# CSCI 1950-F Homework 7: Gaussian Processes, Laplace Approximations, & K-Means Clustering

Brown University, Spring 2011

Homework due at 12:00pm on April 19, 2011

The first question explores Gaussian process classification on a dataset of pedestrian images:
http://www.gavrila.net/Research/Pedestrian_Detection/Daimler_Pedestrian_Benchmarks/Daimler_Pedestrian_Class__Benc/daimler_pedestrian_class__benc.html
The data, collected by Daimler-Chrysler, has been preprocessed so that you do not have to deal with images. We will compare two sets of features: raw intensity values (from cropped image windows), and so-called histogram of oriented gradient (HOG) features, which have been hand-engineered to improve robustness. The preprocessed data is available here:

`/course/cs195f/asgn/pedestrian`.

We will compare classifiers based on linear and squared exponential kernels. The linear kernel is defined as

$$k(x_p, x_q) = \frac{1}{\alpha}(1 + x_p^T x_q)$$

where $\alpha$ is a variance hyperparameter. The squared exponential kernel is defined as

$$k(x_p, x_q) = \sigma^2 * exp\left(-\frac{(x_p - x_q)^T * (x_p - x_q)}{2\ell^2}\right)$$

where $\sigma^2$ is the variance hyperparameter, and $\ell$ is the length-scale hyperparameter.

In this homework, we will use the Gaussian process Matlab toolbox (Rasmussen et al.), which is distributed as part of the pmtk3 Matlab toolbox (http://code.google.com/p/pmtk3/). The two covariances of interest are supported by the toolbox through the functions `covLINone` and `covSEiso`. We will use the Laplace approximation to implement binary GP classifiers, which is also supported in the toolbox through the `binaryLaplaceGP` function:

```
[p, mu, s2, nlZ] = binaryLaplaceGP(loghyper, covfunc, lik, x, y, xstar);
where:
 loghyper is a (column) vector of log hyper parameters
 covfunc  is the name of the covariance function - either 'covSEiso' or 'covLINone'.
 lik      is the name of the likelihood function - 'logistic'
 x        is a N by D matrix of N training examples of length D
 y        is a (column) vector (of size N) of binary +1/-1 targets
 xstar    is a M by D matrix of M test examples of length D
 p        is a (column) vector (of length M) of predictive probabilities
 mu       is a (column) vector (of length M) of predictive latent means
 s2       is a (column) vector (of length M) of predictive latent variances
 nlZ      is the returned value of the negative log marginal likelihood
```

We will select the hyperparameters by maximizing the marginal likelihood. A (possibly local) minimum of the negative log marginal likelihood can be found via the `minimize` function from the GP toolbox:

```
[optim_loghyper logmarglik] = ...
   minimize(init_guess, 'binaryLaplaceGP', -25, covfunc, lik, x, y);
where -25 tells minimize to evaluate 'binaryLaplaceGP' 25 times,
and init_guess is an initial guess at the log hyperparameters.
```

For the questions below, the specified initializations are for log-domain representations of the hyperparameters, as assumed by the GP toolbox.

**Question 1:**

*a) Load the* `intensity` *dataset. Use the* `minimize` *function to estimate the linear kernel hyperparameter $\alpha$. Initialize* `minimize` *by setting* `init_guess` *to 15. What is the optimal hyperparameter determined by the marginal likelihood optimization? Using this hyperparameter, train a linear kernel Gaussian process classifier on the* `train` *data. Report the corresponding error rate on* `test` *data.*

*b) Repeat part (a) with* `init_guess` *equaling -5. Do you observe any differences? Explain.*

*c) Repeat parts (a-b) with the* `hog` *dataset.*

*d) Load the* `intensity` *dataset. Use the* `minimize` *function to estimate the squared exponential kernel hyperparameters $\ell$ and $\sigma^2$. Initialize* `minimize` *by setting* `init_guess` *to* $[10; 3]$*. What are the optimal hyperparameters found by the optimization? Using these hyperparameters, train a squared exponential kernel Gaussian process classifier on the* `train` *data. Report the corresponding error rate on the* `test` *data.*

*e) Repeat part (d) with* `init_guess` *equaling* $[1; 3]$*. Do you observe any differences? Explain.*

*f) Repeat parts (d-e) with the* `hog` *dataset.*

*g) For this data, are the results sensitive to the choice of kernel hyperparameters? Which has a bigger impact on performance, the chosen kernel function or the chosen input feature representation?*

The next question investigates Laplace approximations of the Poisson regression model, a generalized linear model previously seen on the midterm. It is used to predict output counts from input features. For example, it could be used to predict the number of users that will read a news article, or the number of people who will contract the flu in a particular city. The probability mass function for a Poisson distribution with mean $\mu > 0$ equals

$$\text{Pois}(y \mid \mu) = \frac{1}{y!} e^{-\mu} \mu^y, \qquad y = 0, 1, 2, 3, \ldots$$

For training example $i$, let $y_i \in \{0, 1, 2, 3, \ldots\}$ denote the output count (a non-negative integer) we want to predict, $x_i$ the input or conditioning variables, and $\phi(x_i) \in \mathbb{R}^m$ a fixed (possibly non-linear) feature function. The Poisson regression model is then

$$p(y_i \mid x_i, w) = \text{Pois}(y_i \mid \mu_i), \qquad \mu_i = \exp(w^T \phi(x_i)) = \exp\left(\sum_{k=1}^{m} w_k \phi_k(x_i)\right).$$

Here, $w \in \mathbb{R}^m$ is a vector of weights associated with the $m$ features. Note how the exponential function maps real-valued regression outputs to positive Poisson mean parameters.

**Question 2:**

*a) Suppose we place a zero mean Gaussian prior distribution on the weight vector:*

$$p(w \mid \alpha) = \text{Normal}(w \mid 0, \alpha^{-1} I_m)$$

*Give an expression for the negative log posterior distribution $-\log p(w \mid x, y)$ of a generic data set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ under the Poisson regression model.*

*b) Derive an expression for the gradient (vector of first derivatives) of the negative log posterior distribution with respect to the weight vector $w$. Simplify your answer.*

*c) Derive an expression for the Hessian (matrix of second derivatives) of the negative log posterior distribution with respect to the weight vector $w$. Simplify your answer.*

*d) Suppose that you use the preceding gradient and Hessian expressions to find the mode $\hat{w}$ of the posterior distribution. Given an expression for the resulting Laplace approximation to the posterior distribution.*

*e) One standard method for choosing an appropriate prior inverse-variance $\alpha$ is to maximize the following marginal log-likelihood:*

$$\log p(y \mid x, \alpha) = \log \int p(y \mid x, w) p(w \mid \alpha)\, dw$$

*Using the Laplace approximation from part (d), determine a corresponding approximation of this marginal log-likelihood.*

We now consider the problem of clustering simple image data. We will again use a subset of the MNIST handwritten digit database, which in addition to being a benchmark for testing classification algorithms, is often used to validate unsupervised learning algorithms. The particular version of the data which we'll use is available here:

`/course/cs195f/asgn/clustering/mnist_all.mat`

In addition, a Matlab script `hw6_kmeans.m` is posted in the same directory, which contains some code to help you get started.

In the MNIST database, each training or test example is a 28-by-28 grayscale image. To ease programming of learning algorithms, these images have been converted to vectors of length $28^2 = 784$ by sorting the pixels in raster scan (row-by-row) order. The Matlab `reshape` command can be used to convert these vectors back to images for visualization. For example, we can plot the third training example of class 1 as follows:

```
>> imagesc(reshape(train1(3,:), 28, 28)');
```

**Question 3:**

*In this question, we use the K-means algorithm to cluster the handwritten digit data. For all sections, we use* $1,000$ *examples of each of the 10 digit classes, so that there are* $N = 10,000$ *data items in total. Letting* $\mu_k$ *denote the mean for cluster* $k$*, the K-means objective function can be written as*

$$J(y, \mu) = \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} ||x_i - \mu_k||^2$$

*where* $y_{ik} = 1$ *if example* $i$ *is assigned to cluster* $k$*, and 0 otherwise. Note that since we are testing a clustering algorithm, the true MNIST digit class labels will be used only to evaluate hypothesized clusterings,* not *as part of the clustering algorithm.*

a) *Implement and submit a function which runs the K-means algorithm to convergence for any* $K$*, from an initialization specified via a set of starting cluster centers* $\{\mu_k\}_{k=1}^{K}$*. Your function should record and return the value of the K-means objective function* $J(y, \mu)$ *at each iteration. You must write your own implementation, not use or copy an existing Matlab function.*

b) *Run the K-means algorithm on the digit data with* $K = 10$*, the true number of clusters. Randomly initialize K-means by choosing* $K = 10$ *of the observations at random, and setting the initial cluster centers to be these observations. Plot the K-means objective as a function of iteration, and verify that it monotonically decreases.*

   Hint: *Use* `randperm.m` *to generate random initializations, as in the example code.*

c) *Repeat part (b) for 10 different random initializations, running the K-means algorithm to convergence from each. Evaluate the consistency of each resulting clustering with the true digit labels by computing the Rand index (the second output argument of the function* `valid_RandIndex.m`*). Make a scatter plot of the Rand index values, versus the corresponding values of the K-means objective function* $J(y, \mu)$*. Does the K-means objective provide a good predictor of cluster quality?*

d) *When clustering algorithms do not perform perfectly, there are two major sources of error: the objective function or model may not match the data well, or the algorithm used to optimize that objective may be stuck in local optima. We can sometimes separate these issues by "cheating". Consider a K-means initialization in which the cluster centers* $\mu_k$ *are set to the means of the 10 digit classes, as determined via the* true *class labels. Run K-means to convergence from this initialization, and compute the resulting Rand index and objective function values. How do these compare to those from part (c)? What does this suggest about how we should try to build a better clustering method for this data?*

e) *We conclude by considering how K-means performs on this data as the number of clusters,* $K$*, is varied. For each* $K \in \{2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$*, run K-means to convergence from a single random initialization determined as in part (b). Plot both the K-means objective function and the Rand index (computed using the true cluster labels) versus* $K$*. Which value of* $K$ *gives the lowest objective function? Which gives the largest Rand index? How could we determine a good value for* $K$ *without using the true class labels?*