Sparse Coding of Point Cloud Data

CS229 Project Report
Alex Teichman
alex.teichman@cs.stanford.edu

***Abstract*—Point clouds, made available through laser range finders, stereo cameras, or time of flight cameras, are frequently used in robot navigation systems. However, no unsupervised machine perception algorithm exists to provide *understanding* of the data; e.g. that a particular blob of points looks roughly like, say, a car. In this paper, we take steps towards such an algorithm based on sparse coding. The work here generalizes to any binary data.**

An algorithm for learning the bases and the activations for point cloud data is derived and demonstrated. Given precomputed basis vectors and an input vector, calculating the activations is very fast and could be used in real-time applications.

I. Background

Sparse coding is known to be used in the abstract representation of input in biological sensory systems [2]. An efficient method of determining the bases and activations of a sparse coding representation of image data has been specified already [3], but the initial assumptions that are made need to be changed for use in the point cloud regime.

There are two main reasons for doing this work. First, the bases and activations may be useful as features in other machine learning algorithms. Second, stacking layers of these algorithms in a hierarchy may result in even more abstract and useful features that would make the job of recognizing frequently-seen objects in the environment easier. This may also provide some insight into the way the brain generates abstract features, though this is a secondary goal to developing useful unsupervised machine learning algorithms.

II. Optimization Problem Derivation

We use the following conventions:

$$\begin{aligned} x^{(i)} &\in \{-1,1\}^k, \text{ an input vector, } i \in \{1...m\}. \\ b^{(l)} &\in \mathbb{R}^k, \text{ a basis vector, } l \in \{1...n\}. \\ s^{(i)} &\in \mathbb{R}^n, \text{ an activations vector, } i \in \{1...m\}. \end{aligned}$$

Because we working with 3D point cloud data, the the receptive fields in this case are cubes. The geometry of the data is ignored, however, and the receptive field cubes are vectorized to create the inputs $x^{(i)}$. The same reasoning applies for the bases. Matrices X, B, and S have columns of inputs, activations, and bases respectively. B can also be seen as a matrix with rows of b_l^T vectors.

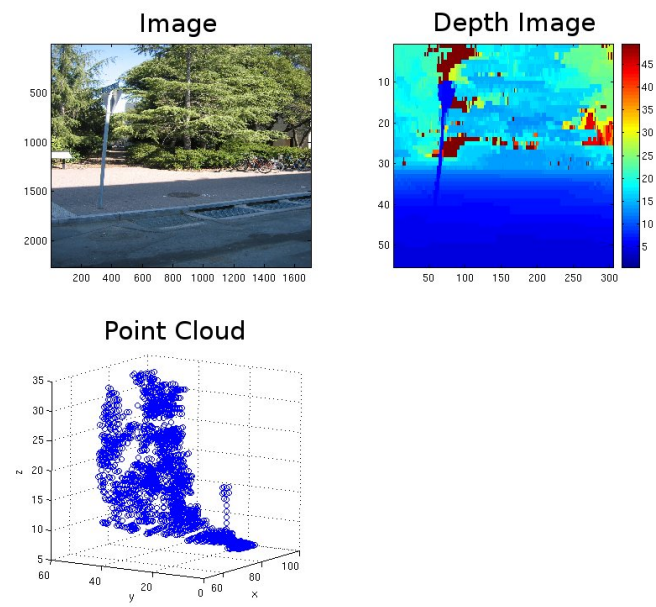

Fig. 1. An example of Ashutosh Saxena's point cloud data of natural scenes [1].

$$\begin{aligned} B &= \begin{bmatrix} | & | & & | \\ b^{(1)} & b^{(2)} & \cdots & b^{(n)} \\ | & | & & | \end{bmatrix} \\ &= \begin{bmatrix} — & b_1^T & — \\ — & b_2^T & — \\ & \vdots & \\ — & b_k^T & — \end{bmatrix} \end{aligned}$$

We make the following assumptions to reflect the binary nature of the data and the sparsity of the activations.

$$
\begin{aligned}
P(x^{(i)}|s^{(i)}, B) &= \prod_{i=1}^{m} P(x_j^{(i)}|s^{(i)}, b_j) \\
P(x_j^{(i)}|s^{(i)}, b_j) &= \sigma(x_j^{(i)} b_j^T s^{(i)}) \\
&= \frac{1}{1+\exp(-x_j^{(i)} b_j^T s^{(i)})} \\
P(s^{(i)}) &\propto \exp(-\beta ||s^{(i)}||_1)
\end{aligned}
$$

Starting with the usual MAP estimate used in sparse coding and applying the above assumptions, we have the optimization problem

$$
\begin{aligned}
\min_{S,B} \quad & \sum_i \beta ||s^{(i)}||_1 - \sum_i \sum_j \log P(x_j^{(i)}|s^{(i)}, b_j) \quad (1) \\
s.t. \quad & ||b^{(l)}||_2^2 \leq 1, \ \forall l.
\end{aligned}
$$

The norm constraint on the bases is necessary because there always exists a linear transformation of $b^{(l)}$ and $s^{(i)}$ which will not change the reconstruction term but will make the sparsity term go to zero.

This optimization problem will be solved by alternating minimization. First, consider the case of holding the bases fixed and finding the activations. (1) can then be written in the following form.

$$
\begin{aligned}
\min_S \quad & \beta \sum_i ||s^{(i)}||_1 - \sum_i \sum_j \log P(x_j^{(i)}|b_j, s^{(i)}) \\
\textstyle\sum_i \quad & \min_{s^{(i)}} \left(\beta ||s^{(i)}||_1 - \sum_j \log P(x_j^{(i)}|b_j, s^{(i)}) \right) \quad (2)
\end{aligned}
$$

Each of the $i = 1..m$ minimization problems in (2) can be solved efficiently using the L1 regularized logistic regression algorithm described in [4].

Now consider the case of holding the activations fixed and finding the bases. We can then write (1) in the following form and use projected gradient descent to solve it.

$$
\begin{aligned}
\min_B \quad & -\sum_i \sum_j \log \frac{1}{1+\exp(-x_j^{(i)} b_j^T s^{(i)})} \\
s.t. \quad & ||b^{(l)}||_2^2 \leq 1, \ \forall l
\end{aligned}
$$

Gradient descent is run on the objective function only; at every iteration, B is projected back to the feasible set. a_j is a column vector used to select a column from B^T (i.e. one of the b_j's). This gives us the update rule

$$
\begin{aligned}
B &:= B - \alpha \nabla_B \text{ (obj)} \\
\nabla_B(obj) &= -\sum_{i=1}^{m} \sum_{j=1}^{k} \frac{x_j^{(i)} a_j s^{(i)^T}}{1+\exp\left(x_j^{(i)} s^{(i)^T} B^T a_j\right)}.
\end{aligned}
$$

Several other methods were tested for calculating the bases. L1 logistic regression with the constraint on the b_j's and gradient descent on the objective function subject to $\sum_l ||b^{(l)}||_2^2 \leq 1$ both failed, likely because of the inadequacy of the constraints.

III. Results

The alternating minimization described in the previous section was run on ten thousand 5x5x5 cubic samples (discretized at three points per meter) from Ashutosh Saxena's laser scans of natural scenes [1]. Convergence was declared when the change in all of the bases or all of the minimizations (defined by euclidean distance of the vectors) dropped below 10^{-4} for at least ten iterations or when 100 iterations completed. The number of bases was chosen to be thirty.

All code was written in Matlab; the Matlab version of the L1 regularized logistic regression solver made available along with [4] was used for the calculation of the activations.

The resulting bases seem to be a mix of planes and gradients at different orientations and small sets of points with no apparent structure. An example of typical bases that result are shown in Fig. 2. The average time to calculate the activations given the bases and an input vector was .0034 seconds; the average time to compute the bases with projected gradient descent given the activations and all the inputs was 15.9 seconds. To give a rough idea of total computation time, the test which produced the example in Fig. 2 took about an hour and a half to converge on a Linux box with an Intel Core 2 Duo T7500, 2.2GHz processor and 2GB RAM.

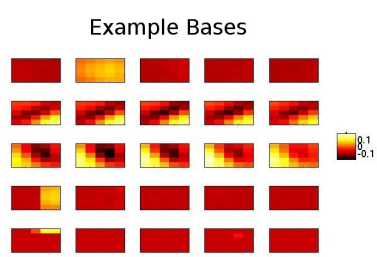

Fig. 2. Five bases are shown here; since they are a 5x5x5 cube of real numbers, 5 slices of each base are shown across the rows.

IV. Discussion

The calculation of the bases is relatively slow, but the calculation of the activations is very fast. With the current code, finding the activations for, say, 100 inputs that comprise the "eye" of a robot would correspond to a processing rate of about 3Hz. This is approaching the rate required for useful real time operation. Further, this code was all in Matlab, and it is likely that faster implementations could be produced. This is encouraging.

A. Shortcomings

Beacuse the activations are positive or negative numbers, there is not a natural interpretation for stacking the algorithm, i.e. making the outputs of one the inputs of the next. This is a somewhat serious concern.

Further, it was inteded that the basis vector activations correspond roughly to which features are present in the data; however, this isn't the case when negative activations are possible. It makes sense to allow positive and negative values for elements of basis vectors in the same way that neurons in the early visual system respond to input with light present in one area and absent in another. However, reconstructing the input by *subtracting* features doesn't have a neural correlate that I am aware of - but I am not an expert in this. More importantly, it seems that positive activations (i.e. presences of basis vectors in the input data) could be more useful in classification tasks. That intuition may be completely wrong, but it is something to explore further.

For example, look at the activations for the input in Fig. 3. The base in the first row is used to subtract from all layers except the middle. The next two bases are gradients and are used to add to the region with the points present. The input, however, has nothing to do with planes or gradients.

These two shortcomings could be addressed by changing the assumptions so that $s^{(i)} \in \mathbb{R}^n_+$. The binary input to the next layer could then be generated from $P(s^{(i)}|x^{(i)}, B)$. Another approach might be to try $s^{(i)} \in \{0,1\}^n$ with a larger set of basis vectors to make up for the lack of granularity in the activations.

B. Future Work

Despite the discussion in Section IV-A, the first thing to do is see if the resulting sparse representation of the data can be used for anything useful. Two immediate opportunities present themselves: using the bases as features in detecting the presence or absence of cars in Velodyne data from Junior and using the bases as features in finding a grasping point for objects with STAIR. Suggestions for other applications are welcome.

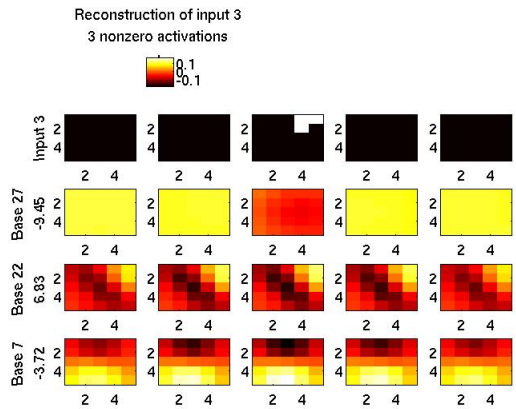

Fig. 3. An example of an undesirable reconstruction resulting from negative activations. For the input, white indicates a point that is present. The numbers underneath the base labels indicate the value of the activations.

Some tweaking of the current algorithm also needs to occur. It is possible that the less-desireable bases - those with just a small set of points with no apparent structure - can be removed by training on a larger and more varied dataset or changing the sparsity parameter β. Also, cross validation over the number of bases would be interesting to see; only the pre-set choice of 30 bases has been tested so far.

Finally, it would be interesting to test this algorithm on binary data of a different sensor modality to see if the resulting bases are useful.

V. Conclusions

In this paper, we derive an efficient sparse coding algorithm for binary inputs. It is possible that the resulting bases could be used as better features for other machine learning algorithms. It is also possible that stacking modified versions of this algorithm would result in hierarchies of more abstract features that would be even more useful.

Building on the very efficient implementation of L1 regularized logistic regression in [4], we show that this method has the potential to be useful in real time applications.

VI. Acknowledgements

Many people were helpful in the creation of this work. Thanks to Honglak Lee, Rajat Raina, Suin Lee, Ashutosh Saxena, Varun Ganapathi, Dan Ramage, Quoc Le, Paul Baumstark, and Catie Chang for making code available, making data available, and/or discussions on the derivation.

REFERENCES

[1] A. Saxena, A. Ng, and S. Chung, "Learning depth from single monocular images," *NIPS*, vol. 18, 2005. [Online]. Available: http://ai.stanford.edu/ asaxena/learningdepth/

[2] B. A. Olshausen and D. J. Field, "Sparse coding of sensory inputs," *Current Opinion in Neurobiology*, vol. 14, no. 4, pp. 481–487, August 2004. [Online]. Available: http://dx.doi.org/10.1016/j.conb.2004.07.007

[3] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. Platt, and T. Hoffman, Eds. Cambridge, MA: MIT Press, 2007, pp. 801–808.

[4] S.-I. Lee, H. Lee, P. Abbeel, and A. Y. Ng, "Efficient l1 regularized logistic regression," in *AAAI*. AAAI Press, 2006. [Online]. Available: http://dblp.uni-trier.de/db/conf/aaai/aaai2006.html#LeeLAN06