An Iterative Algorithm to Learn Sparse Dual Vectors Using Neural Networks

Amin Karbasi
amin.karbasi@epfl.ch

Amir Hesam Salavati
hesam.salavati@epfl.ch

Amin Shokrollahi
amin.shokrollahi@epfl.ch

Algorithmics Laboratory (ALGO)
Ecole Polytechnique Federale de Lausanne (EPFL)

1 Summary

In this document, we propose an iterative algorithm to learn a set of *sparse* vectors orthogonal to a subspace by merely accessing at the vectors belonging to that subspace. The sparsity of dual vectors refers to the fact that we favor those dual vectors that have a relatively large number of zero entries. The algorithm is designed such that it is simple enough to be implemented by a neural network. We prove the convergence of the algorithm by showing that the final result is in fact orthogonal to the given patterns in the subspace. We also derive the conditions for the algorithm not to converge to the trivial all-zero solution.

2 Problem Statement

We assume to be given C vectors of length n with integer-valued entries belonging to $\mathcal{S}$. Furthermore, we assume these patterns belong to a subspace with dimension $k \leq n$. Note that if $k = n$. Our goal is to memorize these patterns by finding a set of non-zero vectors $w_1, \ldots, w_m \in \mathbb{R}^n$ that are orthogonal to the set of given patterns. Furthermore, we are interested in rather sparse vectors. Denoting pattern μ by x^μ, we can formulate the problem of finding *one* such vector w as:

$$\min \sum_{\mu=1}^{C} |x^\mu \cdot w|^2 \tag{1a}$$

subject to

$$\|w\|_0 \leq q \tag{1b}$$

$$\|w\|_2^2 \geq \epsilon \tag{1c}$$

where $\cdot$ represents the inner-product, $q \in \mathbb{N}$ determines the degree of sparsity and $\epsilon \in \mathbb{R}^+$ prevents converging to the trivial all-zero solution.

By repeating the above problem for different vectors w, we will have $m = n - k$ sparse vectors that form the basis for the null space of the patterns which would like to learn.

3 The Learning Algorithm

Since the patterns are assumed to be coming from a subspace in the n-dimensional space, we adapt the algorithm proposed by Oja and Karhunen

[4] to learn the null-space basis of the subspace defined by the patterns. A very similar algorithm is also used in [6] for the same purpose. There are two main differences between the algorithm proposed here and that of [6]: We focus on finding dual vectors that are rather *sparse*, and we do not require the set of dual vectors $w_1, \dots, w_m$ be orthogonal. This last property gives us the additional advantage to make the algorithm *parallel* and *adaptive*. Parallel in the sense that we can design an algorithm to learn one constraint and repeat it several times in order to find all of the constraints with high probability. And adaptive in the sense that we can determine the number of constraints on-the-go, i.e. start by learning just a few constraints. If needed (for instance due to bad performance in the recall phase), the network can easily learn additional constraints. This increases the flexibility of the algorithm and provides a nice trade-off between the time spent on learning and the performance in the recall phase.

3.1 Overview of the proposed algorithm

In order to develop a simple iterative algorithm, we formulate the problem in an optimization framework. The problem to find a constraint *vector* w is given by equation (1). However, instead of tackling problem (1) directly, we make a slight modification and consider the following optimization problem:

$$\min \sum_{\mu=1}^{C} |x^\mu \cdot w|^2 + \lambda g(w). \tag{2a}$$

subject to:

$$\|w\|_2 = 1 \tag{2b}$$

In the above problem, we have replaced the constraint $\|w\|_0 \leq q$ with a penalty function $g(w)$ since $\|.\|_0$ is not easy to handle analytically. The function $g(w)$ is chosen such that it favors sparsity. For instance one can pick $g(w)$ to be $\|.\|_1$, which leads to ℓ_1-norm penalty and is widely used in compressed sensing applications [3], [5]. In this paper, we consider the function

$$g(w) = \sum_{i=1}^{n} \tanh(\sigma w_i^2)$$

where σ is chosen appropriately. Intuitively, $\tanh(\sigma w_i^2)$ approximates $|\text{sign}(w_i)|$ in ℓ_0-norm. Therefore, the larger σ is, the closer $g(w)$ will be to $\|.\|_0$. By calculating the derivative of the objective function, and by considering the update due to each randomly picked pattern x^μ, we will get the following iterative algorithm:

$$y(t) = x(t) \cdot w(t) \tag{3a}$$

$$\tilde{w}(t+1) = w(t) - \alpha_t \left(2y(t)x(t) + \lambda\Gamma(w(t))\right) \tag{3b}$$

$$w(t+1) = \frac{\tilde{w}(t+1)}{\|\tilde{w}(t+1)\|_2} \tag{3c}$$

In the above equations, t is the iteration number, $x(t)$ is the sample pattern chosen at iteration t uniformly at random from the patterns in the training set $\mathcal{X}$, and α_t is a small positive constant. Finally, $\Gamma(w) : \mathcal{R}^n \to \mathcal{R}^n = \nabla g(w)$ is the gradient of the penalty term for non-sparse solutions. This function has the interesting property that for very small values of $w_i(t)$. To see why, consider the i^{th} entry of the function $\Gamma(w(t)))$

$$\Gamma_i(w(t)) = \partial g(w(t))/\partial w_i(t) = 2\sigma_t w_i(t)(1 - \tanh^2(\sigma w_i(t)^2))$$

It is easy to see that $\Gamma_i(w(t)) \simeq 2\sigma w_i(t)$ for relatively small $w_i(t)$'s. And for larger values of $w_i(t)$, we get $\Gamma_i(w(t)) \simeq 0$ (see Figure 1). Therefore, by proper choice of λ and σ, equation (3b) suppresses small entries of $w(t)$ by pushing them towards zero, thus, favoring sparser results.

Following the same approach as [4] and assuming α_t to be small enough such that equation (3c) can be expanded as powers of α_t, we can approximate equation (3) with the following simpler version:

$$y(t) = x(t) \cdot w(t) \tag{4a}$$

$$w(t+1) = w(t) - \alpha_t \left(y(t) \left(x(t) - \frac{y(t)w(t)}{\|w(t)\|_2^2} \right) + \lambda\Gamma(w(t)) \right) \tag{4b}$$

In the above approximation, we also omitted the term $\alpha_t\lambda\left(w(t) \cdot \Gamma(w(t))\right) w(t)$ since $w(t) \cdot \Gamma(w(t))$ would be negligible for large σ's. In fact it tends to zero as σ grows.

The overall learning algorithm for one constraint node is given by Algorithm 1. In words, in Algorithm 1 $y(t)$ is the projection of $x(t)$ on the basis vector $w(t)$. If for a given data vector $x(t)$, $y(t)$ is equal to zero, namely, the data is orthogonal to the current weight vector $w(t)$, then according to equation (4b) the weight vector will not be updated. However, if the data vector $x(t)$ has some projection over $w(t)$ then the weight vector is updated towards the direction to reduce this projection.

Since we are interested in finding m basis vectors, we have to do the above procedure m times in parallel.

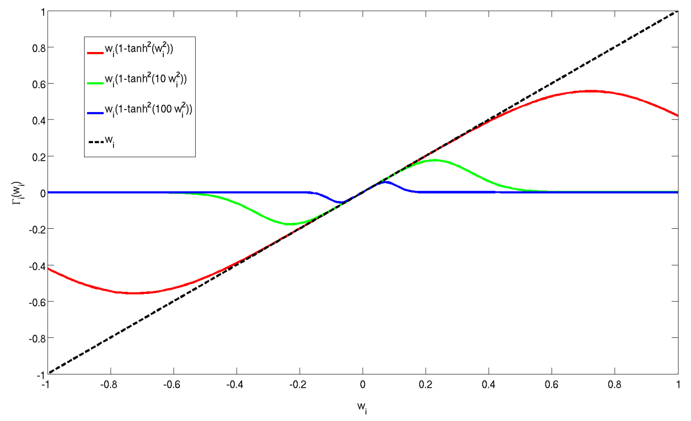

Figure 1: The sparsity penalty $\Gamma_i(w_i)$, which suppresses small values of the i^{th} entry of w in each iteration as a function of w_i and σ. Note that the normalization constant 2σ has been omitted here to make comparison with function $f = w_i$ possible.

Remark 1. *Although we are interested in finding a sparse graph, note that too much sparseness is not desired. Because we are going to use the feedback sent by the constraint nodes to eliminate input noise at pattern nodes. Now if the graph is too sparse, the number of feedbacks received by each pattern node is too small to be relied upon. Therefore, we must adjust the penalty coefficient λ such that resulting neural graph is* rather *sparse. In the section on experimental results, we compare the error correction performance for different choices of λ.*

3.2 Convergence analysis

In order to prove that Algorithm 1 converges to the proper solution, we use results from statistical learning. More specifically, we benefit from the convergence of Stochastic Gradient Descent (SGD) algorithms [1]. To prove the convergence, let $E(w) = \sum_{\mu} |x^{\mu} \cdot w|^2$ be the cost function we would like to minimize. Furthermore, let $A = \mathbb{E}\{xx^T | x \in \mathcal{X}\}$ be the corelation patterns for the patterns in the training set. Therefore, due to uniformity assumption for the patterns in the training set, one can rewrite $E(w) = w^T A w$, where

```
Algorithm 1 Iterative Learning
Input: Set of patterns $x^\mu \in \mathcal{X}$ with $\mu = 1, \dots, C$, stopping point $\varepsilon$.
Output: $w$
  while $\sum_\mu |x^\mu \cdot w(t)|^2 > \varepsilon$ do
    Choose $x(t)$ at random from patterns in $\mathcal{X}$
    Compute $y(t) = x(t) \cdot w(t)$
    Update $w(t+1) = w(t) - \alpha_t y(t) \left( x(t) - \frac{y(t)w(t)}{\|w(t)\|_2^2} \right) - \alpha_t \lambda \Gamma(w(t))$.
    $t \leftarrow t + 1$.
  end while
```

we have omitted the normalizing $1/C$ constant for simplicity. finally, denote $A_\mu = x^\mu (x^\mu)^T$. Now consider the following assumptions:

A1. $\|A\|_2 \leq \Upsilon < \infty$ and $\sup_\mu \|A_\mu\|_2 = \|x^\mu\|^2 \leq \zeta < \infty$.

A2. $\alpha_t \geq 0$, $\sum \alpha_t = \infty$ and $\sum \alpha_t^2 < \infty$.

The following lemma porves the convergence of Algorithm 1 to a local minimum w^*.

Lemma 1. *Let assumptions A1 and A2 hold. Then, Algorithm 1 converges to a local minimum w^* for which $\nabla E(w^*) = 0$.*

Proof. To prove the lemma, we use the convergence results in [1] and show that the required assumptions to ensure convergence holds for the proposed algorithm. For simplicity, these assumptions are listed here:

1. The cost function $E(w)$ is three-times differentiable with continuous derivatives. It is also bounded from below.

2. The usual conditions on the learning rates are fulfilled, i.e. $\sum \alpha_t = \infty$ and $\sum \alpha_t^2 < \infty$.

3. The second moment of the update term should not grow more than linearly with size of the weight vector. In other words,

$$E(w) \leq A + B\|w\|_2^2$$

for some constants A and B.

4. When the norm of the weight vector w is larger than a certain horizon D, the opposite of the gradient $-\nabla E(W)$ points towards the origin. Or in other words:

$$\inf \|w\|_2 > Dw \cdot \nabla E(w) > 0$$

5. When the norm of the weight vector is smaller than a second horizon F, with $F > D$, then the norm of the update term $(2y(t)x(t) + \lambda\Gamma(w(t)))$ is bounded regardless of $x(t)$. This is usually a mild requirement:

$$\forall x(t) \in \mathcal{X}, \sup_{\|w\|_2 \leq F} \| (2y(t)x(t) + \lambda\Gamma(w(t))) \|_2 \leq K_0$$

To start, assumption 1 holds trivially as the cost function is three-times differentiable, with continuous derivatives. Furthermore, $E(w) \geq 0$. Assumption 2 holds because of our choice of the step size α_t, as mentioned in the lemma description.

Assumption 3 ensures that the vector w could not escape by becoming larger and larger. Due to the constraint $\|w\|_2 = 1$, this assumption holds as well.

Assumption 4 holds as well because:

$$\begin{aligned} \mathbb{E}_\mu \left(2A_\mu w + \lambda\Gamma(w)\right)^2 &= 4w^T\mathbb{E}_\mu(A_\mu^2)w + \lambda^2\|\Gamma(w)\|_2^2 \\ &+ 4\lambda w^T\mathbb{E}_\mu(A_\mu)\Gamma(w) \\ &\leq 4\|w\|_2^2\zeta^2 + \lambda^2\|w\|_2^2 + 4\lambda\Upsilon\|w\|_2^2 \\ &= \|w\|_2^2(4\zeta^2 + 4\lambda\Upsilon + \lambda^2) \end{aligned} \tag{5}$$

Finally, assumption 5 holds because:

$$\begin{aligned} \|2A_\mu w + \lambda\Gamma(w)\|_2^2 &= 4w^T A_\mu^2 w + \lambda^2\|\Gamma(w)\|_2^2 \\ &+ 4\lambda w^T A_\mu\Gamma(w) \\ &\leq \|w\|_2^2(4\zeta^2 + 4\lambda\zeta + \lambda^2) \end{aligned} \tag{6}$$

Therefore, $\exists F > D$ such that as long as $\|w\|_2^2 < F$:

$$\sup_{\|w\|_2^2 < E} \|2A_\mu w + \lambda\Gamma(w)\|_2^2 \leq (2\zeta + \lambda)^2 F = \text{constant} \tag{7}$$

Since all necessary assumptions hold for the learning algorithm 1, it converges to a local minimum where $\nabla E(w^*) = 0$. □

Next, we prove the desired result, i.e. the fact that in the local minimum, the resulting weight vector is orthogonal to the patterns, i.e. $Aw = 0$.

Theorem 2. *In the local minimum where $\nabla E(w^*) = 0$, the optimal vector w^* is orthogonal to the patterns in the training set, i.e. $Aw^* = 0$.*

Proof. Since $\nabla E(w^*) = 2Aw^* + \lambda\Gamma(w^*) = 0$, we have:

$$w^* \cdot \nabla E(w^*) = 2(w^*)^T Aw^* + \lambda w^* \cdot \Gamma(w^*) \tag{8}$$

The first term is always greater than or equal to zero. Now as for the second term, we have that $|\Gamma(w_i)| \leq |w_i|$ and $\text{sign}(w_i) = \text{sign}(\Gamma(w_i))$, where w_i is the i^{th} entry of w. Therefore, $0 \leq w^* \cdot \Gamma(w^*) \leq \|w^*\|_2^2$. Therefore, both terms on the right hand side of (8) are greater than or equal to zero. And since the left hand side is known to be equal to zero, we conclude that $(w^*)^T Aw^* = 0$ and $\Gamma(w^*) = 0$. The former means $(w^*)^T Aw^* = \sum_\mu (w^* \cdot x^\mu)^2 = 0$. Therefore, we must have $w^* \cdot x^\mu = 0$, for all $\mu = 1, \ldots, C$. Which simply means the vector w^* is orthogonal to all the patterns in the training set. □

Remark 2. *Note that the above theorem only proves that the obtained vector is orthogonal to the data set and says nothing about its degree of sparsity. The reason is that there is no guarantee that the dual basis of a subspace be sparse. The introduction of the penalty function $g(w)$ in problem (4a) only encourages sparsity by suppressing the small entries of w, i.e. shifting them towards zero if they are really small or leaving them intact if they are rather large. And from the fact that $\Gamma(w^*) = 0$, we know this is true as the entries in w^* are either large or zero, i.e. there are no small entries.*

3.3 Avoiding the all-zero solution

Although in problem (2) we have the constraint $\|w\|_2 = 1$ to make sure that the algorithm does not converge to the trivial solution $w = 0$, due to approximations we made when developing the optimization algorithm, we should make sure to choose the parameters such that the all-zero solution is still avoided.

To this end, denote $w'(t) = w(t) - \alpha_t y(t)\left(x(t) - \frac{y(t)w(t)}{\|w(t)\|_2^2}\right)$ and consider the following inequalities:

$$\begin{aligned} \|w(t+1)\|_2^2 &= \|w(t) - \alpha_t y(t)\left(x(t) - \frac{y(t)w(t)}{\|w(t)\|_2^2}\right) - \alpha_t\lambda\Gamma(w(t))\|_2^2 \\ &= \|w'(t)\|^2 + \alpha_t^2\lambda^2\|\Gamma(w(t))\|^2 - 2\alpha_t\lambda\Gamma(w(t)) \cdot w'(t) \\ &\geq \|w'(t)\|_2^2 - 2\alpha_t\lambda\Gamma(w(t)) \cdot w'(t) \end{aligned} \tag{9}$$

Now in order to have $\|w(t+1)\|_2^2 > 0$, we must have that $2\alpha_t\lambda|\Gamma(w(t))^T w'(t)| \leq \|w'(t)\|_2^2$. Given that, $|\Gamma(w(t)) \cdot w'(t)| \leq \|w'(t)\|_2\|\Gamma(w(t))\|_2$, it is therefore sufficient to have $2\alpha_t\lambda\|\Gamma(w(t))\|_2 \leq \|w'(t)\|_2$. On the other hand, we have:

$$\begin{aligned} \|w'(t)\|_2^2 &= \|w(t)\|_2^2 + \alpha_t^2 y(t)^2 \|x(t) - \frac{y(t)w(t)}{\|w(t)\|_2^2}\|_2^2 \\ &\geq \|w(t)\|_2^2 \end{aligned} \tag{10}$$

As a result, in order to have $\|w(t+1)\|_2^2 > 0$, it is sufficient to have $2\alpha_t\lambda\|\Gamma(w(t))\|_2 \leq \|w(t)\|_2$. Finally, since we have $|\Gamma(w(t))| \leq |w(t)|$ (entry-wise), we know that $\|\Gamma(w(t))\|_2 \leq \|w(t)\|_2$. Therefore, having $2\alpha_t\lambda < 1 \leq \|w(t)\|_2/\|\Gamma(w(t))\|_2$ ensures $\|w(t)\|_2 > 0$.

Remark 3. *Interestingly, the above choice for the function $w - \lambda\Gamma(w)$ looks very similar to the soft thresholding function (11) introduced in [3] to perform iterative compressed sensing. The authors show that their choice of the sparsity function is very competitive in the sense that one can not get much better results by choosing other thresholding functions. However, one main difference between their work and that of ours is that we enforce the sparsity as a penalty in equation (3b) while they apply the soft thresholding function in equation (11) to the whole w, i.e. if the updated value of w is larger than a threshold, it is left intact while it will be put to zero otherwise.*

$$\eta_t(x) = \begin{cases} x - \theta_t & \text{if } x > \theta_t; \\ x + \theta_t & \text{if } x < -\theta_t \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

where θ_t is the threshold at iteration t and tends to zero as t grows.

References

[1] L. Bottou, *Online algorithms and stochastic approximations*, in David Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998.

[2] E. Cands, T. Tao, *Near optimal signal recovery from random projections: Universal encoding strategies?*, IEEE Trans. on Information Theory, Vol. 52, No. 12, 2006, pp. 5406 - 5425.

[3] D. L. Donoho, A. Maleki, A. Montanari, *Message passing algorithms for compressed sensing*, Proc. Nat. Acad. Sci., Vol. 106, 2009, pp. 1891418919.

[4] E. Oja, J. Karhunen, *On stochastic approximation of eigenvectors and eigenvalues of the expectation of a random matrix*, J. Math. Analysis and Applications, Vol. 106, No. 1, 1985, pp 6984.

[5] J. Tropp J, S. J. Wright, *Computational methods for sparse solution of linear inverse problems*, Proc. IEEE, Vol. 98, No. 6, 2010, pp. 948-958.

[6] L. Xu, A. Krzyzak, E. Oja, Neural nets for dual subspace pattern recognition method, Int. J. Neur. Syst., Vol. 2, No. 3, 1991, pp. 169-184.