Assignment 3
IFT 6268, Fall 2013

Date posted: November 6, 2013
Due: November 13, 2013, at the beginning of class

---

1. (5/10) Answer all of the following questions:

   - Explain why whitening helps learning an ICA model.
   - Describe what effect maximum likelihood learning of an ICA model has on the joint entropy of the vector of hidden variables and on the marginal entropies of the individual hiddens.
   - Write down the distribution over hidden units in a binary RBM. Explain in what way this distribution facilitates Gibbs sampling.
   - Explain what K-means clustering has to do with lateral inhibition.
   - Write down the energy function (scalar potential) of an autoencoder with sigmoid hidden units (and real valued outputs). Show that the gradient of the energy function with respect to $\boldsymbol{x}$ is $r(\boldsymbol{x}) - \boldsymbol{x}$.

2. (5/10) In this question, we will extend the recognition pipeline from Assignment 2 to use (i) color images, (ii) proper feature learning, (iii) local pooling, and (iv) proper validation over the classifier hyperparameter. Download the files:

   ```
   www.iro.umontreal.ca/~memisevr/teaching/ift6268_2013/cifarmini_color_images_train.txt
   www.iro.umontreal.ca/~memisevr/teaching/ift6268_2013/cifarmini_labels_train.txt
   www.iro.umontreal.ca/~memisevr/teaching/ift6268_2013/cifarmini_color_images_test.txt
   www.iro.umontreal.ca/~memisevr/teaching/ift6268_2013/cifarmini_labels_test.txt
   ```

   and save them locally. They are color versions of the files we used in the previous assignments. More specifically, each image file contains one RGB image of size $32 \times 32 \times 3$ per line (represented as 3072 integers separated by white space).

   To learn a recognition model from these images, follow these steps (this pipeline is related to, but not identical to, the pipeline you used in Assignment 2):

   (a) For each image (train and test), crop color patches of size 10×10 pixels from all positions within the boundaries of the image. (You should get 484 color patches per image, or $484 \times 4000 = 1936000$ color patches in total).

   (b) DC center and contrast normalize each 300-dimensional vectorized color patch.

(c) Compute the PCA whitening matrix from a subset of the normalized *training* patches, retaining 90% of the variance.

(d) Whiten all normalized patches using your PCA whitening matrix. Do not use ZCA just PCA.

(e) From a subset of your whitened training patches, learn a matrix W, containing 100 "prototypes", using a feature learning algorithm of your choice. After training, each column of W will contain one prototype vector $\boldsymbol{w}_k, k = 1, \ldots, 100$. It is your responsibility to choose a model that gives you sensible features.

(f) Transform *all* of your patches $\boldsymbol{x}$ into a feature vector whose i^{th} response is

$$\max(0, \boldsymbol{w}_i^{\mathrm{T}} \boldsymbol{x})$$

(g) For each image, compute the mean feature vector for all 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 per image in total).

(h) For each image, concatenate the four feature vectors to get a 400-dimensional descriptor per image.

(i) Train a *regularized* logistic regression classifier on this data. Choose the regularization ("weightcost") parameter on a subset of 200 training cases (the validation set), and use the remaining 1800 training cases for training the model (with various choices of weightcost). After picking the winning weightcost on the validation data, retrain with the optimal weightcost on the whole training set, and report your classification error rate on the test data. Make sure not to use any test data for training or validation!

What to hand in:

- A color image showing some of the training patches (this should help you debug any dimension shuffling and reshpaing that you will have to perform).
- A color image showing some of the same training patches *after whitening*.
- A color image showing those PCA whitening filters that you kept in order to retain 90% of the variance.
- One color image showing the 100 learned encoder weights, and one color image showing the 100 learned decoder weights. Show them in terms of the original image space. This means that you have to "undo" any intermediate whitening or de-whitening operations to show them.
- A list of training and validation errors along with the corresponding weightcosts.
- Your classification performance on the train and test data, after retraining with the winning weightcost on the whole training set.
- Do *not* hand in any program code.

Hints:

- To display multiple color images that are contained in a 4D-"tensor", you may use the function defined in the module

  `http://www.iro.umontreal.ca/~memisevr/teaching/ift6268_2013/dispims_color.py`

  For example,

  ```
  dispims_color.dispims_color(patches[:100])
  ```

  would display color patches of size 32×32 pixels, contained an array with dimensions $(100, 32, 32, 3)$.
- For debugging and setting up the pipeline it may be convenient to initially define prototype vectors as a selection of training patches, like we did in Assignment 2.
- This may also be a good way to initialize your features for learning.