

10-701 Machine Learning, Fall 2011: Homework 5

Due 12/5 at the beginning of class.

November 17, 2011

1 Principal Component Analysis: Eigenfaces [25 points, Bin]

PCA is applied to recognize faces by Matthew Turk and Alex Pentland in their “Eigenface” approach, which is considered as the first successful example of face recognition. In this question, you will implement your own face recognition system.

Please print out your codes and attach them at the end of your answer sheet for this question.

1.1 ORL Face Data

The face data set used in this question is the ORL database of faces¹. There are 4 files in HW4_data.zip:

- **X_train.csv**: each of the 360 lines contains data for a 112×92 face image. If you are using MATLAB, you can load the images using `'data = csvread('X_train.csv')'`, and display the i -th image using `'colormap(gray); imagesc(reshape(data(i,:),112,92))'`.
- **Y_train.csv**: each of the 360 lines contains the label for the corresponding line in X_train.csv
- **X_test.csv**: 40 test images organized in the same way as in X_train.csv
- **Y_test.csv**: labels for face images in X_test.csv

1.2 Calculating Eigenfaces

This step will only use X_train.csv:

- **[3 points]** First compute the mean Ψ of the 360 training faces, and subtract the mean from each training data point.
- **[7 points]** Compute the covariance matrix for the 360 training faces, then apply eigen-decomposition on the obtained covariance matrix. In order to save computation time, you could use `'eigs'` in MATLAB to compute only the eigenvectors corresponding to the 50 largest eigenvalues.
- **[5 points]** Visualize the Eigenfaces: display the eigenvectors corresponding to the 10 largest eigenvalues as 112×92 images. Print out your results.

¹<http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>

- [5 points] Face reconstruction. Plot 2 lines of images in your write-up: (1) (first line) the face images corresponding to lines 1, 5, 20, 30, 40 in X_train.csv; (2) (second line) for faces in lines 1, 5, 20, 30, 40 of X_train.csv, plot the face image obtained by projecting the original face into the subspace spanned by the 50 eigenvectors you computed in the second step. To do the projection for a new data point x , first compute $u_i = x^T V_i$ where V_i is the eigenvector you computed, then your reconstructed data point can be obtained as $x' = \sum_i u_i V_i$.

1.3 Using Eigenfaces to Classify a Face Image

Now let's put the eigenfaces to work in face recognition. First subtract each data point in X_train.csv and X_test.csv by the mean Ψ you computed on X_train.csv. Then project each data point in X_train.csv and X_test.csv into the subspace spanned by the 50 eigenvectors you computed previously. Now your face data lives in a 50-dimensional space, instead of the original 112×92 -dimensional space.

- [5 points] The faces in the data set correspond to 40 different people. Apply 1-Nearest neighbor classifier in this 50-dimensional space to predict labels for the faces in X_test.csv. Report your classification accuracy.

2 Topic Models [Qirong Ho, 25 points]

Mixture models are an active area of Machine Learning research, with many extensions proposed over the years. For instance, in the previous homework we explored how to extend a K -Gaussians mixture model to an infinite Gaussian mixture model. In this question, we are going to explore the “mixtures-of-mixtures” concept, which forms the basis of a topic model.

2.1 K -bag-of-words Mixture Model

Before we talk about topic models, we first need to introduce the K -bag-of-words mixture model, which is used to model words in text documents. The K -bag-of-words model begins with K multinomial distributions (the bags of words). Each represents a probability distribution over words from some vocabulary of length V . Their parameter vectors are $\beta_1, \dots, \beta_K$, where each β_k is a non-negative V -dimensional vector summing to 1.

Next, we have N documents numbered $1, \dots, N$, where document i contains M_i words. Note that each document can have a different number of words, and we denote the j -th word of document i by $w_{ij} \in \{1, \dots, V\}$ (we use integers to represent words). The words are modeled as follows: for each document i , we draw a mixture indicator $t_i \in \{1, \dots, K\}$ from a prior π . This indicator t_i tells us which multinomial generates the words in document i . Finally, we draw each word w_{ij} from the multinomial parameter β_{t_i} , where the draws are made independently (i.e. we don't care about word order). This gives rise to the following generative process:

$$\begin{aligned} t_i &\sim \text{Multinomial}(\pi) && \text{for } i \in \{1, \dots, N\} \\ w_{ij} &\sim \text{Multinomial}(\beta_{t_i}) && \text{for } i \in \{1, \dots, N\} \text{ and } j \in \{1, \dots, M_i\}. \end{aligned}$$

Notice that this is similar to the K -multinomials mixture model, except that some of the observed data (the words w_{ij}) share the same mixture indicator t_i .

In all your answers, please use superscripts to denote vector indices. For example, π^k denotes the k -th element of π , and t_i^k denotes the k -th element of t_i .

1. [2 points] Using the definition of the multinomial distribution, explicitly write out $P(t_i | \pi)$.
2. [3 points] Use conditional independence to simplify the expression $P(w_{ij} | t, \beta, \pi)$ as much as possible. In other words, derive an expression $P(w_{ij} | t, \beta, \pi) = P(w_{ij} | \dots)$ where the ' $\dots$ ' is some subset of $\{t, \beta, \pi\}$. The symbols t and β without subscripts represent all $t_1, \dots, t_N$ and $\beta_1, \dots, \beta_K$ respectively. **Hint: try drawing the model as a Bayes net (no need to show this). Your final answer should depend on every single $\beta_1, \dots, \beta_K$.**
3. [2 points] Using the definition of the multinomial distribution, explicitly write out your simplified probability statement from part 2.

2.2 Topic Model with K topics

To develop a topic model², we need to make two changes. First, instead of letting document i take on a single topic t_i , let's allow it to have a *mixture* over topics θ_i , where θ_i is a non-negative K -dimensional vector summing to 1. Think of θ_i as a probability distribution over the K topics represented by $\beta_1, \dots, \beta_K$. The appropriate prior distribution for θ_i is a *Dirichlet distribution*, which will be described shortly.

Second, we now introduce a *topic indicator* z_{ij} for each word w_{ij} , which determines word w_{ij} 's topic. Notice how this differs from the K -bag-of-words model: we're now allowing each word to have its own topic, instead of restricting it to follow the document's topic t_i . Naturally, we shall draw z_{ij} from document i 's topic distribution θ_i .

These two changes give rise to the following generative process:

$$\begin{aligned} \theta_i &\sim \text{Dirichlet}(\alpha) && \text{for } i \in \{1, \dots, N\} \\ z_{ij} &\sim \text{Multinomial}(\theta_i) && \text{for } i \in \{1, \dots, N\} \text{ and } j \in \{1, \dots, M_i\} \\ w_{ij} &\sim \text{Multinomial}(\beta_{z_{ij}}) && \text{for } i \in \{1, \dots, N\} \text{ and } j \in \{1, \dots, M_i\}. \end{aligned}$$

$\alpha > 0$ is a scalar parameter for the (symmetric) Dirichlet distribution, defined as

$$P(\theta_i | \alpha) = \frac{[\Gamma(\alpha)]^K}{\Gamma(K\alpha)} \prod_{k=1}^K (\theta_i^k)^{\alpha-1}$$

where $\Gamma(\alpha)$ is the Gamma function³. Pay attention to how this model is a "mixture of mixtures": each θ_i represents a mixture over topic vocabularies $\beta_1, \dots, \beta_K$, and there are N such mixtures $\theta_1, \dots, \theta_N$, that together constitute the mixture of mixtures.

1. [2 points] Use conditional independence to simplify the expression $P(z_{ij} | \theta, \alpha, \beta)$ as much as possible. The symbols θ and β without subscripts represent all $\theta_1, \dots, \theta_N$ and $\beta_1, \dots, \beta_K$ respectively.
2. [2 points] Using the definition of the multinomial distribution, explicitly write out your simplified probability statement from part 1.
3. [2 points] Use conditional independence to simplify the expression $P(w_{ij} | z, \theta, \alpha, \beta)$ as much as possible.
4. [2 points] Using the definition of the multinomial distribution, explicitly write out your simplified probability statement from part 3.

²For more information, refer to *Latent Dirichlet Allocation* (Blei et al., 2003).

³http://en.wikipedia.org/wiki/Gamma_function

2.3 Interpreting Topic Models

The topic model, like the K -bag-of-words mixture model, is a *latent variable* model: some of the variables are unobserved, and we are interested in finding their values. For the K -bag-of-words model, we are interested in finding the hidden document topics t_i . For the topic model, we are mostly interested in the hidden document topic distributions θ_i (and to some extent the word topics z_{ij}).

1. **[2 points]** Both t_i from the K -BoW model and θ_i from the topic model say something about document i 's topical content. In one sentence, state the main difference between t_i, θ_i .
2. **[3 points]** Discuss the implications of your answer to part 1. How is topic modeling more useful than K -BoW? Your answer should be no more than a few sentences.
3. **[3 points]** In both K -BoW and topic models, the β_k parameters represent vocabularies for each topic k . We didn't talk about learning the values of β , but it turns out that the common learning strategies (Gibbs sampling or Variational EM) will sometimes produce topics that share words — in other words, $\beta_k^v > 0$ and $\beta_\ell^v > 0$ for some topic k and some topic ℓ . Why is this word sharing useful? Again, keep your answer brief.
4. **[2 points]** PCA (Principal Component Analysis) can also be used to learn “topics” from a set of documents. Give at least two differences between PCA and topic models. You don't have to explain the differences, just list them.

3 AdaBoost [Nan Li, 25 pt]

In this exercise, we consider properties of the AdaBoost algorithm.

3.1 True or False [6 pts]

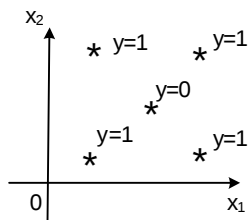
Please answer whether the following statement is true or not, and briefly explain why.

- (a) [3 pt] Training error never increases as the number of rounds increases.
- (b) [3 pt] With sufficiently many iterations, training error will always decrease to a value that is arbitrarily close to zero starting with any classifier.

3.2 Boosting [19 pts]

There is an interesting applet written by Yoav Freund (<http://cseweb.ucsd.edu/~yfreund/adaboost/>). With it, you can create your own data set and train AdaBoost models. For each of the question below, you can print a screen shot which includes both data points and error curves.

- (a) [5 pts] As shown in the figure below, we have five training samples with label 0 or 1.
Please plot these training samples in the applet given above, and see that with the boosting algorithm get zero training error after sufficient iterations.
Note: The applet also requires input of test samples. You may have to add some more samples as test samples as well. We are not concerned about the testing errors in this example. Hence, for the screen shot, please only plot the training set.



- (b) [5 pts] If the base function is are linear classifiers (i.e. $x_i < a$ or $x_i > a$). What is the minimum number of iterations before it can reach zero training error?
- (c) [5 pts] Please design a dataset showing that AdaBoost does overfit. You can print a screen shot which includes both data points and error curves.
- Note:** Please plot both the training and test samples.
- (d) [4 pts] Can you think of a strategy to prevent Boosting from overfitting?