Unsupervised learning of object models

Christopher K. I. Williams
Dept. of Computer Science
University of Toronto
Toronto, ON M5S 1A4
Canada

Richard S. Zemel
Dept. of Computer Science
University of Toronto
Toronto, ON M5S 1A4
Canada

Michael C. Mozer
Computer Science Department
University of Colorado
Boulder, CO 80309-0430
USA

Abstract

Given a set of images, each of which contains one instance of a small but unknown set of objects imaged from a random viewpoint, we show how to perform unsupervised learning to discover the object classes. To group the data into objects we use a mixture model which is trained with the EM algorithm. We have investigated characterizing the the probability distribution for the features of each object either in terms of an object model or by a Gaussian distribution. We compare the performance of these two approaches on a dataset containing six different stick-animals, and on a dataset consisting of seven hand gestures.

1 Introduction

Imagine that you are given a set of images, each containing an instance of one of a small, but unknown set of objects, imaged from a random viewpoint. For example, the object models could be 2-d shapes and the projection an affine transformation; or the object models could be 3-d shapes, which are orthographically projected onto the 2-d image plane. Your task is to sort the images by object model. This is an unsupervised task; you are not given the object models, nor are the images preclassified.

Work on supervised classification often assumes that the correspondences between features across images are known. If we make the same assumption, then the unsupervised task is to untangle the variation in the data due to (i) seeing the same object from different viewpoints and (ii) seeing different objects.

The viewpoint variability can be handled by characterizing a probability distribution in feature space for each object. This may have a parametric form, or be based on models of the objects; we consider both alternatives below. The unsupervised algorithm is obtained by using a *mixture model* of these distributions (one distribution per object), and adapting the parameters of the distributions so that the likelihood of the data is maximized.

This unsupervised method may be useful if you were trying to learn geometric models for a number of fairly similar objects (like chairs, for example) from data that is not labelled with the object identity. A single object model would learn the average of the appropriate class specific models, but the mixture model approach should learn a model for each class.

The organization of the rest of the paper is as follows: in section 2 we investigate what kinds of probability distributions can be expected in feature space from the imaging of object models. In section 3 these distributions are used to build classifiers. Section 4 shows how to create an unsupervised learning algorithm by using mixture models. Results of experiments on two datasets are reported in section 5, and section 6 contains a discussion of the results.

2 Probability distributions in feature space

Our objects are comprised of n (uniquely identified) features, and each feature is described by a vector of k numbers. Any view will correspond to a point in the nk dimensional *feature space.* We may choose some well defined points on the objects as features so that in the image each feature will be described by the x and y coordinates of the point, but it would also be possible to choose features such as line segments that have orientation and scale as well as location in the image. If we know the spatial relations between the features that make up the rigid object, and we know the *instantiation parameters* (position, orientation, and size, for example) of the object, then we can calculate the values of the feature vectors.

If the vector of object instantiation parameters is denoted by $\mathbf{z}$, then the predicted feature vector for feature j is given by $\hat{\mathbf{x}}_j = \mathbf{f}_j(\mathbf{z})$. If the transformation from $\mathbf{z}$ to $\hat{\mathbf{x}}_j$ is linear, then $\mathbf{f}_j$ is a matrix. Assuming a Gaussian probability distribution for the observed feature vector $\mathbf{x}_j$ given the predicted vector $\hat{\mathbf{x}}_j$, we obtain

$$P(\mathbf{x}_j|\mathbf{z}) = \frac{1}{(2\pi)^{k/2}|W_j|^{1/2}} \exp[-\frac{1}{2}(\mathbf{x}_j-\hat{\mathbf{x}}_j)^T W_j^{-1}(\mathbf{x}_j-\hat{\mathbf{x}}_j)] \tag{1}$$

where W_j is the covariance matrix used to calculate the *Mahalanobis distance* $(\mathbf{x}_j - \hat{\mathbf{x}}_j)^T W_j^{-1}(\mathbf{x}_j - \hat{\mathbf{x}}_j)$ between the predicted values for feature j and the observations. In our experiments we have used $W_j = \sigma^2 I$ for all j. The probability of observing all n features is simply calculated by multiplying together the probabilities for the individual features, assuming independence.

Letting $\mathbf{x}$ be the concatenation of all n vectors $\mathbf{x}_j$, then $P(\mathbf{x}|\mathbf{z}) = \prod_j P(\mathbf{x}_j|\mathbf{z})$.

To define a probability distribution in feature space based on equation 1, we would need to know the prior probability distribution of the instantiation parameters $P(\mathbf{z})$, to obtain $P(\mathbf{x}) = \int P(\mathbf{x}|\mathbf{z})P(\mathbf{z})d\mathbf{z}$. However, it is not really necessary to do the integration if we only want to compare the probability of a set of observed features under different models; we can just compare the maximum values $\hat{P}(\mathbf{x}) = \max_{\mathbf{z}} P(\mathbf{x}|\mathbf{z})$. Assuming a non-informative prior for all models, and that all of the W_j's are equal, this maximum likelihood approach is justified because the peaks of the posterior distributions will have the same variance, so the integral will factor into $\hat{P}(\mathbf{x})$ times a hypervolume factor which is the same for all models. If these conditions do not hold, or if the models have different numbers of degrees of freedom, the (Bayesian) integration method should be preferred over the maximum likelihood approach.

If $\mathbf{f}_j$, the function that predicts the feature vector given the instantiation parameters, is linear, the best fit instantiation parameters can be found analytically because the distribution $P(\mathbf{x}|\mathbf{z})$ turns out to be a multivariate Gaussian in $\mathbf{z}$ given the image data, i.e. $P(\mathbf{x}|\mathbf{z}) = \hat{P}(\mathbf{x})\exp\{-(\mathbf{z}-\mathbf{z}^*)^T(\mathbf{z}-\mathbf{z}^*)/2\sigma^2\}$, where $\mathbf{z}^*$ denotes the best fit instantiation parameters.

As an example, consider the case of models made up of n 2-d point features imaged under 2-d affine transformations (corresponding to the orthographic projection of a planar object viewed from a general position). The x and y coordinates of point j in the object-based frame are (μ_x^j, μ_y^j). The instantiation parameters are denoted (t_x, t_y, a, b, c, d), where t_x and t_y are the x and y translations and the other parameters depend on the rotation and scaling of the view, so that the predicted position $(\hat{x}_j, \hat{y}_j)$ of feature j is given by

$$\hat{x}_j = a\mu_x^j + b\mu_y^j + t_x \quad (2)$$

$$\hat{y}_j = c\mu_y^j + d\mu_x^j + t_y \quad (3)$$

If there was no observation noise, the observations would lie on a 6-d subspace of the $2n$-dimensional space. This subspace can be generated by taking linear combinations of six basis vectors (Ullman and Basri, 1991; Edelman and Poggio, 1990). The addition of noise means that the 6-d subspace will become more "fuzzy", occupying a finite volume of the $2n$-dimensional space. This observation leads to our alternative method of characterizing the probability distribution in feature space using a Gaussian model. This Gaussian is a higher dimensional analogue of a pancake or sausage shape; the variance will be large for directions within the 6-d subspace, and small in directions orthogonal to the subspace. In general the Gaussian will require $2n$ parameters to specify its mean, and $n(2n+1)$ parameters for the covariance matrix. The mean and the covariance matrix will depend on the first and second moments of the instantiation parameters, the positions of the features in the model and the covariance matrix of the noise.

Similar arguments can be made for other viewing transformations of interest. For 2-d point models under similarity transformations (rigid rotations, scalings and translations in the plane) the subspace is four dimensional, and for 3-d point models under affine transformations followed by orthographic projection it is eight dimensional. Orthographic projections of 3-d point models form a six dimensional non-linear subspace of this eight dimensional space.

A geometric interpretation of the difference between the model based and Gaussian approaches is as follows. For the Gaussian method, the likelihood of an observation depends on the Mahalanobis distance from the mean to the data point. For the model based approach an object is represented by a lower-dimensional (parametric) surface in the feature space, and the likelihood of a observation now depends on the minimum distance of the point from the surface.

3 Classifiers

To classify a feature vector as belonging to a particular object, we need to calculate $P(\omega_i|\mathbf{x}) \propto P(\omega_i)P(\mathbf{x}|\omega_i)$, where ω_i indexes the classes. If the prior probabilities $P(\omega_i)$ are equal, then classification reduces to finding the class that maximizes the likelihood $P(\mathbf{x}|\omega_i)$. For model based approaches, this is equivalent to finding the model which has the smallest Mahalanobis distance to the data (see equation 1), if we make the assumptions that allow the replacement of $P(\mathbf{x}|\omega_i)$ with $\hat{P}(\mathbf{x}|\omega_i)$.

Ullman's "structure from motion" theorem (1979) showed that it is possible to do (supervised) learning of the models. An example of more recent work along these lines is the paper by Tomasi and Kanade (1992). The work of Bennett et al. (1993) on "recognition polynomials" also falls into the model based category.

A technique closely related to the model based approach is to transform the data so that the effect of the viewing transformation is removed. For example, to obtain invariance to affine transformations of the image features, Lamdan and Wolfson (1988) use three of the features to define a basis, and then measure the positions of the other features relative to this basis. This method works for orthographic projections of planar objects, but cannot be extended to full 3-d objects.

It is also possible to build a classifier without using object models; see, for example, the radial basis function classifier of Poggio and Edelman (1990).

4 Mixture models

We have shown how to characterize the probability distribution in feature space for one object. Observations coming from a number of different models can be characterized by a mixture model $P_{mix}(\mathbf{x}) = \sum_i \pi_i P_i(\mathbf{x})$, where the π_i's are the "mixing proportions" of each model ($\sum_i \pi_i = 1$) and $P_i(\mathbf{x})$ is the probability of the data under model i.[1] To train the mixture model we adjust the parameters of each component in order to

[1]For the object models we can use $\hat{P}_i(\mathbf{x})$ instead of $P_i(\mathbf{x})$.

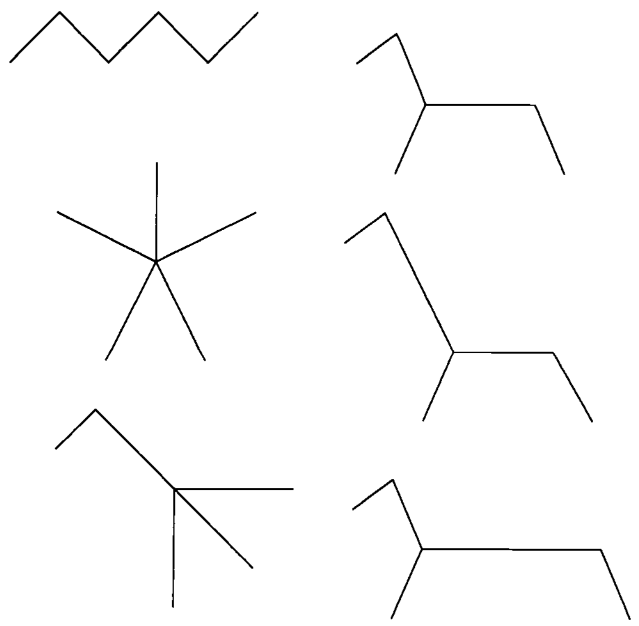

Figure 1: The six "stick figure" models of animals used to generate the "animals" dataset. From top to bottom and left to right they are snake, starfish, bird, cat, giraffe and hippo. The models are really only represented by the points; the lines are added to aid visualization.

maximize the log likelihood of the training data

$$L = \sum_c \log(\sum_i^M \pi_i P_i(\mathbf{x}_c)) \quad (4)$$

where c indexes the training cases.

The training can be done with the EM algorithm (Dempster, Laird and Rubin, 1977), where the E (Expectation) and M (Maximization) steps are applied alternately. Starting from initial randomly chosen models, on the E step we calculate the "responsibility" $r_i = \pi_i P_i(\mathbf{x}_c) / \sum_j \pi_j P_j(\mathbf{x}_c)$ of each model for each image. On the M step, having fixed the responsibilities, we adjust the model parameters to maximize the likelihood. This maximization is split into M sub-problems, one for each model. For a mixture of Gaussians model, the means and covariance matrices of each component are updated. Details of these steps can be found, for example, in McLachlan and Basford (1988).

For a mixture of object models, the M step is non linear and we used conjugate-gradient search in the experiments reported below. The value of σ^2 was re-estimated using the EM algorithm.

5 Results

We have tested the unsupervised learning procedure, using both object models and Gaussians, on two datasets, called "animals" and "hand gestures".

5.1 Datasets

The "animals" dataset is generated from the six models shown in figure 1. Each model has six feature points.

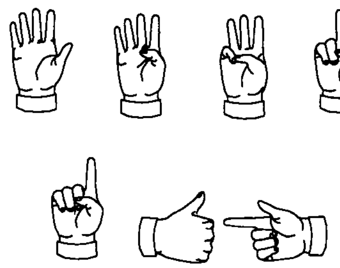

Figure 2: Examples of the seven hand gestures, "five", "four", "three", "two", "one", "thumbs up" and "point", from Ahmad and Tresp (1993).

50 training examples were generated from each model, randomly choosing a different translation, scale and rotation for each instance[2] and then adding Gaussian noise of standard deviation 0.1 to each of the generated points. A further 100 examples of each class were generated from the true models for use as a test set.

The "hand gestures" dataset is the same as was used by Ahmad and Tresp (1993). It consists of the five (x, y) locations of the five fingertips for 500 training and 500 testing examples of each of the seven gestures shown in figure 2.[3] The fingertip locations were obtained by perspective projection of the hand models at various orientations and distances; translations were not used. It should also be noted that only a limited range of depths were used. The data was scaled to so that each feature lies roughly in $[-1, 1] \times [-1, 1]$.

5.2 Training the models

For both datasets we were able to train both the model based and Gaussian classifiers in either a supervised or unsupervised manner. Supervised training is similar to the unsupervised version, except that the "responsibilities" are set to either 1 or 0 based on knowledge of the true class. The supervised training is useful as it provides a check as to whether the method can actually classify the data correctly if the correct parameters can be found.

The final parameters found by the EM algorithm for fitting mixture models will depend on the starting values, so it is necessary to try several randomly chosen initial configurations. The best run is the one that gives the

[2]Let t_x and t_y denote the translations, r the scale and θ the rotation angle, and set $c = r\cos\theta$, $s = r\sin\theta$. t_x, t_y, c and s were all sampled from a unit variance Gaussian. The models were normalized so that $\sum_i(\mu_x^i)^2 + \sum_i(\mu_y^i)^2 = 1$.

[3]We thank Subutai Ahmad for providing these datasets and figure 2.

highest likelihood of the data.

A common problem with using mixture models is trying to decide how many components are present; for example, see the discussion in McLachlan and Basford (1988). In the experiments reported below, we set the number of components to be the correct number of classes, based on knowledge of the datasets. However, experiments starting off with more components and using variable mixing proportions were often able to discover the correct number of classes. Indeed, this procedure may make the search problem easier, helping to avoid local maxima of the objective function where one component accounts for data that actually comes from two different objects.

5.3 Results on the "animals" dataset

Ten different unsupervised runs of the model based method were performed, using six randomly chosen initial models and a starting variance of 0.5. Four of these runs successfully converged to the six classes, in an average of 9 iterations of the the EM algorithm. On the other six runs typically five of the six classes were discovered. The error rate of these classifiers was found to be the same as the true generative models (because the added noise level is quite high, even the correct models give 28 errors on the 600 test examples). Even though the training data was noisy and unlabelled, the learned models closely resemble the true models; the learned feature locations were within 2% of the true locations on average.

Supervised training of a Gaussian classifier gave 32 errors on the test set, slightly worse than the unsupervised model based performance. However, it was not possible to obtain this kind of performance with an unsupervised mixture of Gaussians. We tried ten runs starting from random means at four different initial variances (6.0, 3.0, 1.0 and 0.3); none of these runs discovered the six classes. Typically the classifier confused the cat, giraffe and hippo classes, which is not too surprising as these shapes were designed to be very similar. Gaussian mixture models were successfully trained on a dataset containing only four the classes hippo, snake, starfish and bird.

The performance of the (supervised) Gaussian classifier was affected more severely than the models as the amount of training data was reduced. This is explained by the greater number of free parameters in the covariance matrix compared to the models.

5.4 Results on the "hand gestures" dataset

To use the model based approach, we have to decide on the specification of the models and the allowed transformations. Figure 2 suggests that for each hand shape, the fingertips all approximately lie in a plane. Hence we used the 2-d affine model of equations 2 and 3. Supervised training of the models leads to a 96.7% correct performance on the test set, compared to a best of 93.3% reported by Ahmad and Tresp. However, ten unsupervised runs using seven models failed to find all seven classes. Gestures "two" and "three" were always discovered, but there was always some confusion between the other hand shapes. We were able to find the correct solution by starting with fifteen models, allowing the mixing proportions to be re-estimated and then using a validation set of data (held out of the training data) to select the top seven models.

Two out of ten runs of the mixture of Gaussians method successfully discovered the seven classes in the data in about 20 iterations. The performance of these classifiers was very impressive; there were only five errors in 3500 test examples, giving 99.8% correct. A similar performance was found with the supervised Gaussian classifier.

The superior performance of the Gaussian classifier in comparison to the model based classifier suggests that the assumptions used for the models may not be quite correct; one possibility is that the approximation that all the fingertips lie in a plane will not hold exactly. We tried supervised learning of three-dimensional models on the hand data, but obtained only 82.3% correct. This is due to the fact that the transformation used has eight degrees of freedom, which means that all models will be able to obtain quite a close fit to any given set of five fingertip locations. This problem could be reduced by constraining the transformations, or by having more features[4].

An explanation for the excellent performance of the mixture of Gaussians can be found in the generation of the "hands" dataset; there was limited depth variation and no translational offsets, so the Gaussians will not have learned about variation in these directions. The object models have this variability built in, and thus will have a distribution $P(\mathbf{x}|\omega_i)$ that is less tightly tuned to the data. This was demonstrated by adding random x and y offsets to each image in the "hands" dataset; the performance declined to only 63.7% correct for the Gaussians that had been trained on the unperturbed data, but remained the same for the model based approach.

6 Discussion

We have shown that both a model based approach and a mixture of Gaussians are able to discover objects with unsupervised learning. There are a number of advantages of the model based approach:

- The output is much more meaningful than the covariance matrices produced by the mixtures of Gaussians.
- The search problem in unsupervised learning should be helped by the fact that we are effectively biasing the choice of covariance matrices to the ones that should have the correct invariance properties. This is supported by the experiments on the "animals" data, where the model based approach was able discover all six classes, but the mixture of Gaussians could not.
- The desired invariance properties can be built in, even if they are not fully present in the training data (e.g. the effects of translations with the "hand gestures" dataset).

[4]E.g. the recognition of six-fingered hands.

- The instantiation parameters at the output level can be used as input features to a higher level in an hierarchical scheme.
- Fewer parameters are needed to describe a model as compared to a general covariance matrix[5], which implies that the models should need less training data to achieve equal performance.

On the other hand, there are also some disadvantages

- It is necessary to know what kinds of viewing transformations are possible in order to calculate $\hat{P}(\mathbf{x})$.
- If not all of the degrees of freedom in the transformation are used, better performance may be obtained by another method (e.g. the success of the Gaussian mixture model on unsupervised learning of the hand data).
- The calculations needed to fit the models to the data are rather more complicated than those for fitting mixtures of Gaussians.

The work can be extended to handle missing features very readily—the necessary marginal distributions are found by simply ignoring the features for which we don't have data, and then doing the same kind of calculations as before (c.f. Ahmad and Tresp, 1993).

It should also be possible extend the approach to cases where the computation of the best-fit parameters is not possible analytically; we can simply use an "inner loop" search (like conjugate-gradient). This would be necessary if the feature generation model was extended to handle outliers by using a mixture of Gaussians distribution, or for the case of strict orthographic projection of 3-d objects where there are six free parameters (x and y translations, scaling and three Euler angles).

One of the aims of this paper has been to conduct an investigation of the use of machine learning techniques in computer vision, using image representations (like feature locations) which are rather different to the pixellated images often used in connectionist object recognition work. This stems from the idea that it is not only the kinds of features observed, but also their relative spatial arrangement in the image that is important for object recognition. By explicitly building in knowledge of imaging transformations rather than letting networks discover them it should be possible to considerably reduce the amount of training data required (c.f. TRAFFIC, Zemel, Mozer and Hinton, 1990).

Acknowledgements

The authors would like to thank Geoffrey Hinton for his generous support and guidance. We thank Subutai Ahmad, Richard Mann, and members of the Connectionist Research Group at the University of Toronto for helpful discussions and comments. This research was supported by a grant from the Information Technology Research Centre of Ontario to Geoffrey Hinton, and by NSF PYI award IRI–9058450 and Grant 90–21 from the James S. McDonnell Foundation to MM.

[5]Providing that the noise covariance matrix has a simple form.

References

Ahmad, S. and Tresp, V. (1993). Some Solutions to the Missing Feature Problem in Vision. In Hanson, S. J., Cowan, J. D., and Giles, C. L., editors, *Neural Information Processing Systems, Vol. 5.* Morgan Kaufmann, San Mateo, CA.

Bennett, B. M., Hoffman, D. D., and Prakash, C. (1993). Recognition polynomials. *J. Opt. Soc. Am. A*, 10(4):759–764.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Proceedings of the Royal Statistical Society*, B-39:1–38.

Edelman, S. and Poggio, T. (1990). Bringing the Grandmother back into the picture: a memory-based view of object recognition. Technical Report 1181, AI Laboratory, MIT.

Lamdan, Y. and Wolfson, H. J. (1988). Geometric Hashing: A General and Efficient Model-Based Recognition Scheme. In *Proceedings of the Second International Conference on Computer Vision.* IEEE.

McLachlan, G. J. and Basford, K. E. (1988). *Mixture models: inference and applications to clustering.* Marcel Dekker, Inc.

Poggio, T. and Edelman, S. (1990). A network that learns to recognize three-dimensional objects. *Nature*, 343:263–266.

Tomasi, C. and Kanade, T. (1992). Shape and Motion from Image Streams under Orthography: a Factorization Method. *International Journal of Computer Vision*, 9(2):137–154.

Ullman, S. (1979). *The Interpretation of Visual Motion.* MIT Press.

Ullman, S. and Basri, R. (1991). Recognition by Linear Combinations of Models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(10):992–1006.

Zemel, R. S., Mozer, M. C., and Hinton, G. E. (1990). TRAFFIC: Recognizing objects using hierarchical reference frame transformations. In Touretzky, D. S., editor, *Neural Information Processing Systems, Vol. 2*, pages 266–273. Morgan Kaufmann, San Mateo, CA.