K-MAPPINGS AND REGRESSION TREES

Yi Wang[*] *Arthur Szlam*[†]

SAMSI and Duke University City College of New York

ABSTRACT

We describe a method for learning a piecewise affine approximation to a mapping $f : \mathbb{R}^d \to \mathbb{R}^p$ given a labeled training set of examples $\{x_1, ..., x_n\} = X \subset \mathbb{R}^d$ and targets $\{y_1 = f(x_1), ..., y_n = f(x_n)\} = Y \subset \mathbb{R}^p$. The method first trains a binary subdivision tree that splits across hyperplanes in X corresponding to high variance directions in Y. A fixed number K of affine regressors of rank q are then trained via a K-means like iterative algorithm, where each leaf must vote on its best fit mapping, and each mapping is updated as the best fit for the collection of leaves that chose it.

Index Terms— Piecewise linear Regression, Partial Least squares, Sparse Modeling.

1. INTRODUCTION

In this work we discuss a method for building a piecewise affine regressor of a function $f : \mathbb{R}^d \to \mathbb{R}^p$ given a set of training points $\{x_1, ..., x_n\} = X \subset \mathbb{R}^d$ and targets $\{y_1 = f(x_1), ..., y_n = f(x_n)\} = Y \subset \mathbb{R}^p$. The number of pieces K and the local rank q will be used to control the complexity of the class of regressors.

With K and q fixed, if we choose to measure error in the l_2 sense, and if we allow any partition of the training data, optimizing the error on the training set takes the form

$$\operatorname{argmin}_{\substack{M_1,...,M_K \\ P_1,...P_K}} \sum_{j=1}^{K} \sum_{x \in P_j} ||M_j \begin{bmatrix} x \\ 1 \end{bmatrix} - f(x)||^2, \qquad (1)$$

where each M_j is an affine map, and each P_j is a set of training points using that mapping. The optimal partition depends on the choice of mapping parameters, and the mapping parameters depend on the choice of partition, but solving each of these subproblems with the other fixed (on the training data) is straightforward; this suggests a K-means like iteration for solving them jointly.

Once (1) has been optimized, there is not an obvious method to find the target of an unseen test point x, as x does not belong to any P_j. One approach is to train a classifier to discriminate between the various P_j, and assign x to the P_j decided by the classifier [1]. We have found this approach to be unsatisfactory. Instead, we will first divide $\mathbb{R}^d$ into L pieces $Q_1, ...Q_L$ so that an unseen point can easily be assigned to a Q at test time, but so that the partition elements make sense in Y. Here, it is possible that $L \gg K$. We then modify problem (1) to assign each Q to a map, instead of assigning each individual point to a map. We will build Q via a hierarchical binary clustering, where each subdivision is via a hyperplane in $\mathbb{R}^d$ that attempts to maximize the variance of the associated subdivision in Y.

The contribution of this paper is to build piecewise affine tree structured models as in [2, 3], but using the subdivision criteria as in Section 2, and a global energy on the pieces with an explicit constraint on the rank and number of pieces, analogous to (1), but operating on each Q_s as a group. This leads to an accurate regressor on challenging problems that can be applied quickly to unseen test points.

2. PARTITIONING THE TARGET VIA HYPERPLANE SUBDIVISIONS OF THE DOMAIN

Our final goal is a piecewise linear mapping from $\mathbb{R}^d$ to $\mathbb{R}^p$ that maps X to Y that can be applied efficiently to unseen data points. We will explicitly demarcate the pieces via a tree structured binary "clustering" of X, even though we do not expect X to have reasonable clusters with respect to locality in Y. Our first goal is thus to build a binary partition of Y cutting along large variance directions that is not too complicated to describe in X.

2.1. Partitioning Y

We assume that nearby points in Y are similar, and furthermore, that Y is locally low dimensional in the sense of [4], so that a locally (in Y) low rank mapping makes sense. In this case, [4] shows that a multiscale binary clustering of Y obtained by recursively splitting along directions of large variance adapts to the local geometry. That is: the diameter of the pieces at the leaves of the binary clustering decays at the rate one would expect based on the local intrinsic dimension of Y.

[*]Thanks to NSF award DMS-09-56072 (awarded to Gilad Lerman) for support at the early stage.

[†]Sloan Foundation and NSF no...

2.2. Clustering X with respect to Y

We would like to mirror the clustering on Y with a clustering on X with the property that at test time it is possible to assign an unseen x to its correct cluster. However, we do not wish to assume that points in X have reasonable near neighbors that belong to the same cluster in Y.

Suppose $Y_0 \subset Y$ with cardinality N_0, X_0 is the corresponding set of X, and suppose the mean of Y_0 and X_0 is 0 (in practice we will always center the data we are operating on). For simplicity, assume Y_0 is full rank and that $d, p \leq N_0$. The direction $u \in \mathbb{R}^p$ of largest variance of Y_0 is given by

$$\text{argmax}_{u\in\mathbb{R}^p} ||Y_0^T u||^2 \text{ s.t. } ||u||^2 = 1.$$

This problem is equivalent to

$$\text{argmax}_{z\in\mathbb{R}^{N_0}} ||\Sigma V^T z||^2 \text{ s.t. } ||V^T z||^2 = 1,$$

where $U\Sigma V^T$ is the economy size SVD of Y_0, $U \in \mathbb{R}^{p\times p}$, $V \in \mathbb{R}^{N_0\times p}$ and the above equivalence can be obtained by letting $U^T u = V^T z$. Our goal is to find a subdivision of X_0 that splits Y_0 across a direction of large variance. We can thus write $z = X_0^T w$, and try to find $\text{argmax}_{w\in\mathbb{R}^d} ||\Sigma V^T X_0^T w||^2$ s.t., $||V^T X_0^T w||^2 = 1$. However, this problem is not suitable for us because $X_0^T w$ may be quite different from its projection onto the span of V, as w is not penalized for having large energy in directions unseen by V^T. Thus splitting X_0 via w need not be associated to a good splitting of Y_0. Instead, we use the problem

$$\text{argmax}_{w\in\mathbb{R}^d} ||\Sigma V^T X_0^T w||^2 \text{ s.t. } ||w||^2 = 1, \tag{2}$$

which does not exactly correspond to finding a u of largest variance in Y_0, but does still correspond to the idea of concentrating the energy of $X_0^T w$ in the large variance right singular directions of Y_0. Note that (2) is equivalent to:

$$\text{argmax}_{w\in\mathbb{R}^d} ||Y_0 X_0^T w||^2 \text{ s.t. } ||w||^2 = 1, \tag{3}$$

The problem in (3) is a Partial Least Squares Regression (see [5] and references therein). In fact, $X^T w$ gives the first extracted score vector, or latent vector of X. However, instead of using it possibly with more score vectors as reliable predictors, we use that to cluster X recursively.

2.3. Building the tree

As in [2, 3], we will use a regression tree where the leaves of the tree will point to affine maps instead of function values. The idea is to get a relatively fine subdivision of the data into "pure" pieces, but in order to not overfit, there will be relatively fewer maps, and so many leaves of the tree might be associated to a single affine map.

We recursively subdivide X and Y as above, from the top down. In the experiments below, we always divide at the median. Because each decision in the tree is built as a solution to problem (2) or (4) below, it can be evaluated with a single inner product in the X space, so efficient out of training sample extension is built into the construction.

2.4. 2-means

In order to smooth out artifacts from the cluster boundaries, it is useful to average over many models. One way to inject randomness into the construction is to replace the problem (2), which has an explicit, deterministic solution, with a K-means like algorithm, where the randomness comes from the initialization. Note that the difference between the 2-means centers approximates the largest variance direction of the data [6].

To construct the binary partition, we run the 2-means algorithm (via Lloyd iteration) on Y finding cluster centers y_1 and y_2, set $u = (y_1 - y_2)^T Y$, and use

$$\text{argmax}_{w\in\mathbb{R}^d} ||uX^T w||^2 \text{ s.t. } ||w||^2 = 1, \tag{4}$$

that is, $w = Xu^T / ||Xu^T||$.

3. BUILDING THE MAPS

It is very important in this construction to average over multiple realizations of the model to smooth out the harsh boundaries of the subdivisions. We simply rebuild the model multiple times (including the tree), and average the outputs of each model. Once a tree structured partition of the data has been specified, we will build the regressor by alternating assignment of each leaf to an affine map, and then updating the best fit map via least squares.

3.1. Updating the maps and leaf assignments

Once the maps are initialized, we alternate in batch between updating the best fit mapping for each piece via least squares, and allowing each leaf in the tree to vote on which piece to belong. Denote by $Q_1, ..., Q_L$ the partition of the data at the leaves, and $M_1, ..., M_K$ the affine maps (here each M is the parameters for the map, a $p \times d$ matrix and a p vector). Let y_x be the target of x, then

$$s(Q_i) = \text{argmin}_j \textstyle\sum_{x\in Q_i} ||M_j \begin{bmatrix} x \\ 1 \end{bmatrix} - y_x||^2,$$

$$M_j = \text{argmin}_{M,\ \text{rank}(M)=q} \sum_{s(Q_i)=j} \sum_{x\in Q_i} ||M \begin{bmatrix} x \\ 1 \end{bmatrix} - y_x||^2$$

Because we fix the subdivision of the data before training the maps and the assignments of points to maps, at this stage, we need not worry about the complexity of the partition, as the job of controlling the complexity of the partitions has been subsumed in the subdivision scheme. On the other hand, because the maps are trained against the fixed Q, they can to some extent make up for the defects in the partition.

Also note that once the partition is fixed, just as in K-means, these iterations (with suitable arrangements made for ties) are guaranteed to converge to a local minimum.

4. RELATION WITH PREVIOUS WORK

There is a long history of using trees for regression, e.g. [7]; there is now a huge literature on the topic. There has also been a lot of work on piecewise linear models, see the citations in [8].

Regression trees with affine models on the nodes have also been studied, e.g. [2, 3]. The method described here differs from these two in several respects: the method of subdivision, the structure of the linear maps, and the method of smoothing the outputs. In [2, 3], subdivisions are chosen along a coordinate axis that reduces the variance of the 1-dimensional target as much as possible. Section 2 in this work discusses how to generalize this to multivariate regression and subdivision along hyperplanes not parallel to the coordinate axes, so that the subdivision in Y more quickly approximates the locally linear structure there, as in [4]. The affine maps are also built in a different way in this work. In [2, 3], there is a map for each node, which is the best fit for all the data associated to that node; furthermore, the map takes as input only the coordinates used as decisions in the subtree below that node. This is a kind of adaptive low rank model, but again always parallel to the coordinate axes (and since in those works they regress 1-d functions, correlations in the output are ignored). In our model, the rank of each affine map is explicitly fixed, and there can be far fewer maps than nodes. The affine maps are chosen via a global optimization with a "coarseness" specified by the depth of the tree. The global optimization is a version of the K-means algorithm; or rather, since we use a fixed rank for each map, a version of the K-q flats algorithm [9] in Y.

In this respect, our work is similar to [1], where a similar alternating minimization algorithm is used to determine the partition of the data points. However, in that work, the "features" used to cluster the data were the "local" affine maps that were the best fit for a fixed size nearest neighborhood (in X) of each point. Since we do not assume that nearest neighborhoods in X make sense, in the setting of this paper, these nearest neighbor clusters also do not make sense; however one can think of the leaves of our subdivision tree as standing in for those local clusters. The method in that work does not have an out of sample extension baked in; rather they use an SVM to decide which piece a new data point should belong to.

Recently there has been work on a class of piecewise affine regressors that are not built on an explicit clustering, instead using sparse coding to simultaneously determine the "pieces" and the mapping [10, 11, 12]. In those works, a dictionary is trained on X and Y simultaneously, so that a data point x is coded in the X dictionary, obtaining coefficients α; these coefficients are then used to reconstruct $y = f(x)$ using the Y dictionary. These methods have the intuitive appeal that the preserved structure is in α, and all the action of f is encoded in the correspondence between dictionary elements. On the other hand, this rigidity also forces the X dictionary, via the sparse coding problem, to both choose the active set and the regression coefficients so that the X dictionary reconstructs x as best it can (as opposed to y). That is: even if the X dictionary is trained discriminatively, at test time, the coding problem is reconstructive. The method presented here can also be interpreted as a coupled sparse model, with the X dictionary being the concatenation of all the left singular vectors of each of the mappings, the number of nonzeros used for each x equal to the rank of each of the mappings, and Y dictionary being the product of the singular values and right singular vectors of each of the mappings. However, in this work, the sparsity is very structured (at most one block active), and more importantly, the active set is not chosen by the dictionary, but rather by the tree. Thus this work is similar to several recent works on finding the active set in sparse coding via approximate nearest-neighbor search type data structures [13, 14]. Recently, [15] alternated between updating an energy of the form (1) and finding a single representative for each cluster via the average in X of the points belonging to that cluster. As in the coupled sparse coding, that method chooses the representatives via the reconstruction error in X, not Y, but has the advantage that the partition is trained along with the mapping.

5. EXPERIMENTS

We compare our algorithm with several other approaches: coupled dictionary learning [11] (cplDL), linear regression on k nearest neighbors (knn-LS), K-q flats with Support Vector Machine (SVM) (kq-SVM), and kq-coupled [15]. Given a test point x, the knn-LS algorithm finds k nearest neighbors $x_1, \cdots, x_k$ from the training set X and finds a linear mapping L_x by solving a least squares problem $\min \sum_i \|y_i - L_x x_i\|^2$. Then the response for the test data is predicted as $y_x = L_x x$. The kq-SVM clusters the training responses Y into K clusters and learns a q-dimensional linear mapping within each cluster via the K-q flats algorithm. A SVM classifier is also trained on the training predictors X with the clustering results as labels. Then for a test data x, it is first assigned to the best cluster according to the SVM classifier, then y_x is given by the q-dimensional linear mapping from the assigned cluster. This method is used as a substitute for the method in [1], which was uncompetitive due to its construction with nearest neighbors in X. On the data sets on which it is computationally practical, we also test against the M5 prime algorithm of [3], as coded by Gints Jekabsons [17]. We use the SPAMS package of Julian Mairal [18] for the coupled dictionary learning, and the package [19] by Pierre Geurts for regression trees. For all the methods that allow it except the regression trees, we average over 10 models randomly initialized. The regression trees we average over 10 reconstructions, each of which was generated by 100 trees. The parameters for each of the models were obtained via cross validation

We test on three different data sets; all the results are in Table 1, where they are measured in mean squared error:

5.1. Simulated Manifolds

In this section, we try to recover a manifold from a distractor manifold. The predictor X is created by $X = T + M$, where both T and M are low dimensional manifolds. Then the response Y is a function of T. More specifically, we sample from $[-1, 1]$ at random, generating (a_1, a_2) and (c_1, c_2). Then we create

$$\begin{aligned} T = \quad & (\sin(2a_1), \sin(2a_2), \sin(2a_2 + a_1), \\ & \sin(2a_2a_1), \cos(2a_2 + a_1), \sin(2a_2a_1)) \\ M = \quad & (\cos(2c_1), \sin(c_1), \sin(c_2 + 2c_1), \\ & \cos(c_2c_1), \sin(c_2 + 2c_1), \cos(2c_2c_1)). \end{aligned}$$

X is formed by $X = R(T + M)$, where R is a random rotation, and $Y = Q(a_1, a_2, T_1 + T_2 + \epsilon, 0.1T_2)$, where $Q \in \mathbb{R}^{6\times4}$, $Q^TQ = I$ and $\epsilon \sim \mathcal{N}(0, 0.01)$ i.i.d. 10000 observations are generated for training and 2000 for testing in the same manner.

5.2. Side by Side Digits

In the experiment, the objective is to learn half of a digit from the rest of it plus an extra random digit added as a distractor. More specifically, we randomly choose two digits from the MNIST database, slightly shift and rotate them, put them side by side and then cut the right digit in half. The left part (the left digit and the left half of the right digit) will be the predictor x and the right part will be the response y. Both are vectorized and a pair of (x, y) serves as an observation. Since each digit in MNIST is a 28×28 image, x has dimension 1176 and y has dimension 392. The circular shift and rotation sizes are $[-4, 4]$ and $[-60°, 60°]$ respectively. We create 300K observations for the training data set and 10K for the testing data set. For simplicity, we call this data set SS.

A second data set SS2 is created similarly except that we cut either the left part of the left digit or the right part of the right digit, at random. The regressor has to decide which part was cut.

5.3. Faces

In the experiment, the objective is to learn half of a face from the other half; here we use no distractors. To build the data, we take face images from [20, 21], crop the middle portion, and resize to 64×64. The first 12000 (funneled) faces are used as training, and the remaining 1233 are used as the test set. The training set is augmented by taking all 9 1 pixel shifts and reflecting each face about the vertical axis. We project all of the left face halves onto the first $d = 300$ principal components of the left training data, and the right face halves onto the $p = 300$ principal components of the right training data; there are a total of 216000 training pairs; all points are projected onto the unit sphere.

Fig. 1: Display of the results on SS2. From top to bottom, the ground truth, cplDL, knn, kq-SVM, kq-coupled, RT and the proposed.

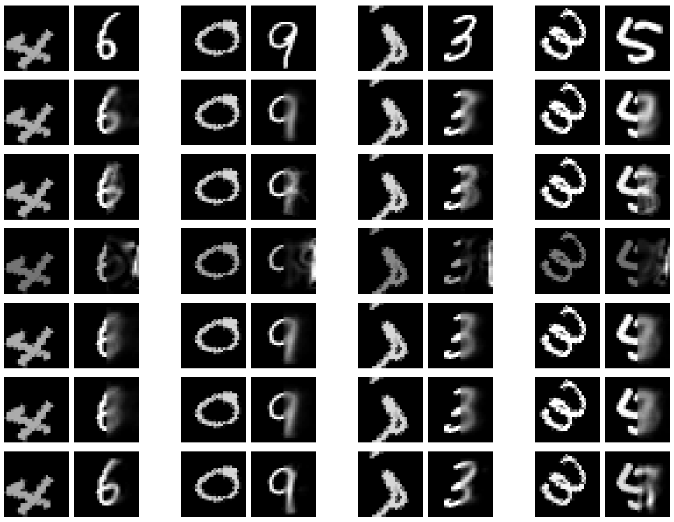

Table 1: Regression results, in MSE.

	manifold	SS	SS2	faces
m5p[3, 17]	45	NA	NA	NA
cplDL [10, 11, 18]	51	1826	1851	252
knn-LS	27	2478	2322	305
kq-SVM	56	1846	1906	317
kq-coupled [15]	41	1785	1657	255
regression trees [22, 19]	28	2085	2135	345
proposed	24	1316	1325	246

6. CONCLUSION

We presented a method for piecewise affine low rank regression. The method can be viewed as a regression tree, in the spirit of [3], but with relatively few affine models, each of which is associated to multiple leaves on the tree, or as a coupled dictionary method, with a tree structured encoder. To build the tree so that the leaves have low variance in the target and are simple to compute in the domain, we introduce a splitting method for dividing data along a hyperplane in the domain that corresponds to a large variance direction in the target. We show experiments demonstrating that on data where the target is locally low dimensional, the method can produce accurate regressions, even when the domain has been corrupted by structured noise.

7. REFERENCES

[1] Giancarlo Ferrari-Trecate and Michael Schinkel, “Conditions of optimal classification for piecewise affine regression,” in *HSCC*, 2003, pp. 188–202.

[2] Ross J. Quinlan, “Learning with continuous classes,” in *5th Australian Joint Conference on Artificial Intelligence*, Singapore, 1992, pp. 343–348, World Scientific.

[3] Yong Wang and Ian H. Witten, “Induction of model trees for predicting continuous classes,” in *Poster papers of the 9th European Conference on Machine Learning*. 1997, Springer.

[4] Nakul Verma, Samory Kpotufe, and Sanjoy Dasgupta, “Which spatial partition trees are adaptive to intrinsic dimension?,” in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, Arlington, Virginia, United States, 2009, UAI ’09, pp. 565–574, AUAI Press.

[5] Roman Rosipal and Nicole Krämer, “Overview and recent advances in partial least squares,” in *Subspace, Latent Structure and Feature Selection, Statistical and Optimization, Perspectives Workshop, SLSFS 2005, Bohinj, Slovenia, February 23-25, 2005, Revised Selected Papers*, 2005, pp. 34–51.

[6] Rafail Ostrovsky, Yuval Rabani, Leonard J. Schulman, and Chaitanya Swami, “The effectiveness of lloyd-type methods for the k-means problem.,” *Proc. of the 47th Ann. IEEE Symp. on Foundations of Computer Science*, 2006.

[7] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J Stone, *Classification and Regression Trees*, Wadsworth and Brooks, Monterey, CA, 1984.

[8] Simone Paoletti, Aleksandar Lj. Juloski, Giancarlo Ferrari-Trecate, and René Vidal, “Identification of hybrid systems: A tutorial,” *Eur. J. Control*, vol. 13, no. 2-3, pp. 242–260, 2007.

[9] Paul S Bradley and Olvi L Mangasarian, “k-plane clustering,” *Journal of Global Optimization*, vol. 16, no. 1, pp. 23–32, 2000.

[10] Jianchao Yang, John Wright, Thomas S. Huang, and Yi Ma, “Image super-resolution as sparse representation of raw image patches,” in *CVPR*, 2008.

[11] Jianchao Yang, Zhaowen Wang, Zhe Lin, Scott Cohen, and Thomas S. Huang, “Coupled dictionary training for image super-resolution,” *IEEE Transactions on Image Processing*, vol. 21, no. 8, pp. 3467–3478, 2012.

[12] Julien Mairal, Francis Bach, and Jean Ponce, “Task-driven dictionary learning,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 4, pp. 791–804, 2012.

[13] Arthur Szlam, Karol Gregor, and Yann LeCun, “Fast approximations to structured sparse coding and applications to object classification,” in *European Conference on Computer Vision (ECCV 2012)*, Florence, Italy, October 2012.

[14] Yadong Mu, John Wright, and Shih-Fu Chang, “Accelerated large scale optimization by concomitant hashing,” in *ECCV*, 2012, pp. 414–427.

[15] Naresh Manwani and P. S. Sastry, “K-plane regression,” *CoRR*, vol. abs/1211.1513, 2012.

[16] Michael I. Jordan, “Hierarchical mixtures of experts and the em algorithm,” *Neural Computation*, vol. 6, pp. 181–214, 1994.

[17] Gints Jekabsons, ,” `http://www.cs.rtu.lv/jekabsons`.

[18] Julien Mairal, ,” `http://spams-devel.gforge.inria.fr/`.

[19] Pierre Geurts, ,” `http://www.montefiore.ulg.ac.be/~geurts/Software.html`.

[20] Gary B. Huang and Vidit Jain, “Unsupervised joint alignment of complex images,” in *In ICCV*, 2007.

[21] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller, “Labeled faces in the wild: A database for studying face recognition in unconstrained environments,” Tech. Rep. 07-49, University of Massachusetts, Amherst, October 2007.

[22] Leo Breiman, “Random forests,” in *Machine Learning*, 2001, pp. 5–32.