Homework 3

Assigned: 10/10/2011.
Due: 21/10/2011.

Exercise 1. BAYESIAN UPDATE WITH CHOLESKY FACTORS

Remember the robot localization example from the lecture. The robot's state is coded in $\boldsymbol{u} \in \mathbb{R}^n$, but the knowledge of the robot about the value of $\boldsymbol{u}$ is uncertain. Say, its belief in $\boldsymbol{u}$ is $N(\boldsymbol{u}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ (robotics people call this *belief state*). Obtaining a noisy measurement y, whose likelihood is $P(y|\boldsymbol{u}) = N(\boldsymbol{x}^T\boldsymbol{u}, \sigma^2)$ for some covariate $\boldsymbol{x}$ (which is known to the robot, say the specification of sensors), the robot can update its belief from prior $P(\boldsymbol{u})$ to posterior $P(\boldsymbol{u}|y)$, which becomes the new belief state. This process is called *Bayesian filtering* (you'll see more of that).
From the lecture, we know that $P(\boldsymbol{u}|y) = N(\boldsymbol{\mu}', \boldsymbol{\Sigma}')$, where

$$\boldsymbol{\Sigma}' = \boldsymbol{\Sigma} - \boldsymbol{\Sigma}\boldsymbol{x}(\sigma^2 + \boldsymbol{x}^T\boldsymbol{\Sigma}\boldsymbol{x})^{-1}\boldsymbol{x}^T\boldsymbol{\Sigma}, \quad \boldsymbol{\mu}' = \boldsymbol{\mu} + \boldsymbol{\Sigma}\boldsymbol{x}(\sigma^2 + \boldsymbol{x}^T\boldsymbol{\Sigma}\boldsymbol{x})^{-1}(y - \boldsymbol{x}^T\boldsymbol{\mu}). \quad (1)$$

Revisit the material, make yourself clear why these formulae are the way they are, and that there is nothing messy about them, because you know where they come from (it's just conditioning, "product"). Also remember which type of Gaussian parameterization we are using here, that there is a different type in which all problems here would disappear, although in that case, other essential operations would be tricky (marginalization, "sum").
However, these update equations do not have good numerical properties. In other words, assume the robot got a lot of measurements, and compare the *online* setting of absorbing one measurement at a time by (1) with the *offline* setting of collecting all measurements, then doing a single update. Quite likely, you would get substantially different results. In fact, with a static robot, it would probably all work out, but once the robot moves around, increasing the uncertainty in $P(\boldsymbol{u})$ due to the movement (diffusion), you'd be in trouble.
In fact, whatever you do in an online scenario, you'll always have more numerical error (on average) than offline. But you can do better than (1): the topic of this exercise. This is not a numerical mathematics course, so we'll not try to find out *why* this is better. Roughly, matrices, and matrix operations, with large condition numbers have to be avoided, and the condition number of a Cholesky factor is much less than of a matrix itself (square root).

1. **[1 point]** Let $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^T$ be the Cholesky factorization. The key is to maintain $\boldsymbol{L}$ instead of $\boldsymbol{\Sigma}$. Also, let $\boldsymbol{\mu} = \boldsymbol{L}\boldsymbol{a}$. Our *belief representation* will be $(\boldsymbol{L}, \boldsymbol{a})$. A representation is any amount of information, fulfilling:

   - Reasonable storage requirements
   - Desired queries to $P(\boldsymbol{u})$ can be computed efficiently enough
   - For Bayesian updates $P(\boldsymbol{u}) \to P(\boldsymbol{u}|y)$, the representation update $(\boldsymbol{L}, \boldsymbol{a}) \to (\boldsymbol{L}', \boldsymbol{a}')$ can be computed *reliably* (with as little numerical error as possible) and efficiently enough

   In short, you describe and communicate your method in terms of distributions $P(\boldsymbol{u})$, maybe graphs, but you *implement* it in terms of a representation.
   For some $\boldsymbol{v}$, $r = \boldsymbol{v}^T\boldsymbol{u}$ ($\boldsymbol{u} \sim P(\boldsymbol{u})$) is Gaussian. Show how to compute mean, variance of r most efficiently. What is the cost? Compare that to the cost for using the representation $(\boldsymbol{\Sigma}, \boldsymbol{\mu})$ instead for this query.

2. **[3 points]** Now to the update, based on observation $(\boldsymbol{x}, y)$. Suppose (in this part) that you have $\boldsymbol{w} = \boldsymbol{L}^T\boldsymbol{x}$. Bring (1) into the form $\boldsymbol{\Sigma}' = \boldsymbol{L}(\boldsymbol{I} - \boldsymbol{v}\boldsymbol{v}^T)\boldsymbol{L}^T$, $\boldsymbol{\mu}' = \boldsymbol{L}\boldsymbol{b}$, and show how to compute $\boldsymbol{v}$, $\boldsymbol{b}$ in $O(n)$.
Hint: You may define another vector as a rescaled version of $\boldsymbol{w}$. If you divide by numbers, argue why they are $\neq 0$.

3. **[1 point]** Suppose I gave you some *Cholesky downdate* code, which maps $(\boldsymbol{L}, \boldsymbol{z}, \boldsymbol{v}, \alpha) \to (\boldsymbol{L}', \boldsymbol{z}')$, so that

$$\boldsymbol{L}'(\boldsymbol{L}')^T = \boldsymbol{L}(\boldsymbol{I} - \boldsymbol{v}\boldsymbol{v}^T)\boldsymbol{L}^T, \quad \boldsymbol{L}'\boldsymbol{z}' = \boldsymbol{L}(\boldsymbol{z} - \alpha\boldsymbol{v})$$

for nonzero $\boldsymbol{v}$ and some α of your choice. Show how to compute the representation update $(\boldsymbol{L}, \boldsymbol{a}) \to (\boldsymbol{L}', \boldsymbol{a}')$, if all you are allowed to do is call the downdate routine once, do a single multiplication matrix-vector multiplication with $\boldsymbol{L}^T$, and $O(n)$ computations otherwise.
Such downdate code is contained in numerical packages. On the course website, I have linked some wrapper code of mine, as well as a report telling you how this works (if you are interested). The procedure you developed here seems a convolved way of doing (1), but this "intuition" is wrong. It comes at exactly the same computational cost (time, memory; in fact, it is even slightly faster), and has much better numerical properties.

Exercise 2. Spectral Analysis of Conjugate Gradients Algorithm
Recall the CG algorithm for approximately solving $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is positive definite. The algorithm minimizes the quadratic $q(\boldsymbol{x}) = (1/2)\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}$ iteratively, constructing a sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$, requiring a single matrix-vector multiplication with $\boldsymbol{A}$ per iteration. After at most n steps, neglecting numerical errors (which, in practice, you cannot!), the exact solution $\boldsymbol{x}_* = \boldsymbol{A}^{-1}\boldsymbol{b}$ is reached, in that $\boldsymbol{x}_n = \boldsymbol{x}_*$. Depending on $\boldsymbol{A}$, this can also happen earlier (you can use the Cayley-Hamilton theorem from linear algebra to understand this point). However, the main rationale for CG today is to *approximate* $\boldsymbol{x}_*$ by $\boldsymbol{x}_k$ with $k \ll n$. Whether $\boldsymbol{x}_k$ is close to $\boldsymbol{x}_*$ or not, depends on properties of $\boldsymbol{A}$ and $\boldsymbol{b}$ (it depends on numerical errors as well, but we ignore these in the present exercise, assuming that all computations are exact). In this exercise, we will analyze the convergence behaviour in terms of the eigenspectrum of $\boldsymbol{A}$.
Let $\boldsymbol{x}_* = \boldsymbol{A}^{-1}\boldsymbol{b}$ and $q_* = q(\boldsymbol{x}_*) = -(1/2)\boldsymbol{b}^T\boldsymbol{A}^{-1}\boldsymbol{b}$. Then, $q(\boldsymbol{x}_k)$ is nonincreasing and $\geq q_*$. We'll try to bound $q(\boldsymbol{x}_k) - q_*$. The eigendecomposition is $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$, $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ orthonormal ($\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$), $\boldsymbol{\Lambda}$ diagonal (positive elements).

1. **[2 points]** Assume that we start from $\boldsymbol{x}_0 = \boldsymbol{0}$. The Krylov subspace $\mathcal{K}_k$ is spanned by $\{\boldsymbol{A}^j\boldsymbol{b} \,|\, j = 0, \ldots, k-1\}$. We saw that $\boldsymbol{x}_k = \mathrm{argmin}_{\boldsymbol{x}\in\mathcal{K}_k} q(\boldsymbol{x})$. For a polynomial $P(t) = \sum_{j=0}^{k-1} \alpha_j t^j$, $\alpha_j \in \mathbb{R}$, define

$$P(\boldsymbol{B}) := \sum_{j=0}^{k-1} \alpha_j \boldsymbol{B}^j, \quad \boldsymbol{B} \in \mathbb{R}^{n\times n}.$$

Take care that $P(\boldsymbol{B})P(\boldsymbol{C}) \neq P(\boldsymbol{C})P(\boldsymbol{B})$ in general if $\boldsymbol{B}$, $\boldsymbol{C}$ do not commute, and recall that $\boldsymbol{B}^0 := \boldsymbol{I}$. Show that

$$P(\boldsymbol{A}) = \boldsymbol{Q}P(\boldsymbol{\Lambda})\boldsymbol{Q}^T.$$

Defining $\boldsymbol{y} = \boldsymbol{Q}^T\boldsymbol{x}$, $\bar{\boldsymbol{b}} = \boldsymbol{Q}^T\boldsymbol{b}$, show that $q(\boldsymbol{x})$ and q_* can be written in terms of $\boldsymbol{y}$, $\bar{\boldsymbol{b}}$, and $\{\lambda_i\}$. *Hint:* What is an eigendecomposition of $\boldsymbol{A}^{-1}$?

2. **[4 points]** Let $\boldsymbol{y}_k = \boldsymbol{Q}^T\boldsymbol{x}_k$. Show that $\boldsymbol{x}_k = P_k(\boldsymbol{A})\boldsymbol{b}$ for some polynomial $P_k(t)$ of degree $< k$, and that $\boldsymbol{y}_k = P_k(\boldsymbol{\Lambda})\bar{\boldsymbol{b}}$. In other words, $y_{k,i} = P_k(\lambda_i)\bar{b}_i$. By writing $q(\boldsymbol{x}_k)$

in terms of $\boldsymbol{y}_k$, prove that

$$q(\boldsymbol{x}_k) - q_* = \min_{P_k \mid \deg(P_k) < k} (1/2) \sum_{i=1}^{n} (\bar{b}_i^2/\lambda_i)(\lambda_i P_k(\lambda_i) - 1)^2.$$

Here, $\deg(P_k)$ is the degree of P_k, the largest j such that t^j features with a nonzero coefficient.

3. **[3 points]** Argue that this means that the error $q(\boldsymbol{x}_k) - q_*$ is bounded in terms of a polynomial P of degree $\leq k$, such that $P(0) = -1$ (equivalent: such that $P(0) = 1$). The existence of such a polynomial which is small on *all* λ_i, implies that the error is small. Prove that if $\{\lambda_1, \ldots, \lambda_n\} = \{\kappa_1, \ldots, \kappa_k\}$, *i.e.* if $\boldsymbol{A}$ has no more than k different eigenvalues, then $\boldsymbol{x}_k = \boldsymbol{x}_*$.
Remark: Using Chebishev polynomials, $\min_{P_k} \max_{t \in [\lambda_{\min}, \lambda_{\max}]} P_k(t)$ can be determined for $0 < \lambda_{\min} \leq \lambda_{\max}$, which leads to the worst-case error bound

$$q(\boldsymbol{x}_k) - q_* \leq \left(\frac{\rho - 1}{\rho + 1} \right)^k, \quad \rho = \sqrt{\lambda_{\max}/\lambda_{\min}}.$$

Therefore, we should aim for well-conditioned matrices $\boldsymbol{A}$ ($\lambda_{\max}/\lambda_{\min}$ small), which is what many preconditioning stategies try to do. On the other hand, if the spectrum of $\boldsymbol{A}$ comes in separate clusters, this bound is overly pessimistic.