Massachusetts Institute of Technology

6.867 Machine Learning, Fall 2006

Problem Set 5

Due Date: Thursday, Nov 30, 12:00 noon

You may submit your solutions in class or in the box.

1. Wilhelm and Klaus are friends, but Wilhelm only speaks English and Klaus only speaks German. Klaus has sent Wilhelm an urgent message, and Wilhelm is locked in a room with only the proceedings of the European Parliament (in both German and English) to keep him company. Luckily, Wilhelm is a machine learning expert.

   He decides to treat Klaus as a "noisy channel": assume that thoughts in Klaus's head start out in English, but before they are emitted, they pass through a medium that replaces the English words with new (German) words and rearranges them from the English word-ordering to a new (German) ordering all according to some unkown distributions. Next, he plans to come up with parametrizations for the distributions and then to optimize the parameters over the European Parliament data he is conveniently locked up with.

   Wilhelm thus achieves the following optimization problem:

$$\begin{aligned} \arg\max_{\mathbf{e}} P(\mathbf{e} \,|\, \mathbf{g}) &= \arg\max_{\mathbf{e}} P(\mathbf{g} \,|\, \mathbf{e}) \frac{P(\mathbf{e})}{P(\mathbf{g})} \\ &= \arg\max_{\mathbf{e}} P(\mathbf{g} \,|\, \mathbf{e}) P(\mathbf{e}), \end{aligned}$$

   where $\mathbf{e}$ and $\mathbf{g}$ denote English and German sentences, respectively. This corresponds to Wilhelm's noisy-channel characterization of Klaus's head.

   Next, Wilhelm decides on parameterizations for the two terms in his optimization criterion. To estimate probabilities of English sentences, he will pretend that English is a Markov process (that is, that English speakers choose each word in a sentence based only on the last two words they have chosen):

$$P(\mathbf{e}) = \prod_{i=1}^{|\mathbf{e}|} LM(\mathbf{e}_i \,|\, \mathbf{e}_{i-1}, \mathbf{e}_{i-2}), \tag{1}$$

   where $|\mathbf{e}|$ denotes the length of the English sentence $\mathbf{e}$, $\mathbf{e}_i$ denotes the ith word of $\mathbf{e}$, and $\mathbf{e}_0$ and $\mathbf{e}_{-1}$ correspond to start symbols S_0 and S_{-1}, respectively, which are before every English sentence for simplicity.

   For the other term, Wilhelm is aware that both the words and their order may change, so he introduces a hidden variable: the alignment between the words.

$$P(\mathbf{g}, a \,|\, \mathbf{e}) = \prod_{j=1}^{|\mathbf{g}|} T(\mathbf{g}_j \,|\, \mathbf{e}_{a_j}) D\left(a_j \,|\, j, |\mathbf{e}|, |\mathbf{g}|\right) \tag{2}$$

Thus,

$$\begin{aligned} P(\mathbf{g} \mid \mathbf{e}) &= \sum_a P(\mathbf{g}, a \mid \mathbf{e}) \\ &= \sum_a \prod_{j=1}^{|\mathbf{g}|} T(\mathbf{g}_j \mid \mathbf{e}_{a_j}) D\left(a_j \mid j, |\mathbf{e}|, |\mathbf{g}|\right) \\ &= \prod_{j=1}^{|\mathbf{g}|} \sum_{a_j=1}^{|\mathbf{e}|} T(\mathbf{g}_j \mid \mathbf{e}_{a_j}) D\left(a_j \mid j, |\mathbf{e}|, |\mathbf{g}|\right). \end{aligned}$$

This set-up is known as IBM Model 2.

(a) To perform the optimization, Wilhelm decides to use the EM algorithm. Derive the updates of the parameters LM, T, and D. We place no constraints on these distributions other than that they are distributions (cf. regularization below).

(b) The initial parameter settings are important. Give an example of a bad choice. What should you choose?

(c) Insert the updates and initial parameter settings in the provided templates `ibm2_train_lm.m` and `ibm2_train.m` and train the system using the provided data `europarl.m`. Use the provided decoder `ibm2_beam_decoder.m` to use the trained system to decode Klaus's message `klaus.m`. The system is able to achieve surprisingly good results with a very simple probability model! (Although the training is easy, the decoding is non-trivial; check out the source of the decoder.)

(d) Suppose Wilhelm believes that his model is reordering the German too aggressively. He is considering regularizing the alignments. He has several ideas; what effect do you think each of these regularization techniques will have?

i. Regularize $D\left(a_j \mid j, |\mathbf{e}|, |\mathbf{g}|\right)$ toward a uniform distribution.

ii. Regularize $D\left(a_j \mid j, |\mathbf{e}|, |\mathbf{g}|\right)$ toward the distribution $D\left(a_j = \left\lceil j \cdot \frac{|\mathbf{e}|}{|\mathbf{g}|} \right\rceil \mid j, |\mathbf{e}|, |\mathbf{g}|\right) = 1$.

iii. Estimate first a simpler distribution $D\left(a_j \mid |\mathbf{e}|, |\mathbf{g}|\right)$ and regularize $D\left(a_j \mid j, |\mathbf{e}|, |\mathbf{g}|\right)$ toward it.

2. In this problem, we will explore the combined use of regression and Expectation Maximization. The motivating application is as follows: we'd like to understand how the expression levels of genes (in a cell) change with time. The expression level of a gene indicates, roughly, the amount of gene-product in the cell. The experimental data we have available are noisy measurements of this level (for each gene) at certain time points. More precisely, we have m genes $(g_1, \ldots, g_m)$ and measurements are available for r timepoints $(t_1, \ldots, t_r)$. The data matrix is $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_m]^T$ where each row $\mathbf{y}_i^T = [y_{i1}, \ldots, y_{ir}]$ corresponds to the expression levels of gene g_i across the r timepoints.

Our goal is to estimate continuous functions, each $\hat{f}_i(t)$ capturing the expression level of one gene g_i. We'll call these expression curves. Clearly, if we treat each gene's expression as independent from others, we would need to perform m unrelated regression tasks. We can do better, however, since we expect that there are groups of genes with similar expression curves.

We hypothesize that the expression curves can be grouped into k classes and that, within each group, the functions $f_i(t)$ look very similar. When estimating $\hat{f}_i(t)$ we can pool together data from all the genes in the same class. We have two problems to solve: (1) figure out which class each gene belongs to, and (2) given class assignments, estimate the expression curves. As you might guess, we will use the

EM algorithm. However, we still need to understand how to estimate continuous curves from discrete data-points.

To perform regression, we will use an idea similar to kernel regression. Recall that even if $f(t)$ is some non-linear function of t, it is possible to find a feature mapping $\phi(t) = [\phi_1(t), \ldots, \phi_p(t)]^T$ such that $f(t)$ is a linear function of ϕ's: $f(t) = \sum w_j \phi_j(t)$. In class, we had discussed how to implement this idea to perform kernel regression without explicitly constructing the feature mapping. In the current setting, we extend this to include a prior over the parameters $\mathbf{w} = [w_1, \ldots, w_p]^T$. We set this to be simply a spherical multivariate Gaussian $N(\mathbf{w}; \mathbf{0}, \sigma_w^2 \mathbf{I})$. Now, given any set of time points $t_1, \ldots, t_r$, we imagine generating function values at those time points according to the following procedure

Step 1: draw one sample $\mathbf{w} \sim N(\mathbf{0}, \sigma_w^2 \mathbf{I})$

Step 2: generate function values for the time points of interest

$$f(t_k) = \sum_{j=1}^{p} w_j \phi_j(t_k), \quad k = 1, \ldots, r \tag{3}$$

Since w_j are normally distributed, so is $\mathbf{f} = [f(t_1), \ldots, f(t_r)]^T$. The distribution of this vector is $N(\mathbf{f}; \mathbf{0}, \mathbf{K})$ where $K_{kl} = K(t_k, t_l) = \sigma_w^2 \phi(t_k)^T \phi(t_l)$. Looks familiar? Let's try to understand it a bit better. We have a distribution over function values at any set of time points. In other words, we have defined a prior distribution over functions! The kernel function $K(t, t')$, where the inputs are the time points, gives rise to a Gram matrix $K_{kl} = K(t_k, t_l)$ over any finite set of inputs. This probability distribution over functions is known as a *Gaussian Process* (GP for short).

The choice of the kernel $K(t, t')$ controls the kind of functions that are sampled. Consider the following kernel, similar to the radial basis function kernel seen before

$$K(t, t') \quad = \quad \sigma_f^2 \exp(-\frac{(t - t')^2}{2\rho^2}) \tag{4}$$

The hyperparameter ρ controls the time span at which the function values are strongly coupled. A large ρ implies, for example, that the sampled functions would appear roughly constant across a long time range. σ_f scales this correlation by a constant factor.

(a) Match each of the following four figures, with the four kernels defined below. Each figure describes some functions sampled using a particular kernel. Note that these functions are drawn by selecting finely spaced time points and drawing a sample from the corresponding function values at these points from $N(\mathbf{f}; \mathbf{0}, \mathbf{K})$ where $\mathbf{K}$ is the Gram matrix over the same points.

Here are the kernels:

i. $3 \exp(-\frac{(t-t')^2}{2(2.1)^2})$

ii. $2 \exp(-\frac{(t-t')^2}{2(0.7)^2})$

iii. $2 \exp(\frac{-sin^2(\frac{\pi(t-t')^2}{4})}{2(0.3)^2})$

iv. $2 \exp(-\frac{(t-t')^2}{2(0.05)^2}) + 0.1(tt')^2$

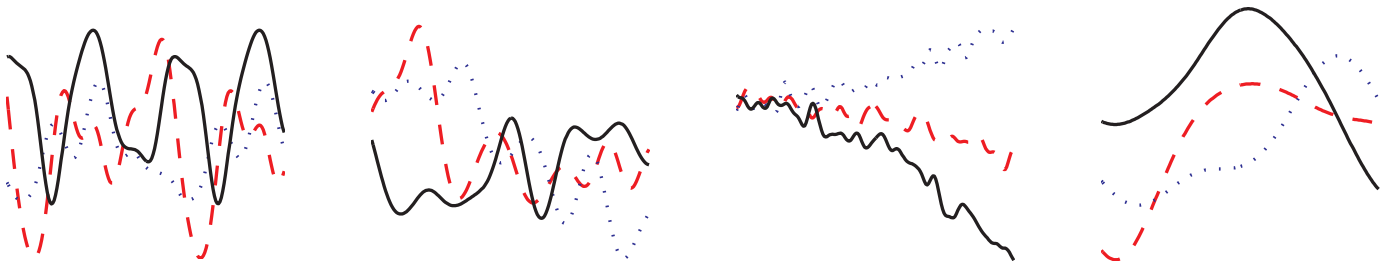

Figure 1: Each subfigure corresponds to one kernel

Let us now consider how we will use GP's to simultaneously perform regression across genes belonging to the same class. Since we assume they have similar expression curves we will model them as samples from the same Gaussian process. Our model for the data is as follows. There are k classes in the data and each gene belongs to a single class. Let the genes in class l be $g_{l1}, g_{l2}, \ldots, g_{lm_l}$ where m_l is the number of genes in that class. The expression curves for these genes are assumed to be samples from the same GP. The kernels of the k GP's (one for each class) all have the same parametric form, but different hyperparameters $\theta = (\sigma_f, \rho, \sigma_n)$:

$$K(t,t') = \sigma_f^2 \exp(-\frac{(t-t')^2}{2\rho^2}) + \sigma_n^2 \delta(t,t'), \quad \text{where} \tag{5}$$

$$\delta(t,t') = \begin{cases} 1 & \text{if } t = t' \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

This is essentially the same as before except for the second term that accounts for measurement noise. In other words, the responses y are modeled as $y(t) = f(t) + \epsilon_t$, $\epsilon_t \sim N(0, \sigma_n^2)$ and $f(t)$ is a Gaussian process with kernel given by Egn 4. Thus, for any time points $t_1, \ldots, t_r$, the response vector $\mathbf{y}$ has a distribution $N(\mathbf{0}, \mathbf{K})$, where $\mathbf{K}$ is the Gram matrix from the kernel in Eqn 5.

(b) We have provided for you a function `log_likelihood_gp(params, t, Yobs)`, where `params` specify the hyperparameters of the kernel, `t` is a vector of r timepoints and `Yobs` is a $q \times r$ matrix of observed gene expression values (for q genes) at those timepoints. For each of the q sets of observations, the function computes the log-likelihood that observations were generated from the GP specified by `params`. Fill in the missing lines of the code. Attach a printout of your code (you only need to show the parts you changed).

We now go back to our original problem and set up an Expectation Maximization framework to solve it. Suppose we know that the genes in the data cluster into k classes (i.e., k is a user-specified parameter). We start off by guessing some initial values for the hyperparameters of the k GPs. In each E-step, we compute the posterior assignment probability $P(k|i)$ i.e., the probability that gene g_i is in class k. To simplify the M-step we will turn these probabilities into hard assignments. In each M-step, we compute the MLE estimates for the hyperparameters of the GPs associated with the classes.

(c) To perform EM, we have provided you a function `[W,V,L] = em_gp(t, Yobs, k, init_class)`. The code is missing some steps in the `Expectation` sub-function. Fill them in. Attach a printout of your code for the `Expectation` function.

(d) Using the dataset provided with this problem, run `em_gp` with different choices of k ($k = 2, 3, 4, 5$). We have provided a function `plot_results(W,V,t,Yobs,k,nn)` which you may find useful. You can use it to plot `nn` randomly chosen curves from each of the k classes. What choice of k seems most appropriate?

(e) Gene expression data often has missing values, i.e., you may not know the value y_{ij}, the expression of gene g_i at time t_j. With some machine learning methods, handling these missing values can be a big hassle. But with our model, the missing values problem can be handled very simply. How?

(f) **(Optional: Extra Credit)** Our initial choice of hyperparameters is constructed as follows: we assign genes to specific classes, either randomly or based upon some insights. Using this class assignment, the initial hyperparameters are computed by an MLE approach. Describe a method that could be used to generate a good initial class assignment.

(g) **(Optional: Extra Credit)** Since we have defined a prior over functions we could turn the clustering problem into a Bayesian model selection problem. We assume that each cluster has a single underlying expression curve $f(t)$, sampled from a fixed GP model, and the responses for each gene in the cluster are subsequently sampled from $y(t) = f(t) + \epsilon_t$, $\epsilon_t \sim N(0, \sigma_n^2)$, using the same $f(t)$. Write an expression for the marginal likelihood of the data corresponding to a fixed assignment of genes into clusters.