Online Learning of Task-specific Word Representations with a Joint Biconvex Passive-Aggressive Algorithm

Pascal Denis¹ and Liva Ralaivola²,³

¹MAGNET, Inria Lille – Nord Europe, Villeneuve d’Ascq, France
pascal.denis@inria.fr
²QARMA, LIF, Aix-Marseille University, Marseille, France
³Institut Universitaire de France
liva.ralaivola@lif.univ-mrs.fr

Abstract

This paper presents a new, efficient method for learning task-specific word vectors using a variant of the Passive-Aggressive algorithm. Specifically, this algorithm learns a word embedding matrix in tandem with the classifier parameters in an online fashion, solving a biconvex constrained optimization at each iteration. We provide a theoretical analysis of this new algorithm in terms of regret bounds, and evaluate it on both synthetic data and NLP classification problems, including text classification and sentiment analysis. In the latter case, we compare various pre-trained word vectors to initialize our word embedding matrix, and show that the matrix learned by our algorithm vastly outperforms the initial matrix, with performance results comparable or above the state-of-the-art on these tasks.

1 Introduction

Recently, distributed word representations have become a crucial component of many natural language processing systems (Koo et al., 2008; Turian et al., 2010; Collobert et al., 2011). The main appeal of these word embeddings is two-fold: they can be derived directly from raw text data in an unsupervised or weakly-supervised manner, and their latent dimensions condense interesting distributional information about the words, thus allowing for better generalization while also mitigating the presence of rare and unseen terms. While there are now many different spectral, probabilistic, and deep neural approaches for building vectorial word representations, there is still no clear understanding as to which syntactic and semantic information they really capture and whether or how these representations really differ (Chen et al., 2013; Levy and Goldberg, 2014b; Schnabel et al., 2015). Also poorly understood is the relation between the word representations and the particular learning algorithm (e.g., whether linear or non-linear) that uses them as input (Wang and Manning, 2013).

What seems clear, however, is that there is no single best embedding and that their impact is very much task-dependent. This in turn raises the question of how to learn word representations that are adapted to a particular task and learning objective. Three different research routes have been explored towards learning task-specific word embeddings. A first approach (Collobert et al., 2011; Maas et al., 2011) is to learn the embeddings for the target problem jointly with additional unlabeled or (weakly-)labeled data in a semi-supervised or multi-task approach. While very effective, this joint training typically requires large amounts of data and often prohibitive processing times in the case of multi-layer neural networks (not to mention their lack of theoretical learning guarantees in part due to their strong non-convexity). Another approach consists in training word vectors using some existing algorithm as in like word2vec (Mikolov et al., 2013) in a way that exploits prior domain knowledge (e.g., by defining more informative, task-specific contexts) (Bansal et al., 2014; Levy and Goldberg, 2014a). In this case, there is still a need for additional weakly- or hand-labeled data, and there is no guarantee that the newly learned embeddings will indeed benefit the performance, as they are trained independently of the task objective. A third approach is to start with some existing pre-trained embeddings and fine-tune them to the task by integrating them in a joint learning objective either using backpropagation (Lebret et al., 2013) or regularized logistic regression (Labutov and Lipson, 2013). These
approaches in effect hit a sweet spot by leveraging pre-trained embeddings and requiring no additional data or domain knowledge, while directly tying them to task learning objective.

Inspired by these latter approaches, we propose a new, online soft-margin classification algorithm, called Re-embedding Passive-Aggressive (or RPA), that jointly learns an embedding matrix in tandem with the model parameters. As its name suggests, this algorithm generalizes the Passive-Aggressive (PA) algorithm of Crammer and Singer (2006) by allowing the data samples to be projected into the lower dimension space defined by the original embedding matrix. Our approach may be seen as extending the work of (Grandvalet and Canu, 2003) which addresses the problem of simultaneously learning the features and the weight vector of an SVM classifier. An important departure, beyond the online nature of RPA, is that it learns a projection matrix and not just a diagonal one (which is essentially what this earlier work does). Our approach and analysis are also related to (Blondel et al., 2014), which tackles non-negative matrix factorization with the PA philosophy.

The main contributions of this paper are as follows. First, we derive a new variant of the Passive-Aggressive algorithm able to jointly learn an embedding matrix along with the weight vector of the model (section 2). Second, we provide theoretical insights as to bound the cumulative squared loss of our learning procedure over any given sequence of examples—the results we give are actually related to our learning procedure over any given sequence of examples—the results we give are actually related to (Grandvalet and Canu, 2003), which addresses the problem of simultaneously learning the features and the weight vector of an SVM classifier. An important departure, beyond the online nature of RPA, is that it learns a projection matrix and not just a diagonal one (which is essentially what this earlier work does). Our approach and analysis are also related to (Blondel et al., 2014), which tackles non-negative matrix factorization with the PA philosophy.

The main contributions of this paper are as follows. First, we derive a new variant of the Passive-Aggressive algorithm able to jointly learn an embedding matrix along with the weight vector of the model (section 2). Second, we provide theoretical insights as to bound the cumulative squared loss of our learning procedure over any given sequence of examples—the results we give are actually related to a learning procedure that slightly differs from the algorithm we introduce but that is more easily and compactly amenable to a theoretical study. Third, we further study the behavior of this algorithm on synthetic data (section 4) and we finally show that it performs well on five real-world NLP classification problems (section 5).

2 Algorithm

We consider the problem of learning a binary linear classification function \( f_{\Phi, w} \), parametrized by both a weight vector \( w \in \mathbb{R}^k \) and an embedding matrix \( \Phi \in \mathbb{R}^{k \times p} \) (typically, with \( k \ll p \)), which is defined as:

\[
f_{\Phi, w} : \mathcal{X} \subset \mathbb{R}^p \rightarrow \{-1, +1\}
\]

\[x \mapsto \text{sign}(\langle w, \Phi x \rangle)\]

We aim at an online learning scenario, wherein both \( w \) and \( \Phi \) will be updated in a sequential fashion. Given a labeled data stream \( S = \{ (x_t, y_t) \}_{t=1}^T \), it seems relevant at each step to solve the following soft-margin constrained optimization problem:

\[
\arg\min_{w \in \mathbb{R}^k, \Phi \in \mathbb{R}^{k \times p}} \frac{1}{2} \| w - w_t \|^2 + \frac{\lambda}{2} \| \Phi - \Phi_t \|^2_F + C \xi^2
\]

s.t. \( \ell_t(w; \Phi; x_t) \leq \xi \) \hspace{1cm} (1b)

where \( \| \cdot \|_2 \), \( \| \cdot \|_F \) stand for the \( l_2 \) and Frobenius norms, respectively, and \( C \) controls the “aggressiveness” of the update (as larger \( C \) values imply updates that are directly proportional to the incurred loss). We define \( \ell_t(w; \Phi; x_t) \) as the hinge loss, that is:

\[
\ell_t(w; \Phi; x_t) = \max(0, 1 - y_t \langle w, \Phi x_t \rangle).
\]

The optimization problem in (1) is reminiscent of the soft-margin Passive-Aggressive algorithm proposed in (Crammer et al., 2006) (specifically, PA-II), but both the objective and the constraint now include a term based on the embedding matrix \( \Phi \). The \( \lambda \) regularization parameter in the objective controls the allowed divergence in the embedding parameters between iterations.

Interestingly, the new objective remains convex, but the margin constraint doesn’t as it involves a multiplicative term between the weight vector and the embedding matrix, making the overall problem bi-convex (Gorski et al., 2007). That is, the problem is convex in \( w \) for fixed values of \( \Phi \) and convex in \( \Phi \) for fixed values of \( w \). Incidentally, the formulation presented by (Labutov and Lipson, 2013) is also bi-convex (as it also involves a similar multiplicative term), although the authors proceed as if it were jointly convex (i.e., convex in both \( w \) and \( \Phi \)). In order to solve this problem, we resort to an alternating update procedure which updates each set of parameters (i.e., either the weight vector or the embedding matrix) while holding the other fixed until some stopping criterion is met (in our case, the value of the objective doesn’t change). As shown in Algorithm 1, this procedure allows us to compute closed-form updates similar to those of PA, and to make use of the same theoretical apparatus for analyzing RPA.

2.1 Unified Formalization

Suppose from now on that \( C \) and \( \lambda \) are fixed and so are \( w_t \) and \( \Phi_t \); this will allow us to drop the explicit dependence on these values and to keep...
Algorithm 1 Re-embedding Passive-Aggressive

Require:
- \( S = \{ (x_t, y_t) \}_{t=1}^T \), a stream of data
- a base embedding \( \Phi_0 \)

Ensure:
- a classification vector \( w \)
- a re-embedding matrix \( \Phi \)

Initialize \( w_0 \)
\( t \leftarrow 0 \)
repeat
\( w_{t+1} \leftarrow w_t, \Phi_{t+1} \leftarrow \Phi_t, \)
\( n \leftarrow 1 \)
repeat
\( w_{t+1}^{n+1} \leftarrow w_{t+1}^{n} + \frac{\ell_t(w_{t+1}; \Phi_{t+1}^{n+1})}{\| \Phi_{t+1}^{n+1} x_t \|^2 + \frac{\lambda}{2} \} y_t \Phi_{t+1}^{n+1} x_t \)
\( \Phi_{t+1}^{n+1} \leftarrow \Phi_{t+1}^{n} + \frac{\ell_t(w_{t+1}; \Phi_{t+1}^{n+1})}{\| w_{t+1}^{n+1} \|^2 + \frac{\lambda}{2} \} y_t w_{t+1}^{n+1} x_t^\top \).
until some stopping criterion is met

return \( w_t, \Phi_t \)

2.2 Bi-convexity

It turns out that problem (5) is a bi-convex optimization problem: it is indeed straightforward to observe that if \( \Phi \) is fixed then the problem is convex in \((w, \xi)\)—it is the classical passive-aggressive II optimization problem—and if \( w \) is fixed then the problem is convex in \((\Phi, \xi)\). If there exist theoretical results on the solving of bi-convex optimization problems, the machinery to use pertains to combinatorial optimization, which might be too expensive. In addition, computing the solution of (5) would drive us away from the spirit of passive-aggressive learning which rely on cheap and statistically meaningful (from the mistake-bound perspective) updates. This is the reason why we propose to resort to an alternate online procedure to solve a proxy of (5).

2.3 An Alternating Online Procedure

Instead of tackling problem (5) directly we propose to solve, at each time \( t \), either of:

\[
\begin{align*}
\hspace{1cm} w_{t+1}, \xi_{t+1} = \arg\min_{w, \xi} Q_t(w, \Phi_t, \xi) \text{ s.t. } q_t(w, \Phi_t) \leq \xi, \\
\end{align*}
\]

(6)

\[
\begin{align*}
\hspace{1cm} \Phi_{t+1}, \xi_{t+1} = \arg\min_{\Phi, \xi} Q_t(w_t, \Phi, \xi) \text{ s.t. } q_t(w_t, \Phi) \leq \xi.
\end{align*}
\]

(7)

This means that the optimization is performed with either \( \Phi \) fixed to \( \Phi_t \) or \( w \) fixed to \( w_t \).

Informally, the iterative Algorithm 1, resulting from this alternate scheme, will solve at each round a simple constrained optimization problem, in which the objective is to minimize the squared Euclidean distances between the new weight vector (resp. the new embedding matrix) and the current one, while making sure that both sets of parameters achieve a correct prediction with a sufficiently high margin. Note that one may recover the standard passive-aggressive algorithm by simply fixing the embedding matrix to the identity matrix. Also note that if the right stopping criteria are retained, Algorithm 1 is guaranteed to converge to a local optimum of (5) (see Gorski et al., 2007).

When fully developed, problems (6)-(7) respectively write as:

\[
\begin{align*}
\hspace{1cm} w_{t+1}, \xi_{t+1} = \arg\min_{w, \xi} \frac{1}{2} ||w - w_t||^2 + C \xi^2 \text{ s.t. } q_t(w, \Phi_t) \leq \xi, \\
\end{align*}
\]

(8a)

\[
\begin{align*}
\hspace{1cm} \Phi_{t+1}, \xi_{t+1} = \arg\min_{\Phi, \xi} \frac{1}{2} ||w_t - w||^2 + C \xi^2 \text{ s.t. } 1 - \langle w, y_t \Phi_t x_t \rangle \leq \xi.
\end{align*}
\]

(8b)
or

\[ \Phi_{t+1}, \xi_{t+1} = \arg\min_{\Phi, \xi} \frac{\lambda}{2} \| \Phi - \Phi \|_F^2 + C \xi^2 \]

s.t. \( 1 - \left\langle \Phi, y_i x_i x_i^\top \right\rangle \leq \xi \).

Using the equivalence in (4), both problems can be seen as special instances of the generic problem:

\[ u^*, \xi^* = \arg\min_{u, \xi} \frac{\lambda}{2} \| u - u_i \|_2^2 + C \xi^2 \]

s.t. \( 1 - \langle u, v_i \rangle \leq \xi \). \hspace{1cm} (8a)

where \( \| \cdot \|_2 \) and \( \langle \cdot, \cdot \rangle \) are generalized versions of the \( l_2 \) norm and the inner product, respectively.

This is a convex optimization problem that can be readily solved using classical tools from convex optimization to give:

\[ u^* = u_t + \tau_u v_t, \text{ with } \tau_u = \frac{\max(0, 1 - \langle u_t, v_t \rangle)}{\| v_t \|_2^2 + \frac{\lambda}{2C}}, \hspace{1cm} (9) \]

which comes from the following proposition.

**Proposition 1.** The solution of (8) is (9).

**Proof.** The Lagrangian associated with the problem is given by:

\[ \mathcal{L}(u, \xi, \tau) = \frac{\lambda}{2} \| u - u_i \|_2^2 + C \xi^2 + \tau (1 - \langle u, v_i \rangle) \]

with \( \tau > 0 \).

Necessary conditions for optimality are \( \nabla_u \mathcal{L} = 0 \), and \( \nabla_\xi \mathcal{L} = 0 \), which imply \( u = u_t + \frac{\tau}{C} v_t \), and \( \xi = \frac{\tau}{C} \). Using this in (10) gives the function:

\[ g(\tau) = \frac{\tau^2 \| v_t \|_2^2}{2\lambda} + \frac{\tau^2}{4C} + \tau - \frac{\tau^2}{2C} - \tau \langle u_t, v_t \rangle - \tau^2 \| v_t \|_2^2, \]

which is maximized with respect to \( \tau \) when \( g'(\tau) = 0 \), i.e.,

\[ \frac{\| v_t \|_2^2}{\lambda} - \frac{1}{2C} - \frac{2 \| v_t \|_2^2}{\lambda} \tau + 1 - \langle u_t, v_t \rangle = 0. \]

Taking into account the constraint \( \tau \geq 0 \), the maximum of \( g \) is attained at \( \bar{\tau} \) with:

\[ \bar{\tau} = \frac{\lambda \max(0, 1 - \langle u_t, v_t \rangle)}{\| v_t \|_2^2 + \frac{\lambda}{2C}}, \]

which, setting \( \tau_u = \bar{\tau} / \lambda \) gives (9). \hfill \Box

The previous proposition allows us to readily have the solutions of (6) and (7) as follows.

**Proposition 2.** The updates induced by the solutions of (6) and (7) are respectively given by:

\[ w^* = w_t + \tau_w y_i \Phi_t x_i, \text{ with } \tau_w = \frac{\ell_t(w_t; \Phi_t)}{\| \Phi_t x_i \|_2^2 + \frac{1}{2C}}, \hspace{1cm} (11) \]

\[ \Phi^* = \Phi_t + \tau_\Phi y_i w_i x_i^\top, \text{ with } \tau_\Phi = \frac{\ell_t(w_t; \Phi_t)}{\| w_t \|_2^2 \| x_i \|_2^2 + \frac{1}{2C}}, \hspace{1cm} (12) \]

where \( \ell_t(w; \Phi) = \max(0, q_t(w, \Phi)) \).

**Proof.** Just instantiate the results of Proposition 1 with \( (u, v_t) = (w, y_i \Phi_t x_i) \) for (11) and \( (u, v_t) = (\Phi, y_i x_i w_i^\top) \) for (12). \hfill \Box

**Remark 1** (Hard-margin case). Note that the hard-margin version of the previous problem:

\[ \arg\min_{\Phi \in \mathbb{R}^{k \times p}} \frac{1}{2} \| w - w_t \|_2^2 + \frac{\lambda}{2} \| \Phi - \Phi_t \|_F^2 \]

s.t. \( 1 - y_i \langle w, \Phi x_i \rangle \leq 0 \)

is degenerate from the alternated optimization point of view. It suffices to observe that the updates entailed by the hard-margin problem correspond to (9) with \( C \) set to \( \infty \); if it happens that either \( \Phi_0 = 0 \) or \( w_0 = 0 \), then one of the optimization problems (wrt to \( w \) or \( \Phi \)) has no solution.

### 3 Analysis

Using the same technical tools as in (Crammer et al., 2006), and the unified formalization of section 2, we have the following result.

**Proposition 3.** Suppose that problem (8) is iteratively solved for a sequence of vectors \( v_1, \ldots, v_T \) to give \( u_2, \ldots, u_{T+1}, u_1 \) being given. Suppose that, at each time step, \( \| v_t \|_2 \leq R \), for some \( R > 0 \). Let \( u^* \) be an arbitrary vector living in the same space as \( u_1 \). The following result holds

\[ \sum_{t=1}^T \xi_t^2 \leq \left( R^2 + \frac{\lambda}{2C} \right) \left( \| u^* - u_1 \|_2^2 + \frac{2C}{\lambda} \sum_{t=1}^T (\xi_t')^2 \right) \]

where

\[ \xi_t = \max(0, 1 - \langle u_t, v_t \rangle), \text{ and } \xi_t' = \max(0, 1 - \langle u^*, v_t \rangle). \]

This proposition and the accompanying lemmas are simply a variation of the results of (Crammer et al., 2006), with the addition that they are based on the generic problem (8). The loss bound applies for a version of Algorithm 1 where one of the parameters, either the weight vector or the re-embedding matrix, is kept fixed for some time.

Proposition 3 makes use of the following lemma (see Lemma 1 in (Crammer et al., 2006)).
Lemma 1. Suppose that problem (8) is iteratively solved for a sequence of vectors \( \mathbf{v}_1, \ldots, \mathbf{v}_T \) to give \( \mathbf{u}_2, \ldots, \mathbf{u}_{T+1} \), \( \mathbf{u}_1 \) being given. The following holds:

\[
\sum_{t=1}^{T} \tau_u (2\ell_t - \tau_u \| \mathbf{v}_t \|_2^2 - 2\ell_t^2) \leq \| \mathbf{u}^* - \mathbf{u}_1 \|_2^2. \tag{17}
\]

Proof. As in (Cramer et al., 2006), simply set \( \Delta_t = \| \mathbf{u}_t - \mathbf{u}^* \|_2^2 - \| \mathbf{u}_{t+1} - \mathbf{u}^* \|_2^2 \) and bound \( \sum_{t=1}^{T} \tau_u \Delta_t \) from above and below. For the upper bound:

\[
\sum_{t=1}^{T} \Delta_t = \| \mathbf{u}_1 - \mathbf{u}^* \|_2^2 - \| \mathbf{u}_{T+1} - \mathbf{u}^* \|_2^2 \leq \| \mathbf{u}_1 - \mathbf{u}^* \|_2^2.
\]

For the lower bound, we focus on the nontrivial situation where \( \ell_t > 0 \), otherwise \( \Delta_t = 0 \) (i.e. no update is made) and the bounding is straightforward. Making use of the value of \( \tau_u \):

\[
\Delta_t = \| \mathbf{u}_t - \mathbf{u}^* \|_2^2 - \| \mathbf{u}_{t+1} - \mathbf{u}^* \|_2^2 = \| \mathbf{u}_t - \mathbf{u}^* \|_2^2 - \| \mathbf{u}_t + \tau_u \mathbf{v}_t - \mathbf{u}^* \|_2^2 \tag{see (8)}
\]

\[
= -2\tau_u (\langle \mathbf{u}_t - \mathbf{u}^*, \mathbf{v}_t \rangle, -\tau_u^2 \| \mathbf{v}_t \|_2^2
\]

\[
= -2\tau_u (\langle \mathbf{u}_t, \mathbf{v}_t \rangle, + \tau_u (\langle \mathbf{u}^*, \mathbf{v}_t \rangle, -\tau_u^2 \| \mathbf{v}_t \|_2^2.
\]

Since \( \ell_t > 0 \), then \( \langle \mathbf{u}_t, \mathbf{v}_t \rangle = 1 - \ell_t \); also, by the definition of \( \ell_t^2 \) (see (16)), \( \ell_t^2 \geq 1 - \langle \mathbf{u}^*, \mathbf{v}_t \rangle, \) or \( \langle \mathbf{u}^*, \mathbf{v}_t \rangle \geq 1 - \ell_t^2 \). This readily gives

\[
\Delta_t \geq -2\tau_u (1 - \ell_t) + 2\tau_u (1 - \ell_t^2) - \tau_u^2 \| \mathbf{v}_t \|_2^2
\]

\[
= 2\tau_u \ell_t - 2\tau_u \ell_t^2 - \tau_u^2 \| \mathbf{v}_t \|_2^2,
\]

hence the targetted lower bound and, in turn, (17).

Proof of Proposition 3. As for the proof of Proposition 3, it suffices, again, to follow the steps given in (Cramer et al., 2006), but this time, for the proof of Theorem 5.

Note that for any \( \beta \neq 0 \), \((\beta \tau_u - \ell_t^2 / \beta)^2\) is well-defined and nonnegative and thus:

\[
\| \mathbf{u}^* - \mathbf{u}_1 \|_2^2
\]

\[
\geq \sum_{t=1}^{T} \tau_u (2\ell_t - \tau_u \| \mathbf{v}_t \|_2^2 - 2\ell_t^2) - (\beta \tau_u - \ell_t^2 / \beta)^2
\]

\[
= \sum_{t=1}^{T} (2\tau_u \ell_t - \tau_u^2 (\| \mathbf{v}_t \|_2^2 + \beta^2) - 2\tau_u \ell_t^2
\]

\[
+ 2\tau_u \ell_t^2 - (\ell_t^2)^2 / \beta
\]

\[
= \sum_{t=1}^{T} (2\tau_u \ell_t - \tau_u^2 (\| \mathbf{v}_t \|_2^2 + \beta^2) - (\ell_t^2)^2 / \beta).
\]

Setting \( \beta = \sqrt{\lambda / 2C} \) and using \( \tau_u = \ell_t / (\| \mathbf{v}_t \|_2^2 + \lambda \ell_t^2 / 2C) \), (9) gives

\[
\| \mathbf{u}^* - \mathbf{u}_1 \|_2^2
\]

\[
\geq \sum_{t=1}^{T} \left( 2\tau_u \ell_t - \tau_u^2 \left( \| \mathbf{v}_t \|_2^2 + \lambda \ell_t^2 / 2C \right) - \frac{2C}{\lambda} (\ell_t^2)^2 \right).
\]

Using the assumption that \( \| \mathbf{v}_t \|_2^2 \leq R \) to all \( t \) and rearranging terms concludes the proof. \( \Box \)

The result given in Proposition 3 bounds the cumulative loss of the learning procedure when one of the parameters, either \( \Phi \) or \( \omega \), is fixed and the other is the optimization variable. Therefore, it does not directly capture the behavior of Algorithm 1, which alternates between the updates of \( \Phi \) and \( \omega \). A proper analysis of Algorithm 1 would require a refinement of Lemma 1 which, to our understanding, would be the core of a new result.

This is a problem we intend to put our energy on in the near future, as an extension to this work.

4 Experiments on Synthetic Data

In order to better understand and validate the RPA algorithm, we conducted some synthetic experiments. First, we simulated a high-dimensional matrix \( \mathbf{X} \in \mathbb{R}^{n \times p} \), with \( n \) data samples realizing \( p \) “words”, with \( p \gg n \), and where the first (resp. last) \( n/2 \) samples each realize one of the first (resp. last) \( p/2 \) words and are labeled +1 (resp. −1). That is, the rows of \( \mathbf{X} \) are \( p \)-dimensional indicator vectors over words. Second, we also assume a hidden matrix \( \Phi \in \mathbb{R}^{2 \times p} \)

Figure 1: Accuracy rates for PA and RPA as a function of the variance of the Gaussian noise added to the “true” embedding \( \Phi \). The observed \( X \) matrix is \( n = 500 \times p = 1000 \).
which relates the \( p \) words to two underlying “concepts”: specifically, the first (resp. second) concept is mapped to the first (resp. last) \( p/2 \) words. That is, the columns of \( \Phi \) are \( 2 \)-dimensional indicator vectors over concepts. As learning from both \( X \) and \( \Phi \) would be trivial, we further assume that \( \Phi \) is only accessible through a noisy embedding matrix \( \tilde{\Phi} \in \mathbb{R}^{2 \times p} \), such that \( \tilde{\Phi} = \Phi + E \), where \( E \in \mathbb{R}^{2 \times p} \) and \( e_i \sim \mathcal{N}(0, \sigma^2 I_p) \).

Given this setting, the goal of the RPA is to learn a set of classification parameters \( w \) in the latent \( 2 \)-dimensional space and to “de-noise” the observed embedding matrix \( \tilde{\Phi} \) by exploiting the labeled data. We are interested in comparing the RPA with a regular PA that directly learns from the noisy data \( X \tilde{\Phi}^\top \). The outcome of this comparison is plotted in Figure 1. For this experiment, we randomly splitted the \( X \) data according to 80/10/10 for train/dev/test and considered increasing noise variance from 0 to 10 by increment of 0.1. Each dot in Figure 1 corresponds to the average accuracy over 10 separate, random initializations of the embedding matrix at a particular noise level. Hyper-parameters were optimized using a grid search on the dev set for both the PA and RPA.¹

As shown in Figure 1, the PA’s accuracy quickly drops to levels that are only slightly above chance, while the RPA manages to maintain an accuracy close to 0.7 even with large noise. This indicates that the RPA is able to recover some of the structure in the embedding matrix. This behavior is also illustrated in Figure 2, wherein the two hidden concepts appear in yellow and green. While the standard PA learns a very bad hyper-plane, which fails to separate the two concepts, the RPA learns a much better hyper-plane. Interestingly, most of the data points appear to have been projected on the margins of the hyper-plane.

5 Experiments on NLP tasks

This section assesses the effectiveness of RPA on several text classification tasks.

5.1 Evaluation Datasets

We consider five different classification tasks which are concisely summarized in Table 1.

20 Newsgroups  Our first three text classification tasks from this dataset² consists in categorizing documents into two related sub-topics: (i) Comp.: IBM vs. Mac, (ii) Religion: atheism vs. christian, and (iii) Sports: baseball vs. hockey.

IMDB Movie Review  This movie dataset³ was introduced by (Maas et al., 2011) for sentiment analysis, and contains 50,000 reviews, divided into a balanced set of highly positive (7 stars out of 10 or more) and negative scores (4 stars or less).

TREC Question Classification  This dataset⁴ (Li and Roth, 2002) involves six question types: abbreviation, description, entity, human, location, and number.

¹ We use \{1, 5, 10, 50\} for the number iterations, \( C \)'s values were \{1.1, 5.1, 10.0, 2.0, 10.0\}, and \( \lambda \) was set to \( 10^k \), with \( k \in \{-2, -1, 0, 1, 2\} \).

²qwone.com/~jason/20Newsgroups
³ai.stanford.edu/~amaas/data/sentiment
⁴cogcomp.cs.illinois.edu/Data/QA/QC
Table 1: Number of labels, samples, vocabulary sizes (incl. non-hapax words) per dataset.

<table>
<thead>
<tr>
<th>Name</th>
<th>lab.</th>
<th>examples</th>
<th>Voc. size</th>
<th>Word Embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>comp</td>
<td>2</td>
<td>1 168 777</td>
<td>30 292 15 768</td>
<td>CnW 40.41</td>
</tr>
<tr>
<td>rel.</td>
<td>2</td>
<td>1 079 717</td>
<td>32 361 18 594</td>
<td>GV6 25.45</td>
</tr>
<tr>
<td>sports</td>
<td>2</td>
<td>1 197 796</td>
<td>33 932 19 812</td>
<td>GV840 35.29</td>
</tr>
<tr>
<td>IMDB</td>
<td>2</td>
<td>255 25k</td>
<td>172 001 86 361</td>
<td>HPCA 39.79</td>
</tr>
<tr>
<td>TREC</td>
<td>6</td>
<td>5 452 500</td>
<td>9 775 3 771</td>
<td>HLBL 3.42</td>
</tr>
</tbody>
</table>

5.2 Preprocessing and Document Vectors

All datasets were pre-processed with the Stanford tokenizer\(^5\), except for the TREC corpus which comes pre-tokenized. Case was left intact unless used in conjunction with word embeddings that assume down-casing (see below).

For constructing document or sentence vectors, we used a simple bag-of-word model, simply summing over the word vectors of occurring tokens, followed by L2-normalization of the resulting vector in order to avoid document/sentence length effects. For each dataset, we restricted the vocabulary to non-hapax words (i.e., words occurring more than once). Words unknown to the embedding were mapped to zero vectors.

5.3 Initial Word Vector Representations

Five publicly available word vectors were used to define initial embedding matrices in the RPA. The coverage of the different embeddings wrt each dataset vocabulary is reported in Table 2.

<table>
<thead>
<tr>
<th>Name</th>
<th>examples</th>
<th>Voc. size</th>
<th>Word Embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>CnW</td>
<td>2</td>
<td>1 168 777</td>
<td>30 292 15 768</td>
</tr>
<tr>
<td>GV6</td>
<td>2</td>
<td>1 079 717</td>
<td>32 361 18 594</td>
</tr>
<tr>
<td>GV840</td>
<td>2</td>
<td>1 197 796</td>
<td>33 932 19 812</td>
</tr>
<tr>
<td>HPCA</td>
<td>2</td>
<td>255 25k</td>
<td>172 001 86 361</td>
</tr>
<tr>
<td>HLBL</td>
<td>2</td>
<td>5 452 500</td>
<td>9 775 3 771</td>
</tr>
</tbody>
</table>

Table 2: Out-of-vocabulary rates for non-hapax words in each dataset-embedding pair.

5.4 Settings

The hyperparameters of the model, \(C\), \(\lambda\), and the number \(it\) of iterations over the training set, were estimated using a grid search over \(C = 10^i \in \{-6,-4,-2,0,2,4,6\}\), \(\lambda = 10^j \in \{-3,-2,-1,0,1,2,3\}\), and \(it \in \{1,5,10\}\) in a 10-fold cross-validation.

\(^5\)nlp.stanford.edu/software/tokenizer.shtml

\(^6\)http://lebret.ch/words

\(^7\)nlp.stanford.edu/projects/glove

\(^8\)code.google.com/archive/p/word2vec
over the training data. For the alternating online procedure, we used the difference between
the objective values from one iteration to the next for defining the stopping criterion, with maximum number of iterations of 50. In practice, we found that the search often converged much fewer iterations. The multi-class classifier used for the TREC dataset was obtained by training the RPA in simple One-versus-All fashion, thus learning one embedding matrix per class.\(^\text{10}\) For datasets with label imbalance, we set a different \(C\) parameter for each class, re-weighting it in proportion to the inverse frequency of the class.

5.5 Results

Table 3 summarizes accuracy results for the RPA against those obtained by a PA trained with fixed pre-trained embeddings. The first thing to notice is that the RPA delivers massive accuracy improvements over the vanilla PA across datasets and embedding types and sizes, thus showing that the RPA is able to learn word representations that are better tailored to each problem. On average, accuracy gains are between 22% and 31% for CnW, HLBL, and HPCA. Sizable improvements, ranging from 8% and 18%, are also found for the better performing GV6B, GV840B, and SkGr. Second, RPA is able to outperform on all five datasets the strong baseline provided by the one-hot version of PA trained on the original high-dimensional space, with some substantial gains on \(\text{sports}\) and \(\text{trec}\).

Overall, the best scores are obtained with the re-embedded SkGr vectors, which yield the best average accuracy, and outperform all the other configurations on two of the five datasets (\(\text{trec}\) and \(\text{imdb}\)). GV6B (dimension 300) has the second best average scores, outperforming all the other configurations on \(\text{sports}\). Interestingly, embeddings learned from random vectors achieve performance that are often on a par or higher than those given by HLBL, HPCA or CnW initializations. They actually yield the best performance for the two remaining datasets: \(\text{comp}\) and \(\text{religion}\). On these tasks, RPA does not seem to benefit from the information contained in the pre-trained embeddings, or their coverage is not just good enough.

For both PA and RPA, performance appear to be positively correlated with embedding coverage: embeddings with lower OOV rates generally perform better those with more missing words. The correlation is only partial, since GV840B do not yield gains compared to GV6B and SkGr despite its better word coverage. Also, SkGr largely outperforms HPCA although they have similar OOV

<table>
<thead>
<tr>
<th>Emb/Task Method</th>
<th>Size</th>
<th>comp</th>
<th>religion</th>
<th>sports</th>
<th>trec</th>
<th>imdb</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>rand</td>
<td>50</td>
<td>54.57</td>
<td>87.13</td>
<td>60.67</td>
<td>91.91</td>
<td>63.07</td>
<td>92.34</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>62.03</td>
<td>87.39</td>
<td>66.95</td>
<td>92.19</td>
<td>65.58</td>
<td>93.22</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>65.51</td>
<td>87.64</td>
<td>73.22</td>
<td>92.19</td>
<td>71.11</td>
<td>92.96</td>
</tr>
<tr>
<td>CnW</td>
<td>50</td>
<td>50.32</td>
<td>85.97</td>
<td>55.09</td>
<td>91.91</td>
<td>56.91</td>
<td>93.72</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>49.29</td>
<td>87.13</td>
<td>47.56</td>
<td>91.63</td>
<td>58.54</td>
<td>93.72</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>50.32</td>
<td>87.00</td>
<td>53.14</td>
<td>91.77</td>
<td>63.94</td>
<td>93.72</td>
</tr>
<tr>
<td>HLBL</td>
<td>50</td>
<td>51.99</td>
<td>87.13</td>
<td>68.62</td>
<td>91.35</td>
<td>61.31</td>
<td>93.72</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>53.54</td>
<td>86.74</td>
<td>65.83</td>
<td>91.77</td>
<td>63.07</td>
<td>93.72</td>
</tr>
<tr>
<td>HPCA</td>
<td>50</td>
<td>50.45</td>
<td>86.87</td>
<td>50.77</td>
<td>91.63</td>
<td>64.20</td>
<td>93.34</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>50.45</td>
<td>86.87</td>
<td>47.98</td>
<td>91.63</td>
<td>64.57</td>
<td>93.72</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>50.45</td>
<td>86.87</td>
<td>48.12</td>
<td>91.91</td>
<td>62.56</td>
<td>93.59</td>
</tr>
<tr>
<td>GV6B</td>
<td>50</td>
<td>55.21</td>
<td>86.87</td>
<td>75.59</td>
<td>91.91</td>
<td>86.68</td>
<td>95.23</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>58.17</td>
<td>86.87</td>
<td>76.43</td>
<td>91.63</td>
<td>90.33</td>
<td>96.48</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>70.40</td>
<td>86.87</td>
<td>73.22</td>
<td>91.63</td>
<td>93.34</td>
<td>97.11</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>76.58</td>
<td>87.13</td>
<td>79.92</td>
<td>91.77</td>
<td>94.97</td>
<td>97.74</td>
</tr>
<tr>
<td>GV840B</td>
<td>300</td>
<td>75.80</td>
<td>87.13</td>
<td>88.15</td>
<td>92.05</td>
<td>88.69</td>
<td>96.23</td>
</tr>
<tr>
<td>SkGr</td>
<td>300</td>
<td>70.79</td>
<td>87.39</td>
<td>88.01</td>
<td>92.05</td>
<td>91.96</td>
<td>97.61</td>
</tr>
<tr>
<td>one-hot</td>
<td></td>
<td>87.26</td>
<td>91.91</td>
<td>93.47</td>
<td>88.00</td>
<td>88.29</td>
<td>89.768</td>
</tr>
</tbody>
</table>

Table 3: Accuracy results on our five datasets for the Re-embedding Passive-Aggressive (RPA) against standard PA-II (PA). For each task, the best results for each embedding method (across different dimensions) has been greyed out, while the overall highest accuracy score has been underlined.

---

\(^{10}\)More interesting configurations (e.g., a single embedding matrix shared across classes), are left for future work.
In this paper, we have proposed a new scalable algorithm for learning word representations that are specifically tailored to a classification objective. This algorithm generalizes the well-known Passive-Aggressive algorithm, and we showed how to extend the regret bounds results of the PA to the RPA when either the weight vector or the embedding matrix is fixed. In addition, we have also provided synthetic and NLP experiments, demonstrating that the good classification performance of RPA.

In future work, we first would like to achieve a more complete analysis of the RPA algorithm when both \( \Phi \) and \( \Phi \) both get updated. Also, we intend to investigate potential exact methods for solving biconvex minimization (Floudas and Viswewaran, 1990), as well as to develop a stochastic version of RPA, thus foregoing running the inner alternate search to convergence. More empirical perspectives include extending the RPA to linguistic structured prediction tasks, better handling of unknown words, and a deeper intrinsic and statistical evaluation of the embeddings learned by the RPA.

**Acknowledgments**

We thank the three anonymous reviewers for their comments. Pascal Denis was supported by ANR Grant GRASP No. ANR-16-CE33-0011-01, as well as by a grant from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020.

<table>
<thead>
<tr>
<th>Emb/Task</th>
<th>comp</th>
<th>religion</th>
<th>sports</th>
<th>trec</th>
<th>imdb</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method Size</td>
<td>PA</td>
<td>RPA</td>
<td>PA</td>
<td>RPA</td>
<td>PA</td>
<td>RPA</td>
</tr>
<tr>
<td>rand 50</td>
<td>56.76</td>
<td>84.04</td>
<td>59.55</td>
<td>90.24</td>
<td>64.20</td>
<td>90.08</td>
</tr>
<tr>
<td></td>
<td>63.06</td>
<td>84.17</td>
<td>67.50</td>
<td>91.35</td>
<td>65.58</td>
<td>90.95</td>
</tr>
<tr>
<td></td>
<td>64.86</td>
<td>85.07</td>
<td>71.27</td>
<td>91.07</td>
<td>68.59</td>
<td>90.95</td>
</tr>
<tr>
<td>cnw 50</td>
<td>50.32</td>
<td>84.81</td>
<td>55.51</td>
<td>90.93</td>
<td>50.63</td>
<td>91.08</td>
</tr>
<tr>
<td></td>
<td>50.71</td>
<td>84.81</td>
<td>55.51</td>
<td>91.35</td>
<td>54.15</td>
<td>91.08</td>
</tr>
<tr>
<td></td>
<td>50.45</td>
<td>84.04</td>
<td>55.51</td>
<td>90.93</td>
<td>54.77</td>
<td>91.08</td>
</tr>
<tr>
<td>HLBL 50</td>
<td>53.15</td>
<td>85.07</td>
<td>59.97</td>
<td>89.82</td>
<td>56.41</td>
<td>90.95</td>
</tr>
<tr>
<td></td>
<td>53.80</td>
<td>84.68</td>
<td>61.09</td>
<td>91.21</td>
<td>56.91</td>
<td>90.95</td>
</tr>
<tr>
<td>HPCA 50</td>
<td>50.45</td>
<td>85.20</td>
<td>55.51</td>
<td>91.35</td>
<td>52.39</td>
<td>89.95</td>
</tr>
<tr>
<td></td>
<td>50.45</td>
<td>85.20</td>
<td>55.51</td>
<td>91.07</td>
<td>49.87</td>
<td>90.08</td>
</tr>
<tr>
<td></td>
<td>50.45</td>
<td>85.20</td>
<td>55.51</td>
<td>90.10</td>
<td>49.87</td>
<td>89.82</td>
</tr>
<tr>
<td>GV6B 50</td>
<td>50.97</td>
<td>85.07</td>
<td>64.16</td>
<td>91.42</td>
<td>55.28</td>
<td>91.08</td>
</tr>
<tr>
<td></td>
<td>50.58</td>
<td>84.94</td>
<td>60.53</td>
<td>89.68</td>
<td>63.82</td>
<td>91.08</td>
</tr>
<tr>
<td></td>
<td>51.22</td>
<td>85.20</td>
<td>64.99</td>
<td>91.49</td>
<td>85.05</td>
<td>90.08</td>
</tr>
<tr>
<td></td>
<td>56.24</td>
<td>85.33</td>
<td>70.15</td>
<td>89.68</td>
<td>89.07</td>
<td>91.21</td>
</tr>
<tr>
<td>GV840B 300</td>
<td>66.02</td>
<td>84.81</td>
<td>77.96</td>
<td>89.68</td>
<td>89.82</td>
<td>91.08</td>
</tr>
<tr>
<td>SkGr 300</td>
<td>67.95</td>
<td>82.50</td>
<td>81.59</td>
<td>89.40</td>
<td>95.10</td>
<td>94.60</td>
</tr>
<tr>
<td>one-hot</td>
<td>84.56</td>
<td>89.96</td>
<td>90.58</td>
<td>85.80</td>
<td>87.40</td>
<td>88.76</td>
</tr>
</tbody>
</table>

Table 4: Accuracy results for RPA against standard PA both run for a single iteration.
References


