CS540 Spring 2010: homework 8

1 Variational Bayes for univariate Gaussian

The goal of this exercise is to reproduce Figure 1. Also, derive an expression for the lower bound $L(q)$, and check that the algorithm monotonically increases it. Turn in your math, code and plots.
To solve this, read sec 12.4 of my chapter on variational inference. These notes are not finished, but should contain enough information for you to solve this problem. If you want to read more, you can either wait until my notes are finished (by Sunday 28th), or you can read chapter 10 of [Bis06], which I have posted on the class web page (the online posting calls it "chapter 5", since it is an old version), or you can read chapter 33 of [Mac03], which is available online at `http://www.inference.phy.cam.ac.uk/mackay/itila`.

2 Forwards vs reverse KL divergence

(Source: Exercise 33.7 of [Mac03]) Consider a factored approximation $q(x, y) = q(x)q(y)$ to a joint distribution $p(x, y)$. Show that to minimize the forwards KL

$$\mathbb{KL}\left(p||q\right) = \sum_{xy} p(x, y) \log \frac{p(x, y)}{q(x, y)} \tag{1}$$

we should set $q(x) = p(x)$ and $q(y) = p(y)$, i.e., the optimal approximation is a product of marginals
Now consider the reverse KL

$$\mathbb{KL}\left(q||p\right) = \sum_{xy} q(x, y) \log \frac{q(x, y)}{p(x, y)} \tag{2}$$

and the following joint distribution, where the rows represent y and the columns x.

	x			
	1	2	3	4
1	1/8	1/8	0	0
2	1/8	1/8	0	0
3	0	0	1/4	0
4	0	0	0	1/4

Show that $\mathbb{KL}\left(q||p\right)$ for this p has three distinct minima. Identify those minima and evaluate $\mathbb{KL}\left(q||p\right)$ at each of them. What is the value of $\mathbb{KL}\left(q||p\right)$ if we set $q(x, y) = p(x)p(y)$?

3 Multi-task regression with mixture of Gaussians prior

Multi-task learning refers to solving multiple related (supervised) learning problems. For example, consider fitting multiple straight lines, all of which are assumed to have a similar slope, but may have different intercepts. This is illustrated in Figure 2(a). We can fit each line separately, but when there is a lot of noise and/or there is very little data, we can do better by fitting all the lines simultaneously, and exploiting the fact that there slopes are similar. We can do this using the following hierarchical model: we assume $\mathbf{w}_t \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\mathbf{w}_t$ are the regression parameters for

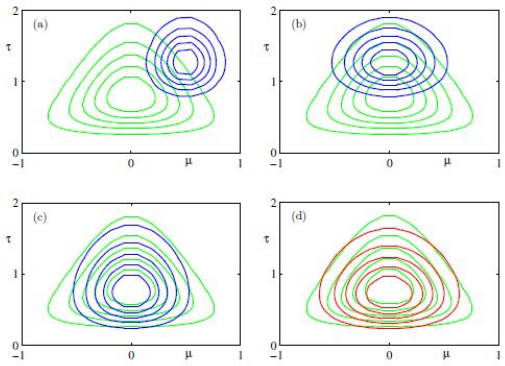


Figure 1: Factored variational approximation to the Gaussian-Gamma distribution. (a) Initial guess. (b) After updating q_μ. (c) After updating q_λ. (d) At convergence. Source: Figure 10.4 of [Bis06]. See also Figure 33.4 of [Mac03]. According to Figure 24.1 of [Mac03], the data has $N = 5$ points, a mean of $\overline{x} = 1$, and $S^2 = \sum_i (x_i - \overline{x})^2 = 1$. However, you are free to define your own data. It is not clear what the prior parameters were; try a few until the plot looks somewhat similar.

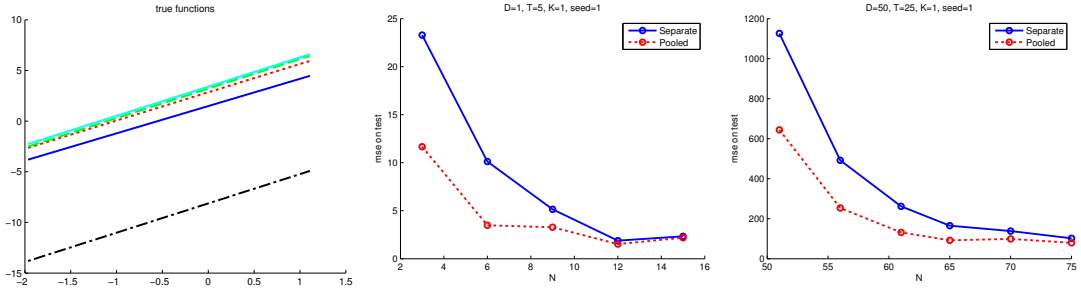


Figure 2: (a) 5 different but related linear regression models in 1d. Notice how all lines have the same slope. (b) Mean squared error on the test set vs the size of training set for two different models: fitting each line separately, or using a common Gaussian prior. (c) Same as (b) but in $D = 50$ dimensions and with $T = 25$ tasks. Made by `multitaskRegDemo`.

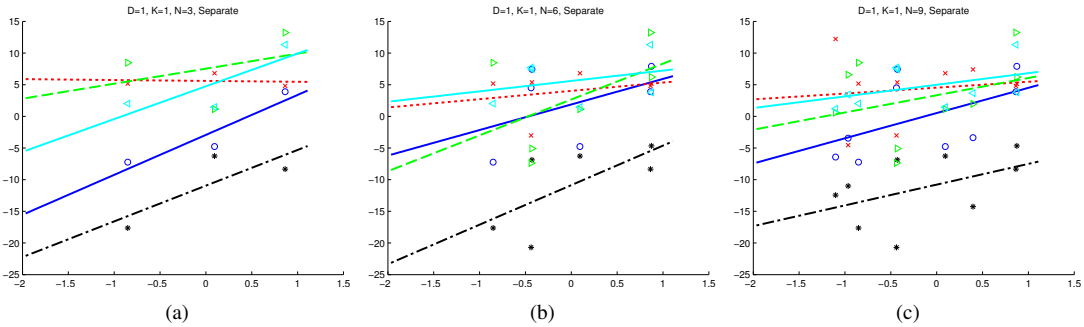
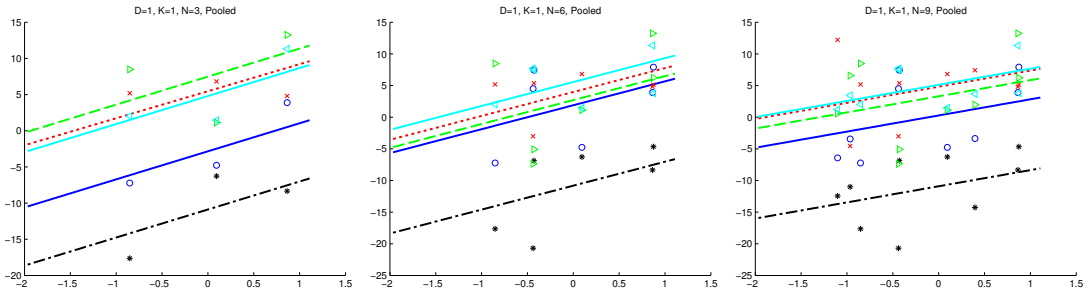


Figure 3: Fit for each line for $N \in \{3, 6, 9\}$ where each model is fit separately. Made by `multitaskRegDemo`.



(a) (b) (c)

Figure 4: Fit for each line for $N \in \{3, 6, 9\}$ where all the models are fit jointly. Notice how the estimates are closer to the truth (Figure 2(b)) than when fitting separately, even for small sample sizes. In particular, all liens are encouraged to be parallel. Made by `multitaskRegDemo`.

task t, $t = 1 : T$. We can then use empirical Bayes to estimate the common slope $\boldsymbol{\mu}$, and the degree of heterogeneity amongst the lines, $\boldsymbol{\Sigma}$. Once $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ have been estimated, we can compute a MAP estimate of each line separately using

$$(\hat{\mathbf{w}}_t, \hat{b}_t, \hat{\sigma}_t^2) = \arg\max \left[\prod_{i=1}^{N} \mathcal{N}(y_{it} | b_t + \mathbf{w}_t^T \mathbf{x}_i, \sigma_t^2) \right] \mathcal{N}(\mathbf{w}_t | \boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{3}$$

where b_t is the bias or offset for task t. This is a bit like ridge regression, except instead of shrinking towards $\mathbf{0}$, we shrink towards the common mean. (In this model, we assumed each task had the same input, so we wrote $\mathbf{x}_i$ instead of $\mathbf{x}_{it}$, but it is straightforward to generalize the technique.)

In Figure 2(b), we see that pooled estimated gives improved performance over estimating each line separately. Figures 3 and 4 show the individual lines for different training set sizes for the separate and pooled techniques. We see that the pooled technique forces the lines to be nearly parallel (the strength of this force being determined by $\boldsymbol{\Sigma}$, which is currently set by hand), which results in better predictive accuracy. Figure 2(c) shows that this improvement is even more dramatic in higher dimensional settings.

A real world application of this is **conjoint analysis**, which is widely used in business and marketing [LDGY96, AEP08]. In this setting, the goal is to figure out which aspects of a product customers like. This is usually assessed by presenting to a small number N of products to a large number T of customers, and asking them to rate the product. Thus $\mathbf{x}_i$ is the feature vector describing product i, y_{it} is the rating of customer t for product i, and $\mathbf{w}_t$ are the parameter of customer t's prediction function.

It is not always reasonable to assume all customers are alike. Similarly, it is not always reasonable to assume all straightlines will have similar slopes. Figure 5(a) illustrates some data where there are two kinds of lines, sloping up and down. If we pool all the parameters on this kind of data, the performance is worse than not using pooling, as illustrated in Figure 5(b-c). The reason is that the inductive bias of our prior is wrong.

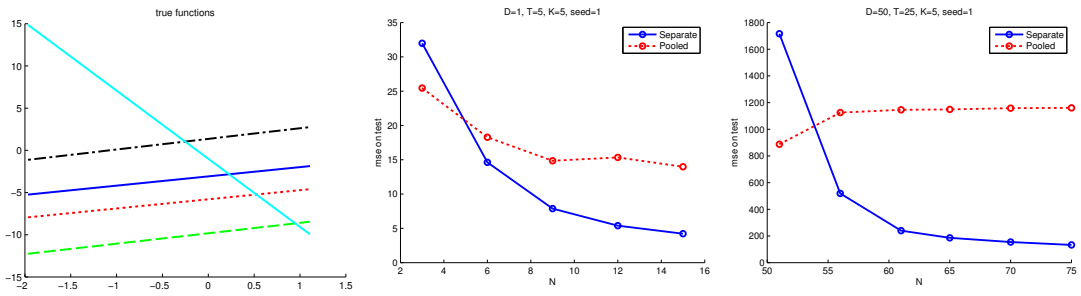
A simple solution to this is to use a mixture model as prior. (A fancier technique is to use a non-parametric prior, such as a Dirichlet process mixture model.) Such flexible priors can provide robustness against prior mis-specification. The model can be defined as follows. For each task, we have a latent variable $z_t \in \{1, \ldots, K\}$ specifying which cluster it belongs to. If $z_t = k$, we assume $\mathbf{w}_t$ is generated from the k'th Gaussian:

$$p(\mathbf{w}_t | z_t = k, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{w}_t | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{4}$$

$$p(z_t = k | \boldsymbol{\theta}) = \pi_k \tag{5}$$

See Figure 6.

The goal of this exercise is to figure out how to fit this model using EM. Derive the E and M steps, and then implement them. Add your code to the `multitaskRegDemo` file as a third method. Run the demo and plot the results; try $K \in \{1, 5, 10\}$ mixture components. (The $K = 1$ case should give qualitatively similar results to the heuristic pooling code that I wrote.) You may need to use priors to regularize the problem. Turn in your math, code and plots. (Plots are only printed for the first random seed, but the program tries 3 seeds just to check that differences are not accidental.)



(a) (b) (c)

Figure 5: Same as Figure 2, except the regression parameters are sampled from a mixture of Gaussians. Notice how some lines slope down and some slope up. In this case, fitting assuming all the model parameters come from a common Gaussian hurts performance. Made by `multitaskRegDemo`.

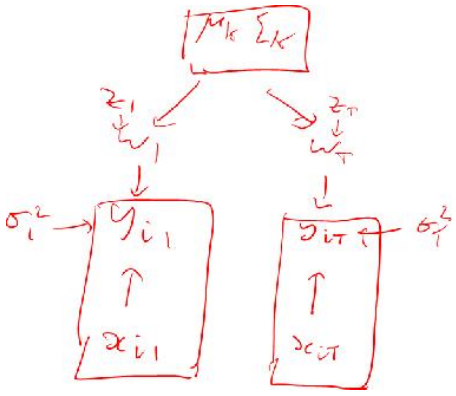


Figure 6: Model for multi-task regression where the regression weights are drawn from a mixture of Gaussians. See text for details.

Hint: let the parameters (to be point-estimated) be $\boldsymbol{\theta} = (\boldsymbol{\mu}_{1:K}, \boldsymbol{\Sigma}_{1:K}, \mathbf{b}_{1:T}, \boldsymbol{\sigma}^2_{1:T})$. Then the complete data log likelihood is given by

$$\log p(\mathcal{D}, \mathbf{z}_{1:T}, \mathbf{w}_{1:T}|\boldsymbol{\theta}) = \sum_t \left\{ \sum_k \mathbb{I}(z_t = k) \left(\log \pi_k + \log \mathcal{N}(\mathbf{w}_t|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) + \sum_i \log \mathcal{N}(y_{it}|b_t + \mathbf{x}_i^T \mathbf{w}_t, \sigma_t^2) \right\} \tag{6}$$

One can take expectations of this using the usual iterated expectation trick and exploiting the fact that if z_t is known, the posterior $p(\mathbf{w}_t|z_t = k, \mathcal{D}, \boldsymbol{\theta})$ is just a Gaussian. In more detail, define $f(z_t, \mathbf{w}_t)$ as one term in the above sum. We have

$$\mathbb{E}\left[f(Z_t, \mathbf{w}_t)|\mathcal{D}, \boldsymbol{\theta}^{old}\right] = \sum_k p(z_t = k|\mathcal{D}, \boldsymbol{\theta}^{old}) \mathbb{E}\left[f(Z_t = k, \mathbf{w}_t)|\mathcal{D}, \boldsymbol{\theta}^{old}\right] \tag{7}$$

So marginalizing out z_t means that $p(\mathbf{w}_t|\mathcal{D}, \boldsymbol{\theta})$ is a mixture of K Gaussians, from which we can compute the mean and covariance; from this we can compute the expected sufficient statistics need to update $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$.
Hint 2: if the above method is too complicated for you, you can use hard-clustering for the z_t's (treat them as part of $\boldsymbol{\theta}_t$). This should simplify the problem somewhat.
Hint 3: if this is still too complicated for you, just solve the $K = 1$ case (i.e., using a single Gaussian prior, not a mixture). This should give performance similar to my heuristic pooling code, but may work better, since I hard-code the value of $\boldsymbol{\Sigma}$, whereas using EM to estimate it should work better (assuming a suitable regularizer on $\boldsymbol{\Sigma}$!)

References

[AEP08] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[Bis06] C. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[LDGY96] P. Lenk, W. S. DeSarbo, P. Green, and M. Young. Hierarchical Bayes Conjoint Analysis: Recovery of Partworth Heterogeneity from Reduced Experimental Designs. *Marketing Science*, 15(2):173–191, 1996.

[Mac03] D. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.