Assignment 3
IFT 6085, Winter 2013

Date posted: March 12, 2013
Due: March 19, 2013, at the beginning of class

---

1. (3/20) An $n \times n$ circulant matrix takes the form

$$C = \begin{pmatrix} c_0 & c_{n-1} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & & c_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ c_{n-1} & c_{n-2} & \cdots & c_1 & c_0 \end{pmatrix}$$

   Show that all circulant matrices have (time-discrete) phasors as eigenvectors.

2. (7/20) Consider a *conditional* K-means model for learning motion from videos: Like the K-means model we had defined in class, we represent an image, $\boldsymbol{x}$, using the nearest prototype $\boldsymbol{w}_k$. But now we define the prototype vector for hidden unit k as a *linear function* of some other image $\boldsymbol{c}$:

$$\boldsymbol{w}_k(\boldsymbol{c}) = W_k \boldsymbol{c}$$

   Thus, the model parameters are now matrices W_k not vectors $\boldsymbol{w}_k$. Think of $\boldsymbol{c}$ as a conditioning, or "context" image. If the training data consists of pairs $(\boldsymbol{c}, \boldsymbol{x})$, we can define the cost function in analogy to standard K-means as the average (conditional) squared distance between $\boldsymbol{x}$ and the nearest conditional prototype $W_k \boldsymbol{c}$. In other words, the cost incurred by training example $(\boldsymbol{c}, \boldsymbol{x})$ may be written

$$E(W_1, \ldots, W_K) = \frac{1}{2} \|(\boldsymbol{x} - \boldsymbol{w}_{s(\boldsymbol{x},\boldsymbol{c})})\|^2 = \frac{1}{2} \|(\boldsymbol{x} - W_{s(\boldsymbol{x},\boldsymbol{c})} \boldsymbol{c})\|^2$$

   (Note that the winner-takes-all selection function now also becomes dependent on $\boldsymbol{c}$.)

   Suggest an online learning rule for this model. To this end, first write down the gradient of the cost function with respect to the parameters. Then briefly discuss the relationship of this model with linear regression.

   What to hand in:

   - Gradient of the cost.
   - Learning rule.

- Your brief discussion of the relationship with linear regression.

3. (10/20) In this question, you will implement a simple recognition pipeline based on your online Hebbian K-means algorithm and the "single layers" paper we discussed in class.

   Download the files:

   ```
   http://www.iro.umontreal.ca/~memisevr/teaching/mlvis2013/cifar_mini_images_train.txt
   http://www.iro.umontreal.ca/~memisevr/teaching/mlvis2013/cifar_mini_labels_train.txt
   http://www.iro.umontreal.ca/~memisevr/teaching/mlvis2013/cifar_mini_images_test.txt
   http://www.iro.umontreal.ca/~memisevr/teaching/mlvis2013/cifar_mini_labels_test.txt
   ```

   and save them locally. (The first file is the same as in assignment 2.)

   The image files, like in assignment 2, contain one RGB image of size $32 \times 32 \times 3$ per line (represented as 3072 integers separated by white space). They are subsets of the CIFAR dataset. The label files contain labels in one-hot encoding.

   Use the training set to learn features exactly like in assignment 2: Crop multiple random patches from each of the 2000 *training images* (only training not test); perform canonical pre-processing and PCA whitening; and learn K-means features with $K = 100$.

   Implement the "triangle" k-means assignment function defined in the paper "An analysis of single-layer networks in unsupervised feature learning" we discussed in class.

   To define a classifier that takes CIFAR color images ($32 \times 32 \times 3$) as the input, and that outputs class labels, follow these steps:

   (a) For each image (train and test), crop patches from all positions within the boundaries of the image. (You should get 400 patches per image).

   (b) Pre-process the patches (canonical + whitening) using the statistics and PCA whitening matrix you estimated from the training data.

   (c) Extract features from the patches using your implementation of the triangle K-means assignment function.

   (d) For each image, compute the mean feature vector for those features that came *from the same quadrant* in the image. This will give you a 100-dimensional vector for each quadrant (and thus four 100-dimensional feature vectors in total per image).

   (e) For each image, concatenate the four feature vectors to get a $400-$dimensional vector (one per image).

   (f) Train a regularized *logistic regression* classifier (see instructions below) to classify this representation into the 10 CIFAR classes. Train your classifier on the *training data* only.

   Hints:

   - Do pre-processing and PCA whitening exactly like in assignment 2.

- Refer to assignment 2 if you do not remember the right way of reshaping the RGB images or for other pre-processing hints.
- If you prefer (and have access to a fast machine), you may use the full CIFAR-10 training set for training and the full testing set for testing:

  `http://www.cs.toronto.edu/~kriz/cifar.html`

  Everything else (preprocessing, etc.) must be exactly the same.
- You may use the logistic regression code at

  `http://www.iro.umontreal.ca/~memisevr/teaching/mlvis2013/logreg.py`

  or use your own logistic regression code.
- Choose the regularization ("weightcost") parameter on a subset containing 500 training cases (10000 if you use the full CIFAR set). Then retrain with the optimal weightcost on the whole training set.

What to hand in:

- An image showing three training examples from each class.
- A plot of the 400-dimensional mean feature vector, averaged over the training data.
- A display of your 100 K-means filters, sorted according to their *average value of the triangle-activation* (where the average is over the training data and the four quadrants in each image).
- Your classification performance on the test data, and the optimal weightcost from the validation set.
- Do not forget to mention which dataset you used (mini CIFAR or the full CIFAR dataset?).
- Do *not* hand in any program code.