Recognizing American Sign Language Letters: A Machine Learning Experience in an Introductory AI Course

Ellen L. Walker

Department of Computer Science
Hiram College
Hiram, OH, 44240

walkerel@hiram.edu

Abstract

This paper describes a class project to introduce machine learning topics to an introductory artificial intelligence course as part of the MLExAI Project. The project's topic was taken from the area of computer vision, specifically the use of principal component analysis for image classification. As a project within their AI class, students developed programs in the GNU Octave programming environment that successfully classified images of hands signing letters from American Sign Language according to the letter being signed. Based on surveys and analysis of course examinations, we found that students enjoyed and were motivated by the project, and the project seemed to enhance their learning of machine learning concepts.

Introduction

Today's students are visual and media-oriented, accounting for the success of the media computing approach to introductory computer science (Rich, Perry and Guzdial, 2004; Sloan and Troy, 2008). Students in upper-division courses continue to find working with images and other media quite motivating. For example, graphics and gaming courses are among the most popular computer science electives at many colleges and universities, including our own. Since many approaches to computer vision now use machine learning techniques (Sebe et. al., 2005), adapting a computer vision problem for a machine learning project in Artificial Intelligence (AI) is a natural fit. This paper describes a group term project in which students in an undergraduate, introductory artificial intelligence course developed programs that applied the technique of Principal Component Analysis (PCA) to the problem of recognizing sign language letters in images.

The sign language recognition project was developed within Project MLExAI: Machine Learning Experiences in Artificial Intelligence (Russell and Markov, 2009). Project MLExAI is a framework for teaching core AI topics within introductory artificial intelligence courses through a unifying theme of Machine Learning. Each project involves the development of a machine learning system in a specific application. The applications span a large area including network security, recommender systems, game playing, intelligent agents, computational chemistry, robotics, conversational systems, cryptography, web document classification, vision, data integration in databases, bioinformatics, pattern recognition, and data mining. A pilot project (Markov et. al. 2005; Russell et. al, 2007) showed that students were more motivated in their AI classes when topics were tied to a concrete, hands-on project.

Background

The course, *Introduction to Artificial Intelligence* at our college is an undergraduate course open to all students who have completed at least one programming course and a short course in Lisp. Although *Data Structures* is not a formal requirement for entry into the course, we strongly recommend it, and all of the students who took Artificial Intelligence in Spring 2009 had taken *Data Structures*. before These students range from sophomores to seniors; in Spring 2009, there were 2 sophomores, 5 juniors and 5 seniors. Students' mathematical background varied widely, but most had not taken a course with the necessary prerequisite mathematics; only two had taken a college-level statistics course, and only one student had a prior course in linear algebra. Therefore, mathematical topics including aspects of linear algebra and statistics, were introduced in class as needed for the project.

In the past, the course had been taught using an agent theme, fairly closely following the outline of the textbook by Russell and Norvig (2003). We began with an overview of agents, then covered search, logic and inference, planning, and reasoning under uncertainty before covering learning almost at the end of the course. Students were often disappointed by the fact that machine

Phase	Topic
0	Introduction to Classification
1	Image Representation & GNU Octave
2	PCA Feature Extraction (Training)
3	Letter Recognition (Testing)
4	Evaluation and Reporting

Table 1: Overview of the Sign Language Assignment

learning was one of the topics that attracted them into the course, but it was delayed so long that there was not enough time to really get into it.

To incorporate the project, we moved learning much earlier in the course, between search and logic. Except for the difference in ordering and the additional emphasis on learning, roughly the same topics were covered. The project was carried out in five phases (labeled Phase 0 to Phase 4), listed in Table 1. While students were working on one phase of the project, the class moved on to new material. Project phases 3 and 4 were carried out while topics after learning were covered during the lecture portion of the course.

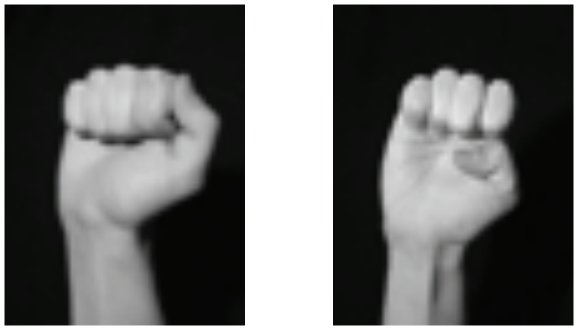
Typically, there had been four programming assignments in the course, each taking roughly 25% of the semester, along with smaller problem sets, no more than a week in duration. The first of these projects, implementing a heuristic search in Lisp, was retained in the revised course. This project took advantage of the students' prior experience with the language and environment. The remaining three projects were replaced by the multi-phase sign language recognition project and a short assignment that required students to represent a simple world in both Prolog and CLIPS.

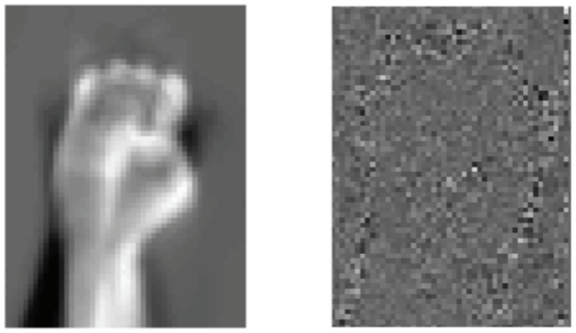
Object Recognition using Principal Component Analysis

The main techniques in the project were inspired by the seminal work on Eigenfaces by Kirby and Sirovich (1990). In this work, a system was trained to recognize faces using Principal Component Analysis (PCA). In our project, images of a hand signing a letter from American Sign Language were used instead. The image is cropped to a standard size, with the hand taking up the large majority of the image. Images have a uniformly black background. (See Figure 1).

Initially, each pixel of the image is considered to be a single feature, so an image with M rows and N columns image would have MN numerical features. These features form a high-dimensional feature space. PCA chooses a new set of axes to span the space, creating a new set of features that better describe the training images. Each of the new features is a linear combination of image pixel values.

The features chosen by PCA are actually the eigenvectors of the covariance matrix of the original image. These features can be sorted by their associated eigenvalues, in that the eigenvector with the highest


(a) letter A (b) letter E
Figure 1: Sample Training Images


(a) 2nd eigenvalue (b) 1000th eigenvalue
Figure 2: PCA Features

eigenvalue explains the most variation in the image. A relatively small subset of these eigenvectors is a sufficient set of features for classification.

To determine the feature's value for an image, the image is normalized and then multiplied (as a vector dot product) with the eigenvector representing the feature. The features for each of the training examples are stored, and features for a test image are computed. Various techniques can then be used to select the best label for the test image. We use a simple nearest-neighbor classifier, i.e. the test image is classified with the same label as the training image whose feature vector is nearest (using a Euclidean distance measure) to the test image. An appropriate extension of the project would be to ask

A key advantage of using an image processing problem as a machine learning example is that the intermediate results can easily be visualized. Each element of an eigenvector represents a coefficient on a single image pixel. Displaying the eigenvector as an image (so that the first row is the first M values, the second row is the second M values, etc.) provides a clear indication of the relative importance of the pixels (and whether they are expected to be positive or negative) in the feature. In fact, each feature can be intuitively considered as a template image to be matched.

Images that are generated from high-eigenvalue features tend to highlight features that humans find useful in these very structured images, while images that are generated from low-eigenvalue features look like noise to humans. This is illustrated in Figure 2. Image (a) is the second-highest eigenvalue for one set of training data. It looks like a bright E with a shadowy A behind it. When this is multiplied by an image, it will distinguish E-like images from A-like images. Image (b) is the thousandth-highest eigenvalue from the same data set. It appears to be making distinctions on a pixel-by-pixel basis, and it is not at all intuitive what these pixels mean. Therefore, students working with these images can clearly see the relationship between eigenvalues and features, and can also gain intuition about the meanings and effects of these numerical vector features.

For images that are structured as these are (similar size and shape object against contrasting background), a subset of the high-eigenvalue features is generally sufficient information to perform classification on. Another use for these features is for a form of lossy image compression, though we did not investigate this in the assignment.

The Class Project

The overall goal of the project was for students to classify images such as those in Figure 1 according to the letter being signed in each image. To do this, students extracted appropriate PCA features from a training set of images, and then used nearest-neighbor classification to select a letter for each test image. Such a system might be used within a larger system for automatically captioning sign language videos, for example. As a result of this experience, we expected the students to satisfy the following learning objectives:

- Learning basic concepts of object recognition in computer vision
- Becoming familiar with the concepts of feature spaces and classification
- Becoming familiar with the techniques of PCA
- Gaining experience analyzing experimental results

Each of the phases of the project had its own deliverables. A detailed project assignment document and a set of images for the five vowels can be found at web-sites http://cs.hiram.edu/~walkerel/RASLUPCA.pdf and http://cs.hiram.edu/~walkerel/cs386/letter images.zip, respectively. Phases 0 and 1 were individual assignments, and the remainder of the assignment was done in groups.

Phase 0: Introduction to Classification

The first introduction to project-related ideas was a lecture on classification in general, training and testing methods, and the importance of finding good features. This was followed up by a paper-and-pencil classification activity, which was phase 0 of the project. The activity required students to experiment with a simple set of features for classifying hand-drawn letters of the alphabet, and to evaluate the feature set and suggest improvements.

```
#Function to compute an average image given a
#list of file names
function [avgImg] = avg_imgs(imgNames)
  #read in all the images, add them up into avgImg
  #and add each image as a column in bigImg
  avgImg = double(imread(imgNames{1}));
  for i = 2:length(imgNames)
    newImg = double(imread(imgNames{i}));
    avgImg = avgImg + newImg;
  endfor
  #divide the image sum by the number of images
  avgImg = avgImg / length(imgNames);
  #display the result
  imshow(uint8(avgImg));
```

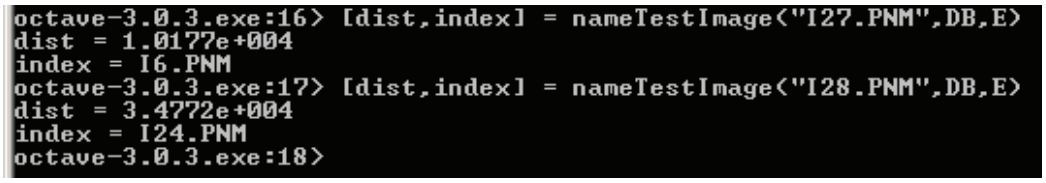
Figure 3: Reading and Averaging Images

Phase 1: Image Representation and GNU Octave

Phase 1 of the project introduced images, their representation, and the programming environment that would be used for all image and matrix manipulation. For this phase, each student had to perform some simple image manipulations, such as reading, averaging and displaying images using GNU Octave (Eaton 2006), a free tool that is compatible with Matlab. This phase was done individually to ensure that all students were familiar with the Octave tool. Figure 3 shows a function that a student might write during this phase.

Most engineering schools use Matlab throughout their curriculum, and it becomes a tool that students use throughout their careers. Computer science programs housed in liberal arts schools are less likely to have this tool available, due to its cost. The GNU Octave tool allows students to perform numerical computations through a command-line interface, as well as to write functions of their own. These functions are compatible with Matlab. Along with another piece of software, ImageMagick (ImageMagick Studio, 2009), GNU Octave allows students to read and write images, converting them into 2D matrices. GNU Octave also provides linear algebra functionality; most importantly for our purposes, it includes functions for the extraction of eigenvectors and eigenvalues from 2D matrices.

While students were working on phase 1, class time was spent introducing the necessary mathematical background for the next phase (matrix multiplication, statistical variance and co-variance, and an intuitive explanation of eigenvectors and eigenvalues). Additionally, students read and discussed two web tutorials on PCA (Smith, 2002; Shlens, 2005). By the time students had completed phase 1, they had the information that they needed to begin on phase 2.



```
octave-3.0.3.exe:16> [dist,index] = nameTestImage("I27.PNM",DB,E)
dist = 1.0177e+004
index = I6.PNM
octave-3.0.3.exe:17> [dist,index] = nameTestImage("I28.PNM",DB,E)
dist = 3.4772e+004
index = I24.PNM
octave-3.0.3.exe:18>
```

Figure 4: Classification Result

Phase 2: Feature Extraction

In phase 2, students worked in small groups of 2 or 3 to implement the PCA method described by Kirby and Sirovich (1990) for feature extraction for face recognition as a sequence of Octave functions. This included finding the average image and normalizing all data, computing the covariance matrix and its eigenvectors, sorting the eigenvectors by eigenvalue, and choosing a subset of eigenvectors for their features.

Phase 2 was very challenging for students, since it required them to use unfamiliar techniques and an unfamiliar tool. Students were strongly encouraged to display and verify their results at each step of the process, and these visualizations helped them understand the project, as well as debug their functions. At the end of this phase, each team had a set of features to carry forward into phase 3.

Phase 3: Letter Recognition

The letter recognition phase required each team to develop a database of features from the training examples and write functions to extract features from a new image and find the nearest training example in the feature space. The result is correct if the training example and the test example are images of the same letter.

Figure 3 shows a screenshot of a portion of one team's results. The first example shows that test image I27 was closest to training image I16, and that test image I28 was closest to training image I24. Because each training and test sample was named with its letter e.g. A2, I27), a successful recognition retrieves an image with the same initial letter in its name. In both cases in the example screenshot, the unknown sign was correctly classified as the letter I.

Due to time pressure, the feature vectors were stored in an array and all teams used a simple sequential search to find the closest image in feature space. Given more time, this phase could involve developing more efficient search structures, or possibly doing some feature clustering to limit the number of comparisons.

Phase 4: Evaluation and Reporting

In the final phase of the project, student teams trained their systems on data subsets of different sizes, then tested on the remaining data. Each group produced a report describing all the functions that they had written, the decisions that they made in choosing features (e.g. how many features to choose based on the sequence of eigenvalues), and the results of their testing. Students chose from three to ten eigenvalues, and all teams achieved greater than 80% success on test data.

The rigor of reporting was somewhat disappointing, mainly due to insufficient instruction, but also because of time limitations. In the future, n-fold cross-validation (Devroye, et. al., 1996) will be taught and required. Using the same measure of success, teams can easily see how their results compare to their classmates'. This comparison will be enhanced by requiring in-class oral presentations for all students, for which time was not allotted this year.

Evaluation

In Spring 2009, twelve students, comprising six teams, completed the project. Every student filled out a post-project survey asking about their attitudes regarding the project, artificial intelligence, and machine learning as a result of the project. In addition, we analyzed specific questions from the midterm and final exams to gauge student learning related to the objectives of this project.

Generally, the survey results were very positive. In reporting details of the survey results, the number listed in parentheses is the total number of students who selected "Agree" or "Strongly Agree" for each statements. There were 12 total surveys, and all questions were answered on each.

Students seemed to generally like the project. They felt the student project was interesting to work on (10), that the level of difficulty was appropriate (10) and (surprisingly, considering the grumbling) that it took a reasonable amount of time to complete (10). They also self-reported that the project contributed to their understanding of course material (9), and that it was an effective way to introduce machine learning concepts (11). Finally, students reported that they had a positive experience in the course (10).

With regard to content, the students said that after the course, they had a good understanding of the fundamental concepts of Artificial Intelligence (12) and of Machine Learning (10). They believed that the problem solving techniques that were covered in the course were valuable (11) and that they had a firm grip of these techniques (10). They are confident that they can identify opportunities to apply the techniques (9) and that they could apply them to different problems (9).

To determine how well students learned the portion of the course about machine learning, we examined their second midterm exam, of which 50 points were from questions directly related to learning. One question (15 points) specifically required students to apply what they had learned in the project giving them a set of data, and asking qualitative questions about variance, covariance, and nearest neighbor classification. The median score of the 12 students on this problem was 12 / 15, and 4 of the 12 students received a perfect score of 15.

The other three machine learning questions asked about decision trees, neural networks, and version space learning. Considering all four questions together (50 total points), the median machine-learning subscore was 39.5 / 50, or 79%. This compares to an overall test median of 65.5%. Although our sample is too small to make any statistical claims, this result is suggestive that the project was successful in helping students learn the machine learning topics, even those that were not directly utilized in the project.

We also looked at the final exam of the last Artificial Intelligence course taught before the machine learning project was added. In this class, machine learning topics (decision trees and neural networks) were covered in a 20-point question on the final exam, taken by 14 students. The median score on this question was 8 / 20, or only 40%. Again, we cannot draw any statistically significant conclusions, as the question itself was different and the student populations were not matched, but our result is suggestive that the class with the machine learning project retained more knowledge about machine learning than the class where no machine learning project was given.

Finally, we can report an anecdotal experience that shows the value of the project for one student who was in the class. This student participated in an undergraduate summer research experience at an engineering school. Part of his project was to adapt and modify Matlab code. He was able to hit the ground running, because "Matlab is just like Octave." Writing in more detail about his project, he indicated that he understood what he had done in class, and how it could be applied in the future: "Anyway, this project reminded me of the image recognition project we did in AI and I was thinking of using the same approach for our needs. … Taking into consideration that we identify the most important features of say, [sound X vs. sound Y], by finding those with higher eigenvalues, would it be possible to identify the source of those features in the sound rather than evaluating and sorting ALL of the features present, thus speeding up the process?"

The student went on to comment about the need to isolate the relevant part of the sound (similar to cropping the image), and offered relevant considerations about the processing time. The student's ability to jump into a new task, recognize the general pattern of classification, and his appropriate consideration of adaptations of the method he had learned in his project, indicates that the project has succeeded in one of its primary goals to prepare students to appropriately use machine learning techniques for novel problems.

Conclusion

We have developed a student project for the introductory artificial intelligence course where students apply Principal Component Analysis (PCA) to the problem of recognizing American Sign Language letters from images. Using the GNU Octave tool, students normalize the images, compute the covariance matrix, extract eigenvectors and rank them by their eigenvalues, choose a subset of the eigenvectors as features, and then use these features to classify novel images.

The project was piloted in a class of 12 students in Spring 2009. Surveys indicated that students were engaged in the activity, and that they felt the project was an effective introduction to machine learning concepts. Students also self-reported that they had a good grasp of the concepts of machine learning, and this was borne out in their exam results.

Acknowledgements

This project was inspired by a capstone project in computer vision performed by Art Geigel III, who also provided the training and test images used in the project. The helpful comments made by anonymous reviewers have improved this paper and are appreciated.

This work was supported by National Science Foundation grant *DUE-.0716338*

References

Devroye, L, Gyorfi, L, and Lugosi, G. 1996. *A Probabilistic Theory of Pattern Recognition*, New York, NY: Springer.

Eaton, John W. 2006. *GNU Octave*. http://www.gnu.org/software/octave/.

ImageMagick Studio, LLC. 2009. *ImageMagick: Convert, Edit, and Compose Images*. http://www.imagemagick.org/script/index.php.

Kirby, M. and Sirovich, L. 1990. Application of the Karhunen-Loeve Procedure for the Characterization of Human Faces. *IEEE Trans. Pattern Anal. Mach. Intell.* 12(1): 103-108.

Markov, Z., Russell, I., Neller, T., Coleman, S. 2005. Enhancing Undergraduate AI Courses Through Machine Learning Projects. In *Proceedings of the 35th ASEE/IEEE Frontiers in Education Conference*. Piscataway, NJ: IEEE Press.

Rich, L., Perry, H., and Guzdial, M. 2004. A CS1 course designed to address interests of women. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*. 190-194. New York, NY: ACM.

Russell, I, Markov, Z. and Coleman, S. 2007. Project MLExAI: Applying Machine Learning to Web Document Classification. *Journal of Computing Sciences in Colleges* 23(2): 226-235.

Russell, I, and Markov, Z. 2009. A Multi-Institutional Project-Centric Framework For Teaching AI Concepts. In *Proceedings of the 39th ASEE/IEEE Frontiers in Education Conference*. Piscataway, NJ: IEEE Press.

Russell, S., and Norvig, P. 2003. *Artificial Intelligence A Modern Approach 2nd ed* New York, NY: Prentice Hall.

Sebe, N., Cohen, I., Garg, A., Huang, T.S. 2005. *Machine Learning in Computer Vision*. Secaucus, NJ: Spring-Verlag New York, Inc.

Shlens, J. 2005. *A Tutorial on Principal Component Analysis*. http://www.snl.salk.edu/~shlens/pub/notes/pca.pdf.

Sloan, R. H. and Troy, P. 2008. CS 0.5: a better approach to introductory computer science for majors. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, 271-275. New York, NY: ACM.

Smith, L. 2002. *A Tutorial on Principal Components Analysis*. http://www.cs.otago.ac.nz/cosc453/student tutorials/princi pal components.pdf