Adversarial Contrastive Estimation

Anonymous ACL submission

Abstract

Learning by contrasting positive and negative samples is a general strategy adopted by many methods in cases where the original formulation has intractable computational requirements or is difficult to model. Noise Contrastive Estimation (NCE) and negative sampling in word embeddings are examples that make use of this contrastive learning strategy. In this paper, we view contrastive learning as an abstraction of all such methods, and augment the negative sampler into a mixture distribution containing an adversarially learned sampler. The resulting adaptive sampler finds harder negative examples, which forces the main model to learn better representation of the data. We evaluate our proposal on learning word embeddings, order embeddings and knowledge graph embeddings and observe both faster convergence and improved results on multiple metrics.

1 Introduction

Many models learn by contrasting losses on observed positive examples with those on some fictitious negative examples, trying to decrease some score on positive ones while increasing it on negative ones. There are multiple reasons why such contrastive learning approach is needed. Computational tractability is one. For instance, instead of using softmax to predict a word for learning word embeddings, noise contrastive estimation (NCE) can be used in skip-gram or CBOW word embedding models [19]. Another reason is modeling need, as certain assumptions are best expressed as some score or energy in margin based or un-normalized probability models. For example, modeling entity relations as translations or variants thereof in a vector space naturally leads to a distance-based score to be minimized for observed entity-relation-entity triplets [24].

Given a scoring function, the gradient of model parameters on observed positive examples can be readily computed, but the negative phase requires a design decision about how to sample data. In noise contrastive estimation for word embeddings, a negative example is formed by replacing a component of a positive pair by randomly selecting a sampled word from the vocabulary, resulting in a fictitious word-context pair which would be unlikely to actually exist in the dataset. This negative sampling by corruption approach is also used in learning knowledge graph embeddings [3, 17, 15, 29, 26, 31, 8], order embeddings [27], caption generation [7] etc.

Typically the corruption distribution is the same for all inputs like in skip-gram or CBOW NCE [6], rather than being a conditional distribution that takes into account information about the input sample under consideration. Furthermore, the corruption process usually only encodes human prior about what constitutes a hard negative sample rather than being learned from data. For these two reasons, the fixed simple corruption process often yield only easy negative examples. Easy negatives are sub-optimal for learning discriminative representation as they do not force the model to find critical characteristics of observed positive data, which has been observed in applications outside NLP previously [25]. Even if hard negatives are occasionally reached, the infrequency means slow convergence. Designing more sophisticated corruption process could be fruitful, but may require costly trial-and-error by a human expert.

In this work, we propose to augment the simple corruption noise process in various embedding models with an adversarially learned conditional distribution, forming a mixture negative sampler that adapts to the underlying data and the embedding model training progress. The resulting method is referred to as adversarial contrastive estimation (ACE). The adaptive conditional model engages in a minimax game with the primary embedding model, much like in Generative Adversarial Networks (GANs)[11], where a discriminator net (D), tries to distinguish samples produced by a generator (G) from real data [12]. In ACE, the main model learns to distinguish between an real positive example and a negative sample selected by the mixture of a weak fixed NCE sampler and an adversarial generator. The main model and the generator takes alternating turns to update their parameters. In fact, our method can be viewed as a conditional GAN [20] on discrete inputs, with a mixture generator consisting of a learned and a fixed distribution, with additional techniques introduced to achieve stable and convergent training of embedding models.

In our proposed ACE approach, the conditional sampler finds harder negatives than NCE, while being able to gracefully fall back to NCE whenever the discriminator cannot finds hard negatives. We demonstrate the efficacy and generality of the proposed method on three different learning tasks, word embeddings [19], order embeddings [27] and knowledge graph embeddings [15].

2 Method

2.1 Background: contrastive learning

In the most general form, our method applies to supervised learning problems with a contrastive objective of the following form:

$$L(\omega) = \mathbb{E}_{p(x^+,y^+,y^-)}\, l_\omega(x^+,y^+,y^-) \qquad (1)$$

where $l_\omega(x^+,y^+,y^-)$ captures both the model with parameters ω and the loss that scores a positive tuple (x^+,y^+) against a negative one (x^+,y^-). $\mathbb{E}_{p(x^+,y^+,y^-)}(.)$ denotes expectation with respect to some joint distribution over positive and negative samples. Furthermore, by the law of total expectation, and the fact that given x^+, the negative sampling is not dependent on the positive label, i.e. $p(y^+,y^-|x^+) = p(y^+|x^+)p(y^-|x^+)$, Eq. 1 can be re-written as

$$\mathbb{E}_{p(x^+)}\left(\mathbb{E}_{p(y^+|x^+)p(y^-|x^+)}\, l_\omega(x^+,y^+,y^-)\right) \qquad (2)$$

Separable loss

In the case where the loss decomposes into a sum of two terms as $l_\omega(x^+,y^+,y^-) = s_\omega(x^+,y^+) - \tilde{s}_\omega(x^+,y^-)$, then Expression. 2 becomes

$$\mathbb{E}_{p^+(x)}(\mathbb{E}_{p^+(y|x)}\, s_\omega(x,y) - \mathbb{E}_{p^-(y|x)}\, \tilde{s}_\omega(x,y)) \qquad (3)$$

where we moved the $+$ and $-$ to p for notational brevity. Learning by stochastic gradient descent aims to adjust ω to pushing down $s_\omega(x,y)$ on samples from p^+ while pushing up $\tilde{s}_\omega(x,y)$ on samples from p^-. Note that for generality, the scoring function for negative samples, denoted by $\tilde{s}_\omega$, could be slightly different from s_ω. For instance, $\tilde{s}$ could contain a margin as in the case of Order Embeddings in Sec. 4.2.

Non separable loss

Eq. 1 is the general form that we would like to consider because for certain problems, the loss function cannot be separated into sums of terms containing only positive (x^+,y^+) and terms with negatives (x^+,y^-). An example of such non-separable loss is the triplet ranking loss [24]: $l_\omega = \max(0, \eta + s_\omega(x^+,y^+) - s_\omega(x^+,y^-))$, which does not decompose due to the rectification.

Noise contrastive estimation

The typical NCE [6] approach in tasks such as word embeddings[19], order embeddings[27], and knowledge graph embeddings can be viewed as a special case of Expression. 2 by taking $p(y^-|x^+)$ to be some unconditional $p_{nce}(y)$.

This leads to efficient computation during training, however, $p_{nce}(y)$ sacrifices the sample efficiency of learning as the negatives produced by a fixed distribution, which does not tailor toward x^+, are not necessarily hard negative examples, and would not force the model to discover discriminative representation of observed positive data. As training progresses, more and more negative examples are correctly learned, the probability of drawing a hard negative examples diminishes further, causing slow convergence.

2.2 Adversarial mixture noise

To remedy the above mentioned problem of a fixed unconditional negative sampler, we propose to augment it into a mixture one $\lambda p_{nce}(y) + (1-\lambda)g_\theta(y|x)$, where g_θ is a conditional distribution with a learnable parameter θ and λ is a hyperparameter. The objective Expression. 2 can then be written as (conditioned on x for notational brevity):

$$L(\omega,\theta;x) = \lambda\, \mathbb{E}_{p(y^+|x)p_{nce}(y^-)}\, l_\omega(x,y^+,y^-) \\ + (1-\lambda)\, \mathbb{E}_{p(y^+|x)g_\theta(y^-|x)}\, l_\omega(x,y^+,y^-) \quad (4)$$

We learn (ω,θ) in a GAN-style minimax game:

$$\min_\omega \max_\theta V(\omega,\theta) = \min_\omega \max_\theta \mathbb{E}_{p^+(x)}\, L(\omega,\theta;x) \quad (5)$$

The embedding model behind $l_\omega(x,y^+,y^-)$ is similar to the discriminator in (conditional) GAN (or critic in Wasserstein[1] or Energy-based GAN[32], while $g_\theta(y|x)$ acts as the generator. Henceforth, we will use the term discriminator (D) and embedding model interchangeably, and refer to g_θ as the generator.

2.3 Learning the generator

There is one important distinction to typical GAN: $g_\theta(y|x)$ defines a categorical distribution over possible y values, and samples are drawn accordingly; in contrast to typical GAN over continuous data space such as images, where samples are generated by an implicit generative model that warps noise vectors into data points. Due to the discrete sampling step, g_θ cannot learn by receiving gradient through the discriminator. One possible solution is to use the Gumbel-softmax reparametrization trick [14], which gives a differentiable approximation. However, this differentiability comes at the cost of drawing N Gumbel samples per each categorical sample, where N is the number of categories. For word embeddings, N is the vocabulary size, and for knowledge graph embeddings, N is the number of entities, both leading to infeasible computational requirements.

Instead, we use the REINFORCE ([30]) gradient estimator for $\nabla_\theta L(\theta,x)$:

$$(1-\lambda)\, \mathbb{E}\left[-l_\omega(x,y^+,y^-)\nabla_\theta \log(g_\theta(y^-|x))\right] \quad (6)$$

where the expectation $\mathbb{E}$ is with respect to $p(y^+,y^-|x) = p(y^+|x)g_\theta(y^-|x)$, and the discriminator loss $l_\omega(x,y^+,y^-)$ acts as the reward.

With a separable loss, the (conditional) value function of the minimax game is:

$$L(\omega,\theta;x) = \mathbb{E}_{p^+(y|x)}\, s_\omega(x,y) \\ - \mathbb{E}_{p_{nce}(y)}\, \tilde{s}_\omega(x,y) - \mathbb{E}_{g_\theta(y|x)}\, \tilde{s}_\omega(x,y) \quad (7)$$

and only the last term depends on the generator parameter ω. Hence, with a separable loss, the reward is $-\tilde{s}(x^+,y^-)$. This reduction does not happen with a non-separable loss, and we have to use $l_\omega(x,y^+,y^-)$.

2.4 Entropy and training stability

GAN training can suffer from instability and degeneracy where the generator probability mass collapses to a few modes or points. Much work has been done to stabilize GAN training in the continuous case [1, 13, 5]. In ACE, if the generator g_θ probability mass collapses to a few candidates, then after the discriminator successfully learns about these negatives, g_θ cannot adapt to select new hard negatives, because the REINFORCE gradient estimator Eq. 6 relies on g_θ being able to explore other candidates during sampling. Therefore, if g_θ probability mass collapses, instead of leading to oscillation in typical GAN, the minmax game in ACE reaches an equilibrium where the discriminator wins and g_θ can no longer adapt, then ACE falls back to NCE, since the negative sampler has another mixture component from NCE.

This behavior of gracefully falling back to NCE is more desirable than the alternative of stalled training if $p^-(y|x)$ does not have a simple p_{nce} mixture component. However, we would still like to avoid such collapse, as the adversarial samples provide greater learning signals than NCE samples. To this end, we propose to use a regularizer to encourage the categorical distribution $g_\theta(y|x)$ to have high entropy. In order to make the the regularizer interpretable and its hyperparameters easy to tune, we design the following form:

$$R_{ent}(x) = min(0, c - H(g_\theta(y|x))) \quad (8)$$

where $H(g_\theta(y|x))$ is the entropy of the categorical distribution $g_\theta(y|x)$, and $c = \log(k)$ is the entropy of a uniform distribution over k choices, and k is a hyper-parameter. Intuitively, R_{ent} expresses the prior that the generator should spread its mass over more than k choices for each x.

2.5 Handling false negatives

During negative sampling, $p^-(y|x)$ could actually produce y that forms a positive pair that exists in the training set, i.e a false negative. This possibility exists in NCE already, but since p_{nce} is not adaptive, the probability of sampling a false negative is low. Hence in NCE, the score on this false negative (true observation) pair is pushed up less hard in the negative term than in the positive term.

However, with the adaptive sampler $g_\omega(y|x)$, false negatives becomes a much more severe issue. $g_\omega(y|x)$ can learn to concentrate its mass on a few false negatives, significantly canceling the learning of those observations in the positive phase. The entropy regularization reduces this problem as it forces the generator to spread its mass, hence reducing the chance of false negative.

To further alleviate this problem, whenever computationally feasible, we apply an additional two-step technique: first, we maintain a hash map of the training data in memory, and use it to efficiently detect if a negative sample (x^+, y^-) is an actual observation, if so, its contribution to the loss is given a zero weight in ω learning step; second, to upate θ in the generator learning step, the reward for false negative samples are replaced by a large penalty, so that the REINFORCE gradient update would steer g_θ away from those samples. The second step is needed to prevent null computation where g_θ learns to sample false negatives which are subsequently ignored by the discriminator update for ω.

2.6 Variance Reduction

The basic REINFORCE gradient estimator is poised with high variance, so in practice one often needs to apply variance reduction technique. The most basic form of variance reduction is to subtract a baseline from the reward. As long as the baseline is not a function of actions (i.e. samples y^- being drawn), the REINFORCE gradient estimator remains unbiased. We propose to use the self-critical baseline method [22], where the baseline is $b(x) = l_\omega(y^+, y^\star, x)$, or $b(x) = -\tilde{s}_\omega(y^\star, x)$ in the separable loss case, and $y^\star = \text{argmax}_i g_\theta(y_i|x)$. In other words, the baseline is the reward of the most likely sample according to the generator.

2.7 Improving exploration in g_θ by leveraging NCE samples

In Sec. 2.4 we briefly touched on the need for sufficient exploration in g_θ. It is possible to also leverage negative samples from NCE to help the generator learning. This is essentially off-policy exploration in reinforcement learning, since NCE samples are not drawn according to $g_\theta(y|x)$. The generator learning can use importance re-weighting to leverage those samples. The resulting REINFORCE gradient estimator is basically the same as Eq. 6 except that the rewards are reweighted by $g_\theta(y^-|x)/p_{nce}(y^-)$, and the expectation is with respect to $p(y^+|x)p_{nce}(y^-)$. This additional off policy learning term provides gradient information for generator learning if $g_\theta(y^-|x)$ is not zero, meaning that for it to be effective in helping exploration, the generator cannot be collapsed at the first place. Hence, in practice, this term is only useful to further help on top of the entropy regularization, but it does not replace it.

3 Related Work

Concurrent to this work, there has been many interests in applying the GAN approach to NLP problems [9, 28, 4]. Knowledge graph models naturally lend to a GAN setup, and has been the subject of study in [28] and [4]. These two concurrent works are most closely related to one of the three tasks on which we study ACE in this work. Beside a more general formulation that applies to problems beyond those considered in [28] and [4], the techniques introduced in our work on handling false negatives and entropy regularization lead to improved experimental results as shown in Sec. 5.4.

4 Application of ACE on three tasks

4.1 Word Embeddings

Word embeddings learn vector representation of words from co-occurrences in a text corpus. NCE casts this learning problem as a binary classification where the model tries to distinguish positive word and context pairs, from negative noise samples composed of word and false context pairs. The NCE objective in Skip-gram ([19]) for word embeddings is a separable loss of the form:

$$\begin{aligned} L = -\sum_{w_t \in V} [\log p(y=1|w_t, w_c^+) \\ + \sum_{c=1}^{K} \log p(y=0|w_t, w_c^-)] \end{aligned} \tag{9}$$

Here, w_c^+ is sampled from the true set of contexts and $w_c^- \sim Q$ is sampled k times from a fixed noise distribution. Mikolov *et al.* [19] introduced a further simplification of NCE, called ”Negative Sampling”. With respect to our ACE framework, the difference between NCE and Negative Sampling is inconsequential, so we continue the discussion using NCE. A drawback of this sampling scheme is that it favors more common word as context. Another issue is that the negative context words are sampled in the same way, rather than tailored toward the actual target word. To apply ACE to this problem we first define the value function for the minimax game, $V(D,G)$, as follows:

$$\begin{aligned} V(D,G) &= \mathbb{E}_{p^+(w_c)}[\log D(w_c,w_t)] \\ &- \mathbb{E}_{p_{nce}(w_c)}[-\log(1-D(w_c,w_t))] \\ &- \mathbb{E}_{g_\theta(w_c|w_t)}[-\log(1-D(w_c,w_t))] \end{aligned} \tag{10}$$

with $D = p(y=1|w_t,w_c)$ and $G = g_\theta(w_c|w_t)$.

Implementation details

For our experiments we train all our models on lowercased unigrams taken from one pass through the May 2017 dump of the English Wikipedia. The vocabulary size is restricted to the top $150k$ most frequent words when training from scratch while for finetuning we use the same vocabulary as [21]. We use 5 NCE samples for each positive sample and 1 adversarial sample in a window size of 10 and the same positive subsampling scheme proposed by [19]. Learning for both G and D uses Adam [16] optimizer with its default parameters. Our conditional discriminator is modeled using the Skip-Gram architecture, which is a two layer neural network with a linear mapping between the layers. The generator network consists of an embedding layer followed by two small hidden layers, followed by output softmax layer. The first layer of generator shares its weights with second embedding layer in the discriminator network, which we find really speeds up convergence as the generator doesn't have to relearn its own set of embeddings. The difference between the discriminator and generator is that a sigmoid nonlinearity is used after the second layer in the discriminator, while in the generator, a softmax layer is used to define categorical distribution over negative word candidates. We find that controlling the generator entropy is critical for finetuning experiments as otherwise the generator collapses to its favorite negative sample. The word embeddings are taken to be the first dense matrix in the conditional discriminator.

4.2 Order Embeddings Hypernym Prediction

As introduced in [27], ordered representations over hierarchy can be learned by order embeddings. An example task for such ordered representation is hypernym prediction. A hypernym pair is a pair of concepts where the first concept is a specialization or an instance of the second.

For completeness, we briefly describe order embeddings, then analyze ACE on the hypernym prediction task. In order embeddings, each entity is represented by a vector in $\mathbb{R}^N$, the score for a positive ordered pair of entities (x,y) is defined by $s_\omega(x,y) = ||max(0,y-x)||^2$ and score for a negative ordered pair (x^+,y^-) is defined by $\tilde{s}_\omega(x^+,y^-) = \max\{0,\eta - s(x^+,y^-)\}$, where is η is margin. Let $f(u)$ be the embedding function which takes an entity as input and outputs en embedding vector. And define P as a set of positive pairs and N as negative pairs, the separable loss function for order embedding task is defined by:

$$L = \sum_{(u,v)\in P} s_\omega(f(u),f(v))) + \sum_{(u,v)\in N} \tilde{s}(f(u),f(v)) \tag{11}$$

Implementation details

Our generator for this task is just a linear fully connected softmax layer, taking an embedding vector from discriminator as input and outputting a categorical distribution over the entity set. For the discriminator, we inherit all model setting from [27], we use 50 dimensions hidden state and bash size 1000, we use learning rate 0.01 and Adam optimizer. For generator, we use batch size 1000, learning rate 0.01 and Adam optimizer. We apply weight decay with rate 0.1 and entropy loss regularization as described in section 2.4. We handle false negative as described in section 2.5. After cross validation, variance reduction and leveraging NCE samples doesn't greatly effect the order embedding task.

4.3 Knowledge Graph Embeddings

Knowledge graphs contain entity and relation data of the form *(head entity, relation, tail entity)*, and the goal of is to learn from observed positive entity relations and predict missing links (aka link prediction). There has been many works on knowledge graph embeddings, e.g. TransE[3], TransR[17], TransH[29], TransD[15], Complex[26], DistMult[31] and ConvE[8], many of them use a contrastive learning objective. Here we take TransD as an example, and modify its noise contrastive learning to ACE, and demonstrate significant improvement in sample efficiency and link prediction results.

Implementation details

Let a positive entity-relation-entity triplet be denoted by $\xi^+ = (h^+,r^+,t^+)$, and a negative triplet could either have its head or tail be a negative sample, i.e. $\xi^- = (h^-,r^+,t^+)$ or $\xi^- = (h^+,r^+,t^-)$. In either case, the general formulation in Sec. 2.1

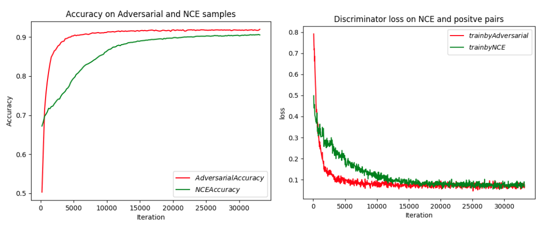

Figure 1: **Left**: Order embedding Accuracy plot. **Right**: Order embedding discriminator Loss plot on NCE sampled negative pairs and positive pairs.

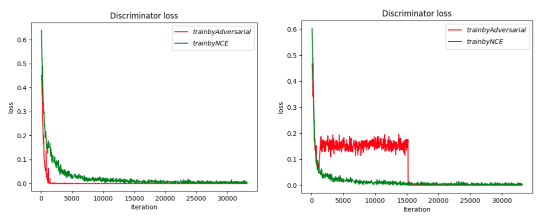

Figure 2: loss curve on NCE negative pairs and ACE negative pairs. **Left**: without entropy and weight decay. **Right**: with entropy and weight decay

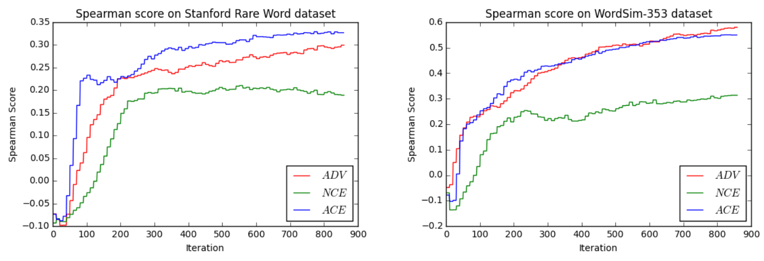

Figure 3: **Left**: Rare Word, **Right**: WS353 similarity scores during the first epoch of training.

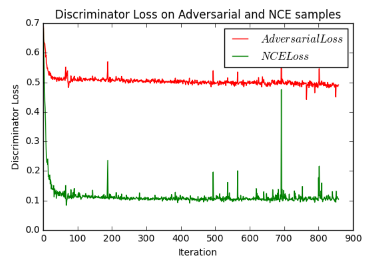

Figure 4: Training from scratch losses on the Discriminator

still applies. The non-separable loss function takes on the form:

$$l = \max(0, \eta + s_\omega(\xi^+) - s_\omega(\xi^-)) \quad (12)$$

The scoring rule is:

$$s = \|\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp\| \quad (13)$$

where $\mathbf{r}$ is the embedding vector for r, and $\mathbf{h}_\perp$ is projection of the embedding of h onto the space of $\mathbf{r}$ by $\mathbf{h}_\perp = \mathbf{h} + \mathbf{r}_p\mathbf{h}_p^\top\mathbf{h}$, where $\mathbf{r}_p$ and $\mathbf{h}_p$ are projection parameters of the model. $\mathbf{t}_\perp$ is defined in a similar way through parameters $\mathbf{t}$, $\mathbf{t}_p$ and $\mathbf{r}_p$.

The form of the generator $g_\theta(t^-|r^+, h^+)$ is chosen to be $f_\theta(\mathbf{h}_\perp, \mathbf{h}_\perp + \mathbf{r})$, where f_θ is a feedforward neural net that concatenates its two input arguments, then propagates through two hidden layers, followed by a final softmax output layer. As a function of (r^+, h^+), g_θ shares parameter with the discriminator, as the inputs to f_θ are the embedding vectors, but during generator learning, only θ is updated, the TransD model embedding parameters are frozen.

5 Experiments

We evaluate ACE with experiments on word embeddings, order embeddings and knowledge graph embeddings tasks.

For word embeddings we evaluate models trained from scratch as well as finetuned Glove models [21] on word similarity tasks that consists of computing the similarity between word pairs where the ground truth is an average of human scores. We choose the Rare word dataset [18] and WordSim-353 [10] by virtue of our hypothesis that ACE learns better representations for both rarer and frequent words. We also qualitatively evaluate ACE word embeddings by inspecting the nearest neighbors of selected words.

For the hypernym prediction task, following [27] hypernym pairs are created from WordNet hierarchy's transitive closure. We use the released random development split and test split from [27], which both contain 4000 edges.

For knowledge graph embeddings, we use TransD [15] as our base model, and perform ablation study to analyze behavior of ACE with various add-on features, and confirm that entropy regularization is crucial for good performance in ACE. We also obtain link prediction results that are competitive or superior to state-of-arts on WN18 dataset[2].

5.1 Training Word Embeddings from scratch

In this experiment we empirically observe that training word embeddings using ACE converges significantly faster than NCE after one epoch. As shown in Fig.3 both ACE (mixture of p_{nce} and g_θ) and just g_θ (denoted by ADV) significantly outperforms the NCE baseline, with an absolute of 73.1% and 58.5% respectively on RW score. We note similar results on WordSim-353 dataset where ACE and ADV outperforms NCE by 40.4% and 45.7%. We also evaluate our model qualitatively by inspecting the nearest neighbors of selected words in Table. 1. We first present the five

Table 1: Top 5 Nearest Neighbors of Words followed by Neighbors 45-50 for different Models.

	Queen	King	Computer	Man	Woman
Skip-Gram NCE Top 5	princess	prince	computers	woman	girl
	king	queen	computing	boy	man
	empress	kings	software	girl	prostitute
	pxqueen	emperor	microcomputer	stranger	person
	monarch	monarch	mainframe	person	divorcee
Skip-Gram NCE Top 45-50	sambiria	eraric	hypercard	angiomata	suitor
	phongsri	mumbere	neurotechnology	someone	nymphomaniac
	safrit	empress	lgp	bespectacled	barmaid
	mcelvoy	saxonvm	pcs	hero	redheaded
	tsarina	pretender	keystroke	clown	jew
Skip-Gram ACE Top 5	princess	prince	software	woman	girl
	prince	vi	computers	girl	herself
	elizabeth	kings	applications	tells	man
	duke	duke	computing	dead	lover
	consort	iii	hardware	boy	tells
Skip-Gram ACE Top 45-50	baron	earl	files	kid	aunt
	abbey	holy	information	told	maid
	throne	cardinal	device	revenge	wife
	marie	aragon	design	magic	lady
	victoria	princes	compatible	angry	bride

nearest neighbors to each word to show that both NCE and ACE models learn sensible embeddings. We then show that ACE embeddings have much better semantic relevance in larger neighborhood (nearest neighbor 45-50).

5.2 Finetuning Word Embeddings

We take off-the-shelf pre-trained Glove embeddings which were trained using 6 billion tokens [21] and fine-tune them using our algorithm. It is interesting to note that the original Glove objective does not fit into the contrastive learning framework, but nonetheless we find that they benefit from ACE. In fact, we observe that training such that 75% of the words appear as positive contexts is sufficient to beat the largest dimensionality pre-trained Glove model on word similarity tasks. We evaluate our performance on the Rare Word and WordSim353 data. As can be seen from our results in Table. 2, ACE on RW is not always better and for the 100d and 300d Glove embeddings is marginally worse. However, on WordSim353 ACE does considerably better across the board to the point where 50d Glove embeddings outperform the 300d baseline Glove model.

5.3 Hypernym Prediction

As shown in Fig.3, with ACE training, our method achieve 1.5% improvement on accuracy over [27] without tunning any discriminator's hyperparameters. We further report training curve in Fig. 1, we report loss curve on random sampled pairs. We stress that in ACE model, we train random pairs and generator generated pairs jointly, as shown in Fig. 2, hard negative helps order embedding model converges faster.

Table 2: Spearman score ($\rho * 100$) on RW and WS353 Datasets. We trained a skipgram model from scratch under various settings for only 1 epoch on wikipedia. For finetuned models we recomputed the scores based on the publicly available 6B tokens Glove models and we finetuned until roughly 75% of the vocabulary was seen.

	RW	WS353
Skipgram Only NCE baseline	18.90	31.35
Skipgram + Only ADV	29.96	58.05
Skipgram + ACE	32.71	55.00
Glove-50 (Recomputed based on[21])	34.02	49.51
Glove-100 (Recomputed based on[21])	36.64	52.76
Glove-300 (Recomputed based on[21])	**41.18**	60.12
Glove-50 + ACE	35.60	60.46
Glove-100 + ACE	36.51	63.29
Glove-300 + ACE	40.57	**66.50**

Table 3: Order Embedding Performance

Method	Accuracy (%)
order-embeddings	90.6
order-embeddings + Our ACE	**92.0**

5.4 Ablation Study and Improving TransD

To analyze different aspects of ACE, we perform ablation study on the knowledge graph embedding task. As described in Sec. 4.3, the base model (discriminator) we apply ACE to is TransD[15]. Fig. 5 shows validation performance as training progresses. All variants of ACE converges to better results than base NCE. Among ACE variants, all methods that include entropy regularization significantly outperform without entropy regularization. Without the self critical baseline variance reduction, learning could progress faster at the beginning but the final performance suffer slightly. The best performance is obtained without the additional off-policy learning of the generator.

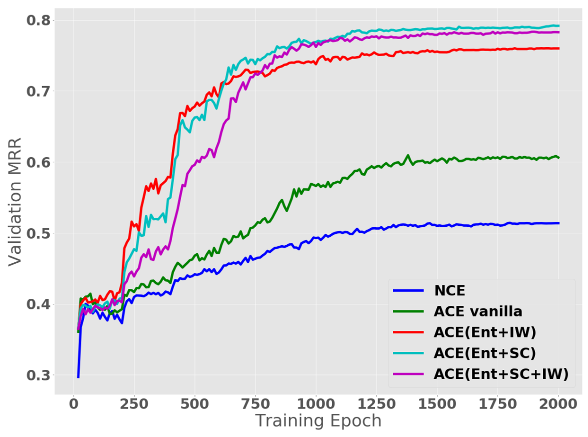

Figure 5: Ablation study: measuring validation Mean Reciprocal Rank (MRR) on WN18 dataset as training progresses.

Table. 4 shows the final test results on WN18 link prediction task. It is interesting to note that ACE improves MMR score more significantly than hit@10. Because MRR is a lot more sensitive to the top rankings, i.e. how the correct configuration ranks among the competitive alternatives, this is consistent with the fact that ACE samples hard negatives and force the base model to learn more discriminative representation of the positive examples.

5.5 Hard Negative Analysis

To better understand the effect of the adversarial samples proposed by the generator we plot the discriminator loss on both p_{nce} and g_θ samples. In this context, a harder sample means a higher loss assigned by the discriminator. Fig. 4 shows that discriminator loss for the word embedding task on g_θ samples are always higher than on p_{nce} samples, confirming that the generator is indeed sampling harder negatives.

	MRR	hit@10
ACE(Ent+SC)	<u>0.792</u>	0.945
ACE(Ent+SC+IW)	0.768	<u>**0.949**</u>
NCE TransD (ours)	0.527	0.947
NCE TransD ([15])	-	0.925
KBGAN(DISTMULT) ([4])	0.772	0.948
KBGAN(COMPLEX) ([4])	0.779	0.948
Wang et al. ([28])	-	0.93
COMPLEX ([26])	**0.941**	0.947

Table 4: WN18 experiments: the first potion of the table contains results where the base model is TransD, the last separated line is the COMPLEX embedding model[26], which achieves the SOTA on this dataset. Among all TransD based models (best results in this group is underlined), ACE improves over basic NCE and another GAN based approach KBGAN. The gap on MRR is likely due to the difference between TransD and COMPLEX models.

For Hypernym Prediction task, Fig.2 shows discriminator loss on negative pairs sampled from NCE and ACE respectively. The higher the loss the harder the negative pair is. As indicated in left plot, loss on ACE negative terms collapse faster than NCE negatives. After adding entropy regularization and weight decay, generator works as expected.

6 Conclusion

In this paper we propose Adversarial Contrastive Estimation as a general technique for improving supervised learning problems that learn by contrasting observed and fictitious samples. Specifically, we use a generator network in a conditional GAN like setting to propose hard negative examples for our discriminator model. We find that a mixture distribution of randomly sampling negative examples along with an adaptive negative sampler leads to improved performances on a variety of embedding tasks. We validate our hypothesis that hard negative examples are critical to optimal learning and can be proposed via our ACE framework. Finally, we find that controlling the entropy of the generator through a regularization term is crucial in preventing generator mode collapse and successful training.

Using ACE leads to much faster convergence but there is computational overhead caused by the generator network learning. Techniques such as [23] is a promising future direction to scaling up ACE to even larger datasets.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, 2014.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

[4] Liwei Cai and William Yang Wang. Kbgan: Adversarial learning for knowledge graph embeddings. *arXiv preprint arXiv:1711.04071*, 2017.

[5] Yanshuai Cao, Gavin Weiguang Ding, Kry Yik-Chau Lui, and Ruitong Huang. Improving gan training via binarized representation entropy (bre) regularization. *International Conference on Learning Representations*, 2018. accepted as poster.

[6] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005.

[7] Bo Dai and Dahua Lin. Contrastive learning for image captioning. In *Advances in Neural Information Processing Systems*, pages 898–907, 2017.

[8] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. *arXiv preprint arXiv:1707.01476*, 2017.

[9] William Fedus, Ian Goodfellow, and Andrew M Dai. Maskgan: Better text generation via filling in the _. *arXiv preprint arXiv:1801.07736*, 2018.

[10] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM, 2001.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[13] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779, 2017.

[14] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[15] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 687–696, 2015.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. 2015.

[18] Thang Luong, Richard Socher, and Christopher D Manning. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113, 2013.

[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[20] M. Mirza and S. Osindero. Conditional Generative Adversarial Nets. *ArXiv e-prints*, November 2014.

[21] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[22] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. *arXiv preprint arXiv:1612.00563*, 2016.

[23] Francisco JR Ruiz, Michalis K Titsias, and David M Blei. Scalable large-scale classification with latent variable augmentation.

[24] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[25] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–769, 2016.

[26] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080, 2016.

[27] Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. Order-embeddings of images and language. In *ICLR*, 2016.

[28] Peifeng Wang, Shuangyin Li, and Rong Pan. Incorporating gan for negative sampling in knowledge representation learning. 2018.

[29] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119. AAAI Press, 2014.

[30] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[31] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

[32] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.