MACHINE LEARNING COMS 4771, HOMEWORK 4
Assigned March 14, 2013. Due April 4, 2013 before 1:00pm.

Here are the instructions for submitting your homework. Archive/package all of the files you are submitting as a single tarball or zip archive: "UNI-HW4.tar.gz" or "UNI-HW4.zip". For example, a compressed tarball would be "ir2322-HW4.tar.gz". Your homework should contain:

- a writeup (PDF, TXT, or PostScript)
- code (as Matlab M files, shorter code is generally better but include comments)
- any figures/pictures not included in the writeup (PDF or PostScript)
- if you have special instructions, include them as a plain text file called README.txt.

Submit your homework through CourseWorks by doing the following:

1 Log into https://courseworks.columbia.edu/
2 Click "Assignments" on the left side.
3 Choose the appropriate HW Folder to submit to.
4 Use the filename "yourUNI-HW4.tar.gz" or "yourUNI-HW4.zip".
5 Make sure that the "title" is yourUNI-HW4 (example: zz9999-HW4).
6 Add any special instructions in both the description and the README.txt.
7 Click "Submit" at the bottom to upload your file.
8 If you submit multiple times, only the last submission prior to the deadline will count.
9 If something goes wrong, ask the TAs for help.
10 In a dire emergency, if nothing else works, send your homework to the TAs.

Handwritten writeups are not allowed without prior approval.

All your code should be written in Matlab (other languages may be used only with prior permission from an instructor). Please submit all your souce files, each function in a separate file. Clearly denote what each function does, its inputs and outputs, and to which problem it belongs. Do not resubmit code or data provided to you. Do not submit code written by others. Identical submissions will be detected and both parties will get zero credit. Sample code is available on the Tutorials web page. Datasets are available from the Handouts web page. You may include figures directly in your write-up, or separately and refer to them by filename.

Each homework counts equally towards your grade (other than your worst which will be dropped). Points shown here for each problem indicate relative weights for this specific homework. As always, up to 10% bonus points are available for exceptional, relevant work going beyond what is asked.

1 Problem 1 (30 points)

Kernels: Consider any Mercer kernel defined by $k(x, \tilde{x}) = \boldsymbol{\phi}(x)^\top \boldsymbol{\phi}(\tilde{x})$. We are given a sample $S = \{x_1, x_2, ..., x_n\}$ of n inputs. We can form the Kernel (Gram) matrix $\mathbf{K}$ as an $n \times n$ matrix of kernel evaluations between all pairs of examples i.e., $\mathbf{K}_{i,j} = k(x_i, x_j)$. **Mercer's Theorem** states that a symmetric function $k(.,.)$ is a kernel iff for any finite sample S the kernel matrix $\mathbf{K}$ is positive semi-definite. Recall that a matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ is positive semi-definite iff $\mathbf{c}^\top \mathbf{K} \mathbf{c} \geq 0$ for all real-valued vectors $\mathbf{c} \in \mathbb{R}^n$.

- Prove Mercer's theorem in one direction: for any Mercer kernel $k(.,.)$ and finite sample S, the kernel matrix $\mathbf{K}$ is positive semi-definite.

- Given any two Mercer kernels $k_1(.,.)$ and $k_2(.,.)$, prove that the following are also Mercer kernels:

  1. $k(x, \tilde{x}) = \alpha k_1(x, \tilde{x}) + \beta k_2(x, \tilde{x})$ for $\alpha, \beta \geq 0$
  2. $k(x, \tilde{x}) = k_1(x, \tilde{x}) \times k_2(x, \tilde{x})$
  3. $k(x, \tilde{x}) = f(k_1(x, \tilde{x}))$ where f is any polynomial with positive coefficients
  4. $k(x, \tilde{x}) = \exp(k_1(x, \tilde{x}))$

2 Problem 2 (20 points)

Kernelized Regression: We are given a training set $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ where $\mathbf{x}_i \in \mathbb{R}^d$ is a d-dimensional real vector and $y_i \in \mathbb{R}$ is a real value. Recall the linear regression: $\hat{y} = \mathbf{w}^\top \mathbf{x}$. We will now consider nonlinear (kernel) regression $\hat{y} = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x})$. Assume we will minimize the regularized cost function:

$$L(\mathbf{w}) = \sum_{i=1}^{n} (\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) - y_i)^2 + \lambda \mathbf{w}^\top \mathbf{w}.$$

1. Solve for $\mathbf{w}$ and show that it lives in the span of feature maps: $\mathbf{w} = \sum_{i=1}^{n} \alpha_i \boldsymbol{\phi}(\mathbf{x}_i)$.

2. Express the cost $L(\mathbf{w})$ as a function of $\alpha_1, \ldots, \alpha_n$ and find the values $\alpha_1, \ldots, \alpha_n$ for which the cost is minimized.

3 Problem 3 (15 points)

PCA (Another Perspective): Given input vectors $\{x_1, \cdots, x_T\}$ where $x_i \in \mathbb{R}^n$, the goal of principal components analysis (PCA) is to find a low-dimensional approximation of the data minimizing the *quadratic compression loss*. More formally, we want to find an n-dimensional vector μ and a rank k projection matrix P, where $k \leq n$, such that the following loss function is minimized:

$$\text{comp}(P) = \sum_{t=1}^{T} \left((x_t - \mu) - P(x_t - \mu)\right)^\top \left((x_t - \mu) - P(x_t - \mu)\right)$$

Differentiating and solving for μ gives: $\mu^* = \frac{1}{T} \sum_{t=1}^{T} x_t$ which is the data mean. Show that substituting μ^* to the expression for loss function yields:

$$\text{comp}(P) = \text{tr}(C) - \text{tr}(PC)$$

where C is the covariance matrix and tr is the matrix trace defined as the sum of the diagonal elements of the matrix.

4 Problem 4 (35 points)

K-Means Clustering:

Implement the k-means clustering algorithm.

You will use this code to investigate the stability properties of the k-means algorithm. You will test your k-means code on synthetic data-sets that you generate. Each data-set is of the form $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ where the samples are drawn *iid* from some distribution $p(\mathbf{x})$. You will generate data from a distribution $p(\mathbf{x})$ which is a mixture of Gaussians. You have control over this underlying distribution and can choose the means, covariances and mixing proportions. As an example of how to generate data from a single Gaussian, see the code on the Tutorials link called gendata.m. Modify this code to generate data from a mixture of Gaussians. You will use this code to sample data from mixtures of Gaussians that you design (different means, covariances and mixing weights).

Consider the following conjecture which we will assume is true (its proof is an open problem): *if the true underlying distribution has K_{true} well-separated clusters*[1]*, the k-means algorithm run with $K = K_{true}$ centers is stable.*

We say that an algorithm is *stable* if, for different input data-sets $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_T$ obtained from the same distribution $p(\mathbf{x})$, the resulting outputs the algorithm produces are similar. So, if k-means is stable, the clusterings it outputs for $\mathcal{D}_1, \ldots, \mathcal{D}_T$ should all group the points in a similar way. You will explore the stability of k-means in the following three cases.

- Case $K = K_{true}$. Generate 5 different data-sets from your distribution $p(\mathbf{x})$ with $K_{true} = 4$ mixture components. Initialize k-means with $K = K_{true}$ components by placing center i in the k-means algorithm *inside* the true cluster i in your mixture (for $i = 1, \ldots, K$). After initializing, run the k-means code. Do this for your 5 data-sets and show that k-means correctly preserves one center per cluster in the end (with high probability). Thus, the clustering results are stable across these 5 runs. Show an example of one data set and one clustering (since it is stable).

- Case $K > K_{true}$. Generate 5 different data-sets from your distribution $p(\mathbf{x})$ with $K_{true} = 4$ mixture components. Try *two different* initialization schemes with k-means using $K > K_{true} = 4$ and apply the algorithm to $\mathcal{D}_1, \ldots, \mathcal{D}_5$. Show that your algorithm obtains different clusterings (instability) across these 10 runs. Show only two examples where you obtained different clusterings to highlight the instability.

- Case $K < K_{true}$. Try two different initialization schemes with k-means with $1 < K < K_{true} = 4$ on 5 data-sets for one distribution $p_1(\mathbf{x})$ and on 5 data-sets from another distribution $p_2(\mathbf{x})$. Here, both $p_1(\mathbf{x})$ and $p_2(\mathbf{x})$ have the same number of components in the mixture (both have $K_{true} = 4$). Explore different choices of $p_1(\mathbf{x})$ and $p_2(\mathbf{x})$ so that you obtain *stability* with $p_1(\mathbf{x})$ across its 5 data-sets but you obtain *instability* with $p_2(\mathbf{x})$ across its 5 data-sets. Show one dataset and one clustering with $p_1(\mathbf{x})$ to highlight stability and show two datasets with different clusterings from $p_2(\mathbf{x})$ to highlight its instability.

In the last two cases ($K > K_{true}$ and $K < K_{true}$), you have control over the initialization scheme (you can start the algorithm's initial centers anywhere you want).

[1] Well-separated clusters loosely means that the Gaussians in the mixture are not significantly overlapping.