Exercises
Computational Intelligence Lab
SS 2010

Machine Learning Laboratory
Dept. of Computer Science, ETH Zürich
Prof. Dr. Joachim M. Buhmann
Web http://ml2.inf.ethz.ch/courses/cil

Series Feb 24th, 2011 (Efficient Matlab Programming)

Introduction

Matlab is an interpreted language with an expressive syntax and extensive runtime checks. Unfortunately, this results in a significant overhead in comparison to compiled low-level languages like C. It is therefore computationally beneficial to use vectorized commands instead of explicit `for` loops, because the number of function calls is reduced, and the vectorized commands are pre-compiled C code that executes fast. In this exercise, you are asked to write efficient implementations for three small problems that are typical for the field of machine learning.
By submitting your implementation to the course's submission system, you get an immediate feedback on the relative efficiency of your code as well as gain familiarity with the system.

Useful Commands

We list here some commands that prove useful for writing vectorized code. You can read the full documentation and examples by issuing `doc <command>` at the Matlab prompt.

- `X.*Y, X./Y`: elementwise multiplication and division of matrices $\mathbf{X}$ and $\mathbf{Y}$
- `max(X)`: find the maximum element and its index for each column of matrix $\mathbf{X}$
- `mean(X), std(X)`: compute the mean and standard deviation of each column of matrix $\mathbf{X}$
- `meshgrid(x,y)`: replicate vectors $\mathbf{x}$ and $\mathbf{y}$ along different dimensions
- `repmat(X, n, m)`: replicate matrix $\mathbf{X}$ n-times along the first, and m-times along the second dimension
- `size(X,k)`: return the size of matrix $\mathbf{X}$ along dimension k
- `sum(X,k)`: sum the elements of matrix $\mathbf{X}$ along dimension k
- `tic, toc`: start and stop a timer to measure execution time

Code Submission

1. Register your group in the submission system. Please note that all the information entered into the system at this stage, including the users details, will be deleted in the end of the exercise.
2. Download the folder named `submission-stub` from from the course homepage. For each of the following tasks, implement a Matlab function and replace the corresponding stub call with your own implementation. When you are ready to submit, zip the entire directory and submit it. You can always resubmit your code if you have made improvements after submitting.
3. The implementations are tested on randomly generated data. You can download the data generation scripts as `data_gen.zip` from the course homepage to test your implementation before submission.
4. Only valid submissions will be ranked and appear on the webpage. Please make sure to have **all** the file stubs in the submitted directory (with the correct function names)

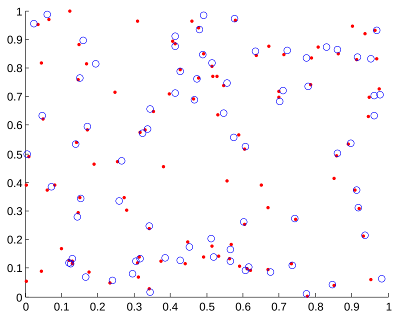

Figure 1: Two sets of points in the plane. The circles are a subset of the dots and have been perturbed randomly.

Task A: Matrix Standardization

The different dimensions or features of a data sample often show different variances. For some subsequent operations, it is a beneficial preprocessing step to standardize the data, i.e. subtract the mean and divide by the standard deviation for each dimension. After this processing, each dimension has zero mean and unit variance. Note that this is not equivalent to data whitening, which additionally decorrelates the dimensions (by means of a coordinate rotation).

Write a function that accepts data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ as input and outputs the same data after normalization. n is the number of samples, and d the number of dimensions, i.e. rows contain samples and columns features.

Function signature: `Y = stdize(X)`

Task B: Pairwise Distances in the Plane

One application of machine learning to computer vision is interest point tracking. The location of corners in an image is tracked along subsequent frames of a video signal (see figure 1 for a synthetic example). In this context, one is often interested in the pairwise distance of all points in the first frame to all points in the second frame. Matching points according to minimal distance is a simple heuristic that works well if many interest points are found in both frames and perturbations are small.

Write a function that accepts two matrices $\mathbf{P} \in \mathbb{R}^{p \times 2}, \mathbf{Q} \in \mathbb{R}^{q \times 2}$ as input, where each row contains the (x, y) coordinates of an interest point. Note that the number of points (p and q) do not have to be equal. As output, compute the pairwise distances of all points in $\mathbf{P}$ to all points in $\mathbf{Q}$ and collect them in matrix $\mathbf{D}$. Element $D_{i,j}$ is the Euclidean distance of the i-th point in $\mathbf{P}$ to the j-th point in $\mathbf{Q}$.

Function signature: `D = pairdist(P, Q)`

Task C: Likelihood of a Data Sample

A subtask of many machine learning algorithms is to compute the likelihood $p(\mathbf{x}|\boldsymbol{\theta})$ of a sample $\mathbf{x}$ for a given density model with parameters $\boldsymbol{\theta}$. Given k models, we now want to assign $\mathbf{x}_i$ to the model for which the likelihood is maximal: $a_i = \arg\max_m p(\mathbf{x}_i|\boldsymbol{\theta}_m)$, where $m = 1, \ldots, k$. $\boldsymbol{\theta}_m$ are the parameters of the m-th density model.

We ask you to implement the assignment step for the two model case, i.e. $k = 2$. As input, your function receives a data matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$ (where $\mathbf{x}_i$ is the i-th column of $\mathbf{X}$) and the parameters $(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ of two multivariate Gaussian distributions:

$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right).$$

$|\boldsymbol{\Sigma}|$ is the determinant of $\boldsymbol{\Sigma}$ and $\boldsymbol{\Sigma}^{-1}$ its inverse. Your function returns the assignment vector $\mathbf{a} \in \{1, 2\}^n$, where $a_i = 1$ means that $\mathbf{x}_i$ has been assigned to model 1. Therefore, it holds that $p(\mathbf{x}_i|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) > p(\mathbf{x}_i|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$.

Function signature: `a = lassign(X, mu_1, Sigma_1, mu_2, Sigma_2)`