Markov models for speech. Direct loss minimization was also shown to work well for inverse optimal control by Doerr et al. (2015).

The first contribution of our work is to generalize this theorem to arbitrary scoring functions, *i.e.*, non-convex functions. This allow us to derive a new training algorithm for deep neural nets which directly minimizes the task loss.

Theorem 1 (General Loss Gradient Theorem)**.** *When given a finite set $\mathcal{Y}$, a scoring function $F(x, y, w)$, a data distribution, as well as a task-loss $\Delta(y, \hat{y})$, then, under some mild regularity*

```
Algorithm: Direct Loss Minimization for Deep Networks
Repeat until stopping criteria
    1. Forward pass to compute $F(x, \hat{y}; w)$
    2. Obtain $y_w$ and $y_{\text{direct}}$ via inference and loss-augmented inference
    3. Single backward pass via chain rule to obtain gradient $\nabla_w \mathbb{E}\left[\Delta(y, y_w)\right]$
    4. Update parameters using stepsize $\eta$: $w \leftarrow w - \eta \nabla_w \mathbb{E}\left[\Delta(y, y_w)\right]$
```

Figure 1: Our algorithm for direct loss minimization.

conditions (see the Appendix for details), the direct loss gradient has the following form:

$$\nabla_w \mathbb{E}\left[\Delta(y, y_w)\right] = \lim_{\epsilon \to 0} \frac{s}{\epsilon} \mathbb{E}\left[\nabla_w F(x, y_{\text{direct}}, w) - \nabla_w F(x, y_w, w)\right], \tag{2}$$

with

$$\begin{aligned} y_w &= \arg\max_{\hat{y} \in \mathcal{Y}} F(x, \hat{y}, w), \\ y_{\text{direct}} &= \arg\max_{\hat{y} \in \mathcal{Y}} F(x, \hat{y}, w) + s\epsilon\Delta(y, \hat{y}). \end{aligned} \tag{3}$$

where $s \in \{+1, -1\}$.

Proof. We refer the reader to the Appendix Sec. 5.1, for a proof of the theorem. □

According to Theorem 1, to obtain the gradient we need to find the solution of two inference problems. The first computes y_w, which is a standard inference task, solved by employing the forward propagation algorithm. The second inference problem is prediction using a scoring function which is perturbed by the task loss $\Delta(y, \hat{y})$. This is typically non-trivial to solve, particularly when the task loss is not decomposable. We borrow terminology from the structured prediction literature, where this perturbed inference problem is commonly referred to as *loss-augmented inference* (Tsochantaridis et al., 2005; Chen et al., 2015). In the following section we derive an efficient dynamic programming algorithm to perform loss-augmented inference when the task loss is average precision.

Note that loss-augmented inference, as specified in Eq. (3), can take the task loss into account in a positive or a negative way. Depending on the sign of s, the gradient direction changes. McAllester et al. (2010) provide a nice intuition for the two different directions. The positive update performs a step away from a worse configuration, while the negative direct loss gradient encourages moves towards better outputs. This can be seen when considering that maximization of the loss returns the worst output configuration, while maximization of its negation returns the label with the lowest loss. We explore both of these alternatives below. Further note the relationship between direct loss minimization and optimization of the hinge-loss. While we compute the gradient via a difference between the loss-augmented inference result and the prediction in the former case, the latter requires computation of the difference between the loss-augmented inference solution and ground truth.

In Fig. 1 we summarize the resulting learning algorithm, which consists of the following four steps. First we use a standard forward pass to evaluate F. We then perform inference and loss-augmented inference as specified in Eq. (3) to obtain the prediction y_w and y_{direct}. We combine the predictions to obtain the gradient $\nabla_w \mathbb{E}\left[\Delta(y, y_w)\right]$ via a single backward pass which is then used to update the parameters. For notational simplicity we omit details like momentum-based gradient updates, the use of mini-batches, *etc*.

3 Direct Loss Minimization for Average Precision

In order to directly optimize the task-loss we are required to compute the gradient defined in Eq. (2). As mentioned above, we need to solve both the standard inference task as well as the loss-augmented inference problem given in Eq. (3). While the former is typically easy, the latter depends on Δ and might be very complex to solve, *e.g.*, when the loss is not decomposable.

In this paper we consider ranking problems, where the desired task loss Δ is average precision (AP). For the linear setting, efficient algorithms for positive loss-augmented inference with AP loss were proposed by Yue et al. (2007) and Mohapatra et al. (2014). Their results can be extended to the non-linear setting only in the positive case, where $y_{\text{direct}} = \arg\max_{\hat{y}\in\mathcal{Y}} F(x,\hat{y},w) + \epsilon\Delta(y,\hat{y})$. For the negative setting inequalities required in their proof do not hold and thus their method is not applicable in this case. In this section we propose a more general algorithm that can handle both cases.

To compute the AP loss we are given a set of positive (*i.e.*, relevant) and negative samples. The sample x_i belongs to the positive class if $i \in \mathcal{P} = \{1,\dots,|\mathcal{P}|\}$, and x_i is part of the negative class if $i \in \mathcal{N} = \{|\mathcal{P}|+1,\dots,|\mathcal{P}|+|\mathcal{N}|\}$.

We define the output to be composed of pairwise comparisons, with $y_{i,j} = 1$ if sample i is ranked higher than j, $y_{i,i} = 0$, and $y_{i,j} = -1$ otherwise. We subsumed all these pairwise comparisons in $y = (\cdots, y_{i,j}, \cdots)$. Similarly $x = (x_1, \cdots, x_N)$ contains all inputs, with $N = |\mathcal{P}| + |\mathcal{N}|$ the total number of data points in the training set. In addition, we assume the ranking across all samples y to be complete, *i.e.*, consistent. During inference we obtain a ranking by predicting scores $\phi(x_i, w)$ for all data samples x_i which are easily sorted afterwards.

For learning we generalize the feature function defined by Yue et al. (2007) and Mohapatra et al. (2014) to a non-linear scoring function, using

$$F(x,y,w) = \frac{1}{|\mathcal{P}||\mathcal{N}|}\sum_{i\in\mathcal{P},j\in\mathcal{N}} y_{i,j}\left(\phi(x_i,w) - \phi(x_j,w)\right).$$

where $\phi(x_i, w)$ is the output of the deep neural net when using the i-th example as input.

AP is unfortunately a non-decomposable loss function, i.e., it does not decompose into functions dependent only on the individual $y_{i,j}$. To define the non-decomposable AP loss formally, we construct a vector $\hat{p} = \text{rank}(\hat{y}) \in \{0,1\}^{|\mathcal{P}|+|\mathcal{N}|}$ by sorting the data points according to the ranking defined by the configuration $\hat{y}$. This vector contains a 1 for each positive sample and a value 0 for each negative element. Using the rank operator we obtain the AP loss by comparing two vectors $p = \text{rank}(y)$ and $\hat{p} = \text{rank}(\hat{y})$ via

$$\Delta_{\text{AP}}(p,\hat{p}) = 1 - \frac{1}{|\mathcal{P}|}\sum_{j:\hat{p}_j=1} \text{Prec@j}, \tag{4}$$

where Prec@j is the percentage of relevant samples in the prediction $\hat{p}$ that are ranked above position j.

To solve the loss-augmented inference task we have to solve the following program

$$\arg\max_{\hat{y}} F(x,\hat{y},w) \pm \epsilon\Delta_{\text{AP}}(\text{rank}(y),\text{rank}(\hat{y})), \tag{5}$$

In the following we derive a dynamic programming algorithm that can handle both the positive and negative case and has the same complexity as Yue et al. (2007). Towards this goal, we first note that Observation 1 of Yue et al. (2007) holds for both the positive and the negative case. For completeness, we repeat their observation here and adapt it to our notation.

Observation 1 (Yue et al. (2007))**.** *Consider rankings which are constrained by fixing the relevance at each position in the ranking (*e.g.*, the 3rd sample in the ranking must be relevant). Every ranking satisfying the same set of constraints will have the same Δ_{AP}. If the positive samples are sorted by their scores in descending order, and the irrelevant samples are likewise sorted by their scores, then the interleaving of the two sorted lists satisfying the constraints will maximize Eq.* (5) *for that constrained set of rankings.*

Observation 1 means that we only need to consider the interleaving of two sorted lists of $\mathcal{P}$ and $\mathcal{N}$ to solve Eq. (5). From now on we therefore assume that the elements of $\mathcal{P}$ and $\mathcal{N}$ are sorted in descending order of their predicted score.

We next assert the optimal substructure property of our problem. Let the restriction to subsets of i positive and j negative examples be given by $\mathcal{P}_i = \{1,\dots,i\}$ and $\mathcal{N}_j = \{|\mathcal{P}|+1,\dots,|\mathcal{P}|+j\}$.

```
Algorithm: AP loss-augmented inference
    1. Set $h(1,0) = \mp\epsilon\frac{1}{|\mathcal{P}|}$ and $h(0,1) = 0$
    2. For $i = 1, \ldots, |\mathcal{P}|, j = 1, \ldots, |\mathcal{N}|$, recursively fill the matrix
           $h(i,j) = \max\begin{cases} h(i-1,j) \mp \epsilon\frac{1}{|\mathcal{P}|}\frac{i}{i+j} + B(i,j), & i \geq 1, j \geq 0 \\ h(i,j-1) + G(i,j), & i \geq 0, j \geq 1 \end{cases}$
    3. Backtrack to obtain configuration $\hat{y}^*$
```

Figure 2: Our algorithm for AP loss-augmented maximization or minimization.

The cost function value obtained when restricting loss-augmented inference to the subsets can be computed as:

$$h(i,j) = \max_{\hat{y}} \frac{1}{|\mathcal{P}||\mathcal{N}|} \sum_{m\in\mathcal{P}_i} \sum_{n\in\mathcal{N}_j} \hat{y}_{m,n}(\phi(x_m,w) - \phi(x_n,w)) \pm \epsilon\Delta_{AP}^{i,j}(\mathrm{rank}(y), \mathrm{rank}(\hat{y})), \quad (6)$$

where $\Delta_{AP}^{i,j}$ refers to the AP loss restricted to subsets of i positive and j negative elements.

Lemma 1. *Suppose that* $\mathrm{rank}(\hat{y}^*)$ *is the optimal ranking for Eq.* (6) *when restricted to* i *positive and* j *negative samples. Any of its sub-sequences starting at position 1 is then also an optimal ranking for the corresponding restricted sub-problem.*

We provide the proof of this lemma in the Appendix, Sec. 5.2. Based on Lemma 1 we can construct Bellman equations to recursively fill in a matrix of size $\mathcal{P} \times \mathcal{N}$, resulting in an overall time complexity of $O(|\mathcal{P}||\mathcal{N}|)$. We can then obtain the optimal loss-augmented predicted ranking via back-tracking.

The Bellman recursion computes the optimal cost function value of the sub-sequence containing data from $\mathcal{P}_i$ and $\mathcal{N}_j$, based on previously computed values as follows:

$$h(i,j) = \max\begin{cases} h(i-1,j) \mp \epsilon\frac{1}{|\mathcal{P}|}\frac{i}{i+j} + B(i,j), & i \geq 1, j \geq 0 \\ h(i,j-1) + G(i,j), & i \geq 0, j \geq 1 \end{cases},$$

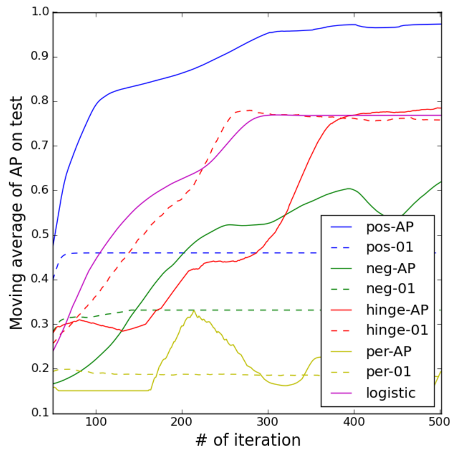
with initial conditions $h(1,0) = \mp\epsilon\frac{1}{|\mathcal{P}|}$ and $h(0,1) = 0$. Note that we used the pre-computed matrices of scores

$$B(i,j) = -\frac{1}{|\mathcal{P}||\mathcal{N}|} \sum_{k\in\mathcal{N}_j} (\phi(x_i,w) - \phi(x_k,w))$$
$$G(i,j) = \frac{1}{|\mathcal{P}||\mathcal{N}|} \sum_{k\in\mathcal{P}_i} (\phi(x_k,w) - \phi(x_j,w)).$$

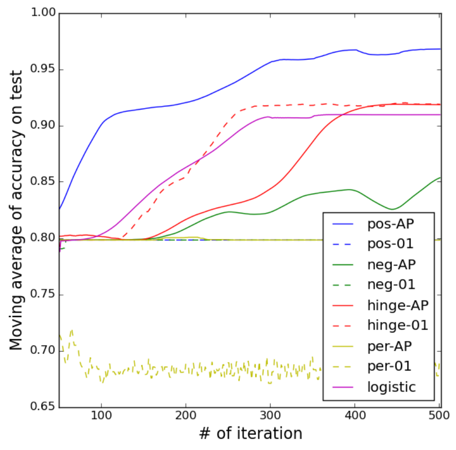
Intuitively, the Bellman recursion considers two cases: (i) how does the maximum score $h(i,j)$ for the solution restricted to sequences $\mathcal{P}_i$ and $\mathcal{N}_j$ change if we add a new positive sample; (ii) how does the maximum score $h(i,j)$ change when adding a new negative sample. In both cases we need to add the respective scores $B(i,j)$ or $G(i,j)$. When adding a positive sample we additionally need to consider a contribution from the precision which contains i positive elements from a total of $i+j$ elements.

The matrices $B(i,j)$ and $G(i,j)$ store the additional contribution to Eq. (6) obtained when adding a positive or a negative sample respectively. They can be efficiently computed ahead of time using the recursion

$$B(i,j) = B(i,j-1) - \frac{1}{|\mathcal{P}||\mathcal{N}|}(\phi(x_i,w) - \phi(x_j,w))$$
$$G(i,j) = G(i-1,j) + \frac{1}{|\mathcal{P}||\mathcal{N}|}(\phi(x_i,w) - \phi(x_j,w)),$$



(a)



(b)

Figure 3: Synthetic experiment: we illustrate average precision over the number of iterations on the test set in (a) and accuracy over the number of iterations on the test set in (b).

with initial conditions $B(i, 0) = 0$, $G(0, j) = 0$.

We summarize our dynamic programming algorithm for AP loss-augmented inference in Fig. 2. Note that after having completed the matrix $h(i, j)$ we can backtrack to obtain the best loss-augmented ranking. We hence presented a general solution to solve positive and negative loss-augmented inference defined in Eq. (3) for the AP loss given in Eq. (4).

4 Experimental Evaluation

To evaluate the performance of our approach we perform experiments on both synthetic and real datasets. We compare the positive and negative version of our direct loss minimization approach to a diverse set of baselines.

4.1 Synthetic data

Dataset: We generate synthetic data by randomly drawing the parameters of a 4-layer neural network with activations being rectified linear units. In particular, we use a normal distribution with zero mean and unit variance. We generate 20,000 input data points x_i from a 10 dimensional standard Gaussian. We compute the score of each sample x_i and sort them in descending order. The top 20% of the samples are assigned to the positive set $\mathcal{P}$. We then randomly divide the generated data into a training set containing 10,000 elements and a test set containing the rest. To ensure that we don't suffer from model misspecification we employ the same network structure when training the parameters from random initializations using different algorithms.

Algorithms: We evaluate the positive and negative versions of our direct loss minimization when using two different task losses: AP and 0-1 loss. For the latter we predict for each sample x_i whether it is a member of the set $\mathcal{P}$ or whether it is part of the set $\mathcal{N}$. We named these algorithms, "pos-AP," "neg-AP," "pos-01," and "neg-01." We also evaluate training the network using hinge loss, when employing AP and 0-1 loss as the task losses. We called these baselines "hinge-AP" and "hinge-01." We use the perceptron updates as additional baselines, which we call "per-AP" and "per-01." Finally, the last baseline uses cross-entropy to train the network. We call this approach "logistic". The parameters of all algorithms are individually determined via grid search to produce the best AP on the training set.

Results: We report both AP and accuracy on the test set in Fig. 3. We observe that the perceptron update does not perform well, since it does not take the task loss into account. Contrary to the claim

	jumping	phoning	playing instrument	reading	riding bike	riding horse	running	taking photo	using computer	walking	**average**
Logistic	71.3	36.2	75.7	37.5	**90.5**	89.7	73.5	44.8	63.8	65.1	64.8
Hinge	69.8	36.5	74.2	38.8	90.4	88.9	74.2	41.6	61.4	60.6	63.6
Ours	**71.9**	**42.0**	**76.0**	**40.0**	90.0	**90.4**	**77.6**	**46.5**	**66.6**	**66.4**	**66.7**

Table 1: Comparison of our direct AP loss minimization approach to surrogate loss function optimization on the action classification task.

in (McAllester et al., 2010), the negative augmented inference is not competitive in our setting. This might be due to the fact that it tends to overly correct the classifier in noisy situations. We expect the negative update to lead to better performance in less noisy situations, *e.g.*, if the data is nearly linearly separable. To verify this hypothesis we provide additional results in the appendix, Sec. 5.3.

It is important to note that when employing positive updates we outperformed all baselines by a large margin. Encouraged by the gain in performance we focus on positive non-linear direct loss minimization during the experimental evaluation on real datasets.

4.2 Action classification task

Dataset: In the next experiment we use the PASCAL VOC2012 action classification dataset provided by Everingham et al. (2014). The dataset contains 4588 images and 6278 "trainval" person bounding boxes. For each of the 10 target classes, we divide the trainval dataset into equally sized training, validation and test sets. We tuned the learning rate, regularization weight, and ϵ for all the algorithms based on their performance on our validation dataset, and report the results on the test set. For all algorithms we used the entire available training set in a single batch and performed 300 iterations.

Algorithms: We train our nonlinear direct loss minimization as well as all the baselines individually for each class. As baselines we use a deep network trained with cross entropy and hinge loss as baselines. The deep network used in those experiments follows the architecture of Krizhevsky et al. (2012), with the top dimension adjusted to 1 and the parameters initialized using weights trained on ILSVRC2012 (Russakovsky et al., 2015). Inspired by RCNN (Girshick et al., 2014), we cropped the regions of each image with a padding of 16 pixels and interpolated them to a size of $227 \times 227 \times 3$ to fit the input data dimension of the network. All the algorithms we compare to as well as our approach use raw pixels as input.

Results: We provide quantitative results in Tab. 1. Our direct AP loss minimization clearly outperforms the baselines by 1.9% and 3.1% respectively. This is quite significant for this task.

4.3 Object detection task

Dataset: We use the PASCAL VOC2012 object detection dataset collected by Everingham et al. (2014). The dataset contains 5717 images for training, 5823 images for validation and 10991 images for test. Moreover, for each image, we use the fast mode of selective search by Uijlings et al. (2013) to produce around 2000 bounding boxes. We train algorithms on the training set and report results on the validation set.

Algorithms: On this dataset we follow the RCNN paradigm Girshick et al. (2014). For direct loss minimization, we adjust the dimension of the top layer of the network Krizhevsky et al. (2012) to be 1 and fine-tune using weights pre-trained on ILSVRC2012 (Russakovsky et al., 2015). We train direct loss minimization for all 20 classes separately. Different from the action classification task, we cannot calculate the overall AP in each iteration, because of the large number of bounding boxes.

	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	sofa	train	tvmonitor	**average**
Softmax	**66.4**	57.8	42.4	26.1	25.6	62.3	52.9	61.0	23.9	**44.6**	34.7	56.4	51.0	**65.7**	51.2	**29.8**	**50.5**	29.6	48.7	60.0	47.0
Logistic	63.8	**61.0**	42.6	30.7	23.5	63.2	51.7	58.5	20.1	37.0	32.0	52.8	50.8	62.5	50.1	23.5	48.3	33.1	48.5	57.4	45.6
Ours	65.1	59.8	**43.7**	**31.4**	**27.7**	**64.6**	**53.1**	**63.7**	**25.6**	40.2	**36.2**	**58.1**	**52.8**	63.6	**56.2**	28.1	50.0	**38.9**	**50.0**	**61.3**	**48.5**

Table 2: Comparison of our direct AP loss minimization approach to surrogate loss function optimization on the object detection task.

Instead, we use the AP on each mini-batch to approximate the overall AP. We find that using a batch size of 512 balances computational complexity and performance, though using a larger batch size (such as 2048) will generally result in better performance. For our final results, we use a learning rate of 0.1, a regularization parameter of 0.0000001, and $\epsilon = 0.1$ for all classes.

As baselines, we use the fine-tuned softmax network of Girshick et al. (2014). Since this network was jointly trained for all classes, we also evaluate a network which uses cross-entropy and is trained separately for each class. We can thus evaluate whether a separate training of each class results in significant benefits. Again, the network structure was chosen to be identical and we use the parameters provided by Russakovsky et al. (2015) for initialization.

Results: Tab. 2 shows that the logistic network trained independently for each class yields worse performance than the network trained jointly with softmax. Likely this is due to the fact that shared information benefits the softmax network. Importantly, we are also able to show competitive results of stochastic structured direct loss minimization, outperforming the strongest baseline by 1.5%.

5 Conclusion

In this paper we have proposed a direct loss minimization approach to train deep neural networks. We have demonstrated the effectiveness of our approach in the context of maximizing average precision for ranking problems. This involves minimizing a non-smooth and non-decomposable loss. Towards this goal we have proposed a dynamic programing algorithm that can efficiently compute the weight updates. Our experiments showed that this is beneficial when compared to a large variety of baselines in the context of action classification and object detection. In the future, we plan to investigate direct loss minimization in the context of other non-decomposable loses such as intersection over union.

Acknowledgments

We thank David McAllester for discussions. This work was partially supported by ONR Grant N00014-14-1-0232, and a Google researcher award.

References

Bengio, Y., Goodfellow, I. J., and Courville, A. Deep learning. Book in preparation for MIT Press, 2015. URL http://www.iro.umontreal.ca/~bengioy/dlbook.

Chen, L.-C., Schwing, A. G., Yuille, A. L., and Urtasun, R. Learning Deep Structured Models. In *Proc. ICML*, 2015.

Cheng, C.-C., Sha, F., and Saul, L. K. Matrix updates for perceptron training of continuous density hidden markov models. In *Proc. ICML*, 2009.

Doerr, A., Ratliff, N., Bohg, J., Toussaint, M., and Schaal, S. Direct loss minimization inverse optimal control. *Proc. of robotics: science and systems (R: SS)*, 2015.

Everingham, M., Eslami, A. S. M., van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The pascal visual object classes challenge: A retrospective. *IJCV*, 2014.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. CVPR*, 2014.

Keshet, J., Cheng, C.-C., Stoehr, M., and McAllester, D. Direct Error Rate Minimization of Hidden Markov Models. In *Proc. Interspeech*, 2011.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*, 2012.

LeCun, Y. and Huang, F. J. Loss Functions for Discriminative Training of Energy-Based Models. In *Proc. AISTATS*, 2005.

McAllester, D. A., Keshet, J., and Hazan, T. Direct loss minimization for structured prediction. In *Proc. NIPS*, 2010.

Mohapatra, P., Jawahar, C. V., and Kumar, M. P. Efficient Optimization for Average Precision SVM. In *Proc. NIPS*, 2014.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

Tarlow, D. and Zemel, R. S. Structured Output Learning with High Order Loss Functions. In *Proc. AISTATS*, 2012.

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. Large Margin Methods for Structured and Interdependent Output Variables. *JMLR*, 2005.

Uijlings, J., van de Sande, K., Gevers, T., and Smeulders, A. Selective search for object recognition. *IJCV*, 2013.

Volkovs, M. N. and Zemel, R. S. BoltzRank: Learning to Maximize Expected Ranking Gain. In *Proc. ICML*, 2009.

Yue, Y., Finley, T., Radlinski, F., and Joachims, T. A support vector method for optimizing average precision. In *Proc. SIGIR*, 2007.

APPENDIX

5.1 PROOF OF THE GENERAL LOSS GRADIENT THEOREM

In order to lay the foundation for the proof of the general loss gradient theorem, we first show the following lemma. In short, it provides the bases for exchanging integral bounds when ϵ approaches 0 from above.

Lemma 2.

$$\lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \int_0^{a\epsilon + o(\epsilon)} \int_{b\epsilon + o(\epsilon)}^{\infty} f(x, y, \epsilon) dx dy = \lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \int_0^{a\epsilon + o(\epsilon)} \int_0^{\infty} f(x, y, \epsilon) dx dy$$

Proof. We have

$$\begin{aligned} &\lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \int_0^{a\epsilon + o(\epsilon)} \int_{b\epsilon + o(\epsilon)}^{\infty} f(x, y, \epsilon) dx dy \\ &= \lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \int_0^{a\epsilon + o(\epsilon)} \int_0^{\infty} f(x, y, \epsilon) dx dy - \lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \int_0^{a\epsilon + o(\epsilon)} \int_0^{b\epsilon + o(\epsilon)} f(x, y, \epsilon) dx dy. \end{aligned}$$

Suppose f is continuous w.r.t x, y, ϵ, then as $\epsilon \to 0^+$, it can be bounded by some constant M. As a result, we have

$$\frac{1}{\epsilon} \int_0^{a\epsilon + o(\epsilon)} \int_0^{b\epsilon + o(\epsilon)} |f(x, y, \epsilon)| dx dy \leq M(ab\epsilon + ao(\epsilon) + bo(\epsilon) + o(\epsilon)o(\epsilon)/\epsilon), \tag{7}$$

which means

$$\lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \int_0^{a\epsilon + o(\epsilon)} \int_0^{b\epsilon + o(\epsilon)} f(x, y, \epsilon) dx dy = 0. \tag{8}$$

□

Repeated application of Lemma 2 as demonstrated in the following is directly helpful for the proof of the general loss gradient theorem, which is why we state it explicitly:

Lemma 3. *Let $a > 0$, then we assert*

$$\begin{aligned} &\lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \int_0^{a\epsilon + o(\epsilon)} \int_{b_1\epsilon + o(\epsilon)}^{\infty} \cdots \int_{b_n\epsilon + o(\epsilon)}^{\infty} f(x, y_1, \cdots, y_n) dx dy_1 \cdots dy_n \\ &= \lim_{\epsilon \to 0^+} \frac{1}{\epsilon} a \int_0^{\epsilon} \int_0^{\infty} \cdots \int_0^{\infty} f(x, y_1, \cdots, y_n) dx dy_1 \cdots dy_n \end{aligned}$$

Proof. Let $f(x, y_1, \epsilon) = \int_{b_2\epsilon + o(\epsilon)}^{\infty} \cdots \int_{b_n\epsilon + o(\epsilon)}^{\infty} f(x, y_1, \cdots, y_n) dy_2 \cdots dy_n$. Due to Lemma 2 we obtain the following:

$$\begin{aligned} &\lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \int_0^{a\epsilon + o(\epsilon)} \int_{b_1\epsilon + o(\epsilon)}^{\infty} f(x, y_1, \epsilon) dx dy_1 = \lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \int_0^{a\epsilon + o(\epsilon)} \int_0^{\infty} f(x, y_1, \epsilon) dx dy_1 \\ &= \lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \int_0^{a\epsilon + o(\epsilon)} \int_0^{\infty} \int_{b_2\epsilon + o(\epsilon)}^{\infty} \cdots \int_{b_n\epsilon + o(\epsilon)}^{\infty} f(x, y_1, \cdots, y_n) dx dy_1 \cdots dy_n. \end{aligned}$$

Now we denote $f(x, y_2, \epsilon) = \int_0^\infty \int_{b_3\epsilon+o(\epsilon)}^\infty \cdots \int_{b_n\epsilon+o(\epsilon)}^\infty f(x, y_1, \cdots, y_n) dy_1 dy_3 \cdots dy_n$ and follow a similar procedure:

$$
\begin{aligned}
&\lim_{\epsilon\to 0^+} \frac{1}{\epsilon} \int_0^{a\epsilon+o(\epsilon)} \int_0^\infty \int_{b_2\epsilon+o(\epsilon)}^\infty \cdots \int_{b_n\epsilon+o(\epsilon)}^\infty f(x, y_1, \cdots, y_n) dx dy_1 \cdots dy_n \\
&= \lim_{\epsilon\to 0^+} \frac{1}{\epsilon} \int_0^{a\epsilon+o(\epsilon)} \int_{b_2\epsilon+o(\epsilon)}^\infty f(x, y_2, \epsilon) dx dy_2 \\
&= \lim_{\epsilon\to 0^+} \frac{1}{\epsilon} \int_0^{a\epsilon+o(\epsilon)} \int_0^\infty f(x, y_2, \epsilon) dx dy_2 \\
&= \cdots \text{(following this procedure recursively)} \\
&= \lim_{\epsilon\to 0^+} \frac{1}{\epsilon} \int_0^{a\epsilon+o(\epsilon)} \int_0^\infty \cdots \int_0^\infty f(x, y_1, \cdots, y_n) dx dy_1 \cdots dy_n \\
&= \lim_{\epsilon\to 0^+} \frac{a\epsilon + o(\epsilon)}{\epsilon} \lim_{\epsilon\to 0^+} \frac{1}{a\epsilon + o(\epsilon)} \int_0^{a\epsilon+o(\epsilon)} \int_0^\infty \cdots \int_0^\infty f(x, y_1, \cdots, y_n) dx dy_1 \cdots dy_n \\
&= a \lim_{\epsilon\to 0^+} \frac{1}{a\epsilon + o(\epsilon)} \int_0^{a\epsilon+o(\epsilon)} \int_0^\infty \cdots \int_0^\infty f(x, y_1, \cdots, y_n) dx dy_1 \cdots dy_n \\
&= a \lim_{a\epsilon+o(\epsilon)\to 0^+} \frac{1}{a\epsilon + o(\epsilon)} \int_0^{a\epsilon+o(\epsilon)} \int_0^\infty \cdots \int_0^\infty f(x, y_1, \cdots, y_n) dx dy_1 \cdots dy_n \\
&= \lim_{\epsilon\to 0^+} \frac{1}{\epsilon} a \int_0^{\epsilon} \int_0^\infty \cdots \int_0^\infty f(x, y_1, \cdots, y_n) dx dy_1 \cdots dy_n.
\end{aligned}
$$

This completes the proof. □

For readability we repeat the main theorem:

Theorem 1 (General Loss Gradient Theorem). *When given a finite set $\mathcal{Y}$, a scoring function $F(x, y, w)$, a data distribution, as well as a task-loss $\Delta(y, \hat{y})$, then, under some mild regularity conditions (see the proof for details), the direct loss gradient has the following form:*

$$\nabla_w \mathbb{E}\left[\Delta(y, y_w)\right] = \lim_{\epsilon\to 0} \frac{\pm 1}{\epsilon} \mathbb{E}\left[\nabla_w F(x, y_{\text{direct}}, w) - \nabla_w F(x, y_w, w)\right],$$

with

$$
\begin{aligned}
y_w &= \arg\max_{\hat{y}\in\mathcal{Y}} F(x, \hat{y}, w), \\
y_{\text{direct}} &= \arg\max_{\hat{y}\in\mathcal{Y}} F(x, \hat{y}, w) \pm \epsilon\Delta(y, \hat{y}).
\end{aligned}
$$

In the following we prove the positive case and note that the negative case is easily proved using a similar procedure.

Proof. Without loss of generality, in this proof we assume $\mathcal{Y} = \{1, 2, \ldots, |\mathcal{Y}|\}$ and no tie in maximization.

By definition of the directional derivative we have

$$\Delta w^\intercal \nabla_w \mathbb{E}\left[L(y, y_w(x)\right] = \lim_{\epsilon\to 0} \frac{\mathbb{E}\left[L(y, y_{w+\epsilon\Delta w}(x))\right] - \mathbb{E}\left[L(y, y_w(x))\right]}{\epsilon}.$$

Hence we need to prove the following equivalence:

$$
\begin{aligned}
&\lim_{\epsilon\to 0} \frac{\mathbb{E}\left[L(y, y_{w+\epsilon\Delta w}(x)) - L(y, y_w(x))\right]}{\epsilon} &(9)\\
&= \lim_{\epsilon\to 0} \frac{\Delta w^\intercal \mathbb{E}\left[\nabla_w F(x, y_{\text{direct}}, w) - \nabla_w F(x, y_w(x), w)\right]}{\epsilon}. &(10)
\end{aligned}
$$

Denote $\Delta F_w^{i,j}(x) = F(x,i,w) - F(x,j,w)$, $\Delta L(y)^{i,j} = L(y,i) - L(y,j)$. Note that $\Delta F_w^{i,i}(x) = 0$ and $\Delta L(y)^{i,i} = 0$. Therefore we just have to consider terms where the classification result changes when moving from w to $w + \epsilon \Delta w$. To this end we decompose the expectation in Eq. (9) into pairwise terms to yield

$$\mathbb{E}\left[L(y, y_{w+\epsilon\Delta w}(x) - L(y, y_w(x)))\right] = \sum_{i \neq j} \mathbb{E}\left[\Delta L(y)^{i,j} \mathbf{1}_{\{x \in \mathcal{A}^{i,j}\}}\right],$$

where the indicator set for a change from class label i to category j when moving from w to $w + \Delta w$ is given by

$$\begin{aligned}
\mathcal{A}^{i,j} &= \{x : y_{w+\Delta w}(x) = i, y_w(x) = j\} \\
&= \{x : \Delta F_{w+\epsilon\Delta w}^{i,k} > 0, \Delta F_w^{j,k} > 0, \quad \forall k \in \mathcal{Y}\} \\
&= \{x : \Delta F_w^{i,k} + \epsilon \Delta w^\intercal \nabla \Delta F_w^{i,k} + o(\epsilon) > 0, \Delta F_w^{j,k} > 0, \quad \forall k \in \mathcal{Y}\} \\
&= \{x : 0 < \Delta F_w^{j,i} < -\epsilon \Delta w^\intercal \nabla \Delta F_w^{j,i} + o(\epsilon) \\
&\qquad \Delta F_w^{j,k} > 0, \quad k \neq i \\
&\qquad \Delta F_w^{i,k} > -\epsilon \Delta w^\intercal \nabla \Delta F_w^{i,k} + o(\epsilon), \quad k \neq j\}.
\end{aligned}$$

Integrating the loss difference over the set, we obtain

$$\begin{aligned}
&\mathbb{E}\left[\Delta L(y)^{i,j} \mathbf{1}_{\{x \in \mathcal{A}^{i,j}\}}\right] \\
&= \mathbb{E}_\mu\Bigg[\Delta L(y)^{i,j} \int_0^{\epsilon \Delta w^\intercal \nabla \Delta F_w^{i,j} + o(\epsilon)} d\Delta F_w^{j,i} \int_0^\infty d\Delta F_w^{j,1} \cdots \int_0^\infty d\Delta F_w^{j,n} \\
&\int_{-\epsilon \Delta w^\intercal \nabla \Delta F_w^{i,1} + o(\epsilon)}^\infty d\Delta F_w^{i,1} \cdots \\
&\int_{-\epsilon \Delta w^\intercal \nabla \Delta F_w^{i,n} + o(\epsilon)}^\infty f(\Delta F_w^{j,1} \cdots \Delta F_w^{i,n} \mid \Delta L(y)^{i,j}, \Delta w^\intercal \nabla \Delta F_w^{i,j}) d\Delta F_w^{i,n}\Bigg].
\end{aligned}$$

As in (McAllester et al., 2010), we assume in the last equality above that any joint measure ρ on $\Delta F_w^{j,1} \cdots \Delta F_w^{i,n}, \Delta L(y)^{i,j}, \Delta w^\intercal \nabla \Delta F_w^{i,j}$ can be expressed as a measure μ on $\Delta L(y)^{i,j}, \Delta w^\intercal \nabla \Delta F_w^{i,j}$ and a bounded continuous conditional density function f. Based on Lemma 3, we conclude that Eq. (9) is equivalent to

$$\begin{aligned}
&\lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \mathbb{E}\left[\Delta L(y)^{i,j} \mathbf{1}_{\{x \in \mathcal{A}^{i,j}\}}\right] \\
&= \mathbb{E}_\mu\Bigg[\Delta L(y)^{i,j} (\Delta w^\intercal \nabla \Delta F_w^{i,j})^+ \lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \int_0^\epsilon d\Delta F_w^{j,i} \int_0^\infty d\Delta F_w^{j,1} \cdots \int_0^\infty d\Delta F_w^{j,n} \\
&\int_0^\infty d\Delta F_w^{i,1} \cdots \int_0^\infty f(\Delta F_w^{j,1} \cdots \Delta F_w^{i,n} | \Delta L(y)^{i,j}, \Delta w^\intercal \nabla \Delta F_w^{i,j}) d\Delta F_w^{i,n}\Bigg].
\end{aligned} \tag{11}$$

Following a similar procedure, we decompose the expectation in Eq. (10) to

$$\mathbb{E}\left[\Delta w^\intercal \nabla F(x, y_{\text{direct}}, w) - \Delta w^\intercal \nabla F(x, y_w(x), w)\right] = \sum_{i \neq j} \mathbb{E}\left[\Delta w^\intercal \Delta \nabla F(x)^{i,j} \mathbf{1}_{\{(x,y) \in \mathcal{B}^{i,j}\}}\right],$$

where the indicator set for a change from label i to a configuration j when changing from w to loss augmented inference is given by

$$\begin{aligned}
\mathcal{B}^{i,j} &= \{(x,y) : y_{\text{direct}} = i, y_w(x) = j\} \\
&= \{(x,y) : \Delta F_w^{i,k} + \epsilon \Delta L^{i,k} > 0, \Delta F_w^{j,k} > 0, \quad k \in \mathcal{Y}\} \\
&= \{(x,y) : 0 < \Delta F_w^{j,i} < -\epsilon \Delta L(y)^{j,i} \\
&\qquad \Delta F_w^{j,k} > 0, \quad k \neq i \\
&\qquad \Delta F_w^{i,k} > -\epsilon \Delta L(y)^{i,k}, \quad k \neq j\}.
\end{aligned}$$

Integrating the directional derivative over the set of configuration changes, we obtain

$$\begin{aligned}
&\mathbb{E}\left[\Delta w^\intercal \Delta\nabla F(x)^{i,j}\mathbf{1}_{\{(x,y)\in\mathcal{B}^{i,j}\}}\right] \\
&= \mathbb{E}_\mu\Bigg[\Delta w^\intercal \Delta\nabla F(x)^{i,j}\int_0^{\epsilon\Delta L(y)^{i,j}} d\Delta F_w^{j,i}\int_0^\infty d\Delta F_w^{j,1}\cdots\int_0^\infty d\Delta F_w^{j,n} \\
&\int_{-\epsilon\Delta L(y)^{i,1}}^\infty d\Delta F_w^{i,1}\cdots \\
&\int_{-\epsilon\Delta L(y)^{i,n}}^\infty f(\Delta F_w^{j,1}\cdots\Delta F_w^{i,n}|\Delta w^\intercal\Delta\nabla F_w(x)^{i,j},\Delta L(y)^{j,i})d\Delta F_w^{i,n}\Bigg].
\end{aligned}$$

Assuming bounded data distribution and a bounded and continuous integrand we can exchange the limit operation and expectation to get

$$\begin{aligned}
&\lim_{\epsilon\to0^+}\frac{1}{\epsilon}\mathbb{E}\left[\Delta w^\intercal \Delta\nabla F(x)^{i,j}\mathbf{1}_{\{(x,y)\in\mathcal{B}^{i,j}\}}\right] \\
&= \mathbb{E}_\mu\Bigg[(\Delta L(y)^{i,j})^+\Delta w^\intercal\Delta\nabla F_w^{i,j}\lim_{\epsilon\to0^+}\frac{1}{\epsilon}\int_0^\epsilon d\Delta F_w^{j,i}\int_0^\infty d\Delta F_w^{j,1}\cdots\int_0^\infty d\Delta F_w^{j,n} \qquad (12)\\
&\int_0^\infty d\Delta F_w^{i,1}\cdots\int_0^\infty f(\Delta F_w^{j,1}\cdots\Delta F_w^{i,n}|\Delta w^\intercal\Delta\nabla F_w(x)^{i,j},\Delta L(y)^{j,i})d\Delta F_w^{i,n}\Bigg].
\end{aligned}$$

Next we group expectations for a change from label i to configuration j and the reverse. To this end we first consider the resulting Eq. (11) obtained from rephrasing Eq. (9). Combining both label change directions yields

$$\begin{aligned}
&\lim_{\epsilon\to0^+}\frac{1}{\epsilon}\mathbb{E}\left[\Delta L(y)^{i,j}\mathbf{1}_{\{x\in\mathcal{A}^{i,j}\}}\right]+\lim_{\epsilon\to0^+}\frac{1}{\epsilon}\mathbb{E}\left[\Delta L(y)^{j,i}\mathbf{1}_{\{x\in\mathcal{A}^{j,i}\}}\right] \\
&= \mathbb{E}_\mu\Bigg[\Delta L(y)^{i,j}(\Delta w^\intercal\nabla\Delta F_w^{i,j})^+\lim_{\epsilon\to0^+}\frac{1}{\epsilon}\int_0^\epsilon d\Delta F_w^{j,i}\int_0^\infty d\Delta F_w^{j,1}\cdots\int_0^\infty d\Delta F_w^{j,n} \\
&\int_0^\infty d\Delta F_w^{i,1}\cdots\int_0^\infty f(\Delta F_w^{j,1}\cdots\Delta F_w^{i,n}|\Delta L(y)^{i,j},\Delta w^\intercal\nabla\Delta F_w^{i,j})d\Delta F_w^{i,n}\Bigg] \\
&+\mathbb{E}_\mu\Bigg[\Delta L(y)^{j,i}(\Delta w^\intercal\nabla\Delta F_w^{j,i})^+\lim_{\epsilon\to0^+}\frac{1}{\epsilon}\int_0^\epsilon d\Delta F_w^{i,j}\int_0^\infty d\Delta F_w^{i,1}\cdots\int_0^\infty d\Delta F_w^{i,n} \qquad (13)\\
&\int_0^\infty d\Delta F_w^{j,1}\cdots\int_0^\infty f(\Delta F_w^{j,1}\cdots\Delta F_w^{i,n}|\Delta L(y)^{i,j},\Delta w^\intercal\nabla\Delta F_w^{i,j})d\Delta F_w^{i,n}\Bigg] \\
&= \mathbb{E}_\mu\Bigg[\Delta L(y)^{i,j}\Delta w^\intercal\nabla\Delta F_w^{i,j}\lim_{\epsilon\to0^+}\frac{1}{\epsilon}\int_0^\epsilon d\Delta F_w^{j,i}\int_0^\infty d\Delta F_w^{j,1}\cdots\int_0^\infty d\Delta F_w^{j,n} \\
&\int_0^\infty d\Delta F_w^{i,1}\cdots\int_0^\infty f(\Delta F_w^{j,1}\cdots\Delta F_w^{i,n}|\Delta L(y)^{i,j},\Delta w^\intercal\nabla\Delta F_w^{i,j})d\Delta F_w^{i,n}\Bigg].
\end{aligned}$$

Similarly, we group expectations for both label change directions for the resulting Eq. (12) obtained from rephrasing Eq. (10), which yields

$$\begin{aligned}
&\lim_{\epsilon\to0^+}\frac{1}{\epsilon}\mathbb{E}\left[\Delta w^\intercal \Delta\nabla F(x)^{i,j}\mathbf{1}_{\{(x,y)\in\mathcal{B}^{i,j}\}}\right]+\lim_{\epsilon\to0^+}\frac{1}{\epsilon}\mathbb{E}\left[\Delta w^\intercal \Delta\nabla F(x)^{j,i}\mathbf{1}_{\{(x,y)\in\mathcal{B}^{j,i}\}}\right] \\
&= \mathbb{E}_\mu\Bigg[\Delta L(y)^{i,j}\Delta w^\intercal\Delta\nabla F_w^{i,j}\lim_{\epsilon\to0^+}\frac{1}{\epsilon}\int_0^\epsilon d\Delta F_w^{j,i}\int_0^\infty d\Delta F_w^{j,1}\cdots\int_0^\infty d\Delta F_w^{j,n} \qquad (14)\\
&\int_0^\infty d\Delta F_w^{i,1}\cdots\int_0^\infty f(\Delta F_w^{j,1}\cdots\Delta F_w^{i,n}|\Delta w^\intercal\Delta\nabla F_w(x)^{i,j},\Delta L(y)^{j,i})d\Delta F_w^{i,n}\Bigg].
\end{aligned}$$

We therefore have equivalence between Eq. (13) and Eq. (14) for a change from configuration i to j and the reverse. Since this holds for all pairwise configurations, Eq. (9) is identical to Eq. (10), which proves the theorem. $\square$

The conditions for the above results to hold are similar to the conditions for the proof for the binary linear case (McAllester et al., 2010). The conditions can be inferred from the proof above. We require that the joint measure ρ can be expressed as a measure μ and a corresponding bounded continuous conditional density function f. For exchangeability of limits and expectations, it is sufficient to require the integrand to be continuous and bounded as well as the range of integral to be bounded, *i.e.*, the range of data is bounded. Further we require the scoring function F to have continuous derivatives w.r.t. w.

5.2 Proof for Lemma 1

Next we provide the proof for Lemma 1 which we repeat for completeness. We note again that the cost function value obtained when restricting loss-augmented inference to the subsets can be computed as:

$$h(i,j) = \max_{\hat{y}} \frac{1}{|\mathcal{P}||\mathcal{N}|} \sum_{m\in\mathcal{P}_i} \sum_{n\in\mathcal{N}_j} \hat{y}_{m,n}(\phi(x_m,w) - \phi(x_n,w)) \pm \epsilon\Delta_{AP}^{i,j}(\text{rank}(y), \text{rank}(\hat{y})), \quad (15)$$

where $\Delta_{AP}^{i,j}$ refers to the AP loss restricted to subsets of i positive and j negative elements.

Lemma 1. *Suppose that* $\text{rank}(\hat{y}^*)$ *is the optimal ranking for Eq.* (15) *when restricted to i positive and j negative samples. Any of its sub-sequences starting at position 1 is then also an optimal ranking for the corresponding restricted sub-problem.*

Proof. We consider the prefix $r_1, r_2, \cdots, r_k$, where $k < |\mathcal{P}| + |\mathcal{N}|$ and $(r_1, r_2, \cdots, r_{|\mathcal{P}|+|\mathcal{N}|}) := \text{rank}(\hat{y}^*)$. Suppose there are i relevant objects and j irrelevant objects in the prefix, and $k = i + j$. What we need to prove is that $r_1, r_2, \cdots, r_{i+j}$ is already an optimal ranking.

Let

$$h(i,j) = \max_{\hat{y}} \underbrace{\frac{1}{|\mathcal{P}||\mathcal{N}|} \sum_{m\in\mathcal{P}_i} \sum_{n\in\mathcal{N}_j} \hat{y}_{m,n}(\phi(x_m,w) - \phi(x_n,w)) \pm \epsilon\Delta_{AP}^{i,j}(\text{rank}(y), \text{rank}(\hat{y}))}_{:=s(i,j)}.$$

We decompose the optimal value obtained when considering all samples, *i.e.*, $h(|\mathcal{P}|, |\mathcal{N}|)$, into three parts:

$$\begin{aligned}
& h(|\mathcal{P}|, |\mathcal{N}|) \\
&= \underbrace{\frac{1}{|\mathcal{P}||\mathcal{N}|} \sum_{m\in\mathcal{P}_i} \sum_{n\in\mathcal{N}_j} \hat{y}^*_{m,n}(\phi(x_m,w) - \phi(x_n,w)) \pm \epsilon\Delta_{\text{AP}}^{i,j}(\text{rank}(y), \text{rank}(\hat{y}^*))}_{\text{Prefix terms} = s(i,j)} \\
&+ \underbrace{\frac{1}{|\mathcal{P}||\mathcal{N}|} \sum_{m\in\mathcal{P}\setminus\mathcal{P}_i} \sum_{n\in\mathcal{N}\setminus\mathcal{N}_j} \hat{y}^*_{m,n}(\phi(x_m,w) - \phi(x_n,w)) \pm \epsilon\Delta_{\text{AP}}^{\neg i,\neg j}(\text{rank}(y), \text{rank}(\hat{y}^*))}_{\text{Suffix terms}} \\
&+ \underbrace{\frac{1}{|\mathcal{P}||\mathcal{N}|} \left[\sum_{m\in\mathcal{P}_i} \sum_{n\in\mathcal{N}\setminus\mathcal{N}_j} \hat{y}^*_{m,n}(\phi(x_m,w) - \phi(x_n,w)) + \sum_{m\in\mathcal{P}\setminus\mathcal{P}_i} \sum_{n\in\mathcal{N}_j} \hat{y}^*_{m,n}(\phi(x_m,w) - \phi(x_n,w)) \right]}_{\text{Cross terms}}.
\end{aligned}$$

Here, $\Delta_{\text{AP}}^{\neg i,\neg j}(\text{rank}(y), \text{rank}(\hat{y}^*))$ refers to the loss obtained by considering all the samples not within the prefix.

Intuitively, when changing the interleaving pattern of the prefix, the suffix terms and cross terms remain the same. This is true since the suffix terms are independent of the ranking of the prefix terms. In addition the cross terms only depend on the number and scores of positive and negative elements in the prefix but not their specific ranking.

More formally, suppose that $f(i,j) \neq h(i,j)$, then we can substitute $h(i,j)$ into the prefix term and get a larger value than $h(|\mathcal{P}|, |\mathcal{N}|)$, contradicting the fact that $h(|\mathcal{P}|, |\mathcal{N}|)$ is already the largest, which concludes the proof. $\square$

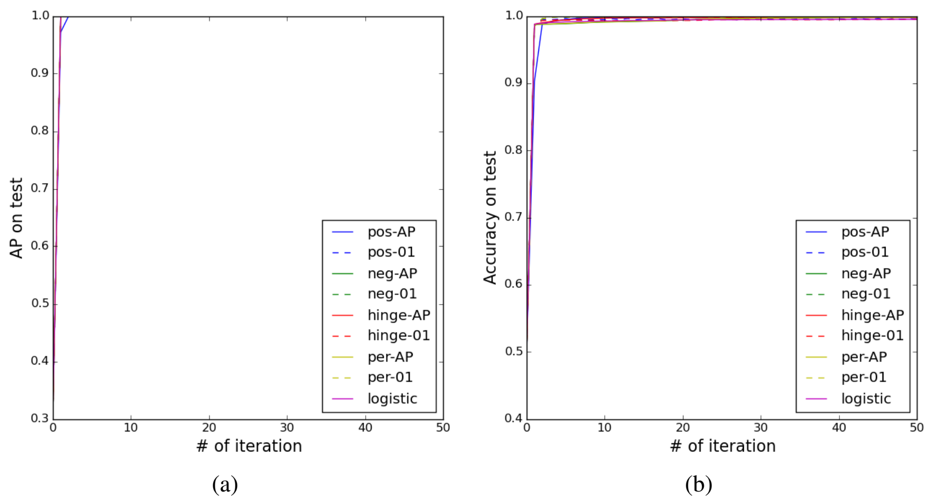


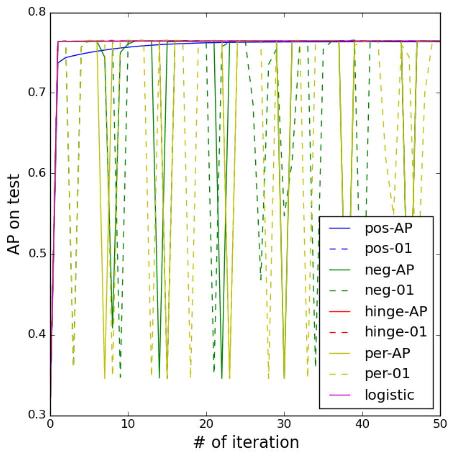
Figure 4: This figure shows the results on linear synthetic data without noise. We illustrate average precision over the number of iterations on the test set in (a) and accuracy over the number of iterations on the test set in (b).

5.3 Experiments on linear synthetic data

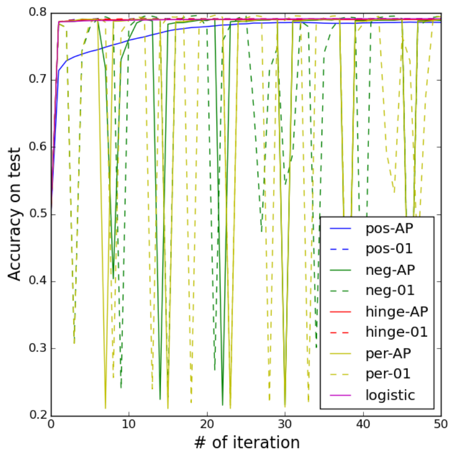
To test the linear case, we generated two different datasets, one of which is linearly separable while the other one is not. We randomly generated 20,000 data points by sampling from a 10 dimensional standard Gaussian distribution. The data points with a sum of numbers in all dimensions being larger than 0 are assigned to the positive class while those having a negative sum are classified as the negative objects. We then divide the whole dataset into training set and test set of 10,000 elements each. To produce the non-linearly separable dataset, we randomly flip 20% of the binary labels. We select $\phi(x, w) = w^\intercal x$ in this linear setting. The results are depicted in Fig. 4 and Fig. 5.

In the noiseless linear case we observe the negative update to achieve a slightly better performance than the positive update. The perceptron method also performs well. We think this is the reason why McAllester et al. (2010) report the negative update to perform better. Note that Cheng et al. (2009) also reported good performance for the perceptron method on the TIMIT dataset, the same one used in McAllester et al. (2010).

Negative and perceptron updates perform similarly on the noisy and not linearly separable dataset. They also do not perform well on our nonlinear datasets shown in the main paper.



(a)



(b)

Figure 5: This figure shows the results on linear synthetic data with 20% noisy labels. We illustrate average precision over the number of iterations on the test set in (a) and accuracy over the number of iterations on the test set in (b).