lab 3 - Bayesian PCA

Unknown Author

November 28, 2013

1 Lab 3: Bayesian PCA

Machine Learning: Principles and Methods, November 2013

- The lab exercises should be made in groups of three people, or at least two people.
- The deadline is Friday, 13 December, 23:59.
- Assignment should be sent to T.S.Cohen at uva dot nl (Taco Cohen). The subject line of your email should be "[MLPM2013] lab#_lastname1_lastname2_lastname3".
- Put your and your teammates' names in the body of the email
- Attach the .IPYNB (IPython Notebook) file containing your code and answers. Naming of the file follows the same rule as the subject line. For example, if the subject line is "[MLPM2013] lab01_Kingma_Hu", the attached file should be "lab01_Kingma_Hu.ipynb". Only use underscores ("_") to connect names, otherwise the files cannot be parsed.

Notes on implementation:

- You should write your code and answers in an IPython Notebook: http://ipython.org/notebook.html. If you have problems, please contact us.
- Among the first lines of your notebook should be "%pylab inline". This imports all required modules, and your plots will appear inline.
- NOTE: test your code and make sure we can run your notebook / scripts!

Introduction

In this lab assignment, we will implement a variational algorithm for Bayesian PCA. Unlike regular PCA based on maximization of retained variance or minimization of projection error (see Bishop, 12.1.1 and 12.1.2), probabilistic PCA defines a proper density model over observed and latent variables. We will work with a fully Bayesian model this time, which is to say that we will put priors on our parameters and will be interested in learning the posterior over those parameters. Bayesian methods are very elegant, but require a shift in mindset: we are no longer looking for a point estimate of the parameters (as in maximum likelihood or MAP), but for a full posterior distribution.

The integrals involved in a Bayesian analysis are usually analytically intractable, so that we must resort to approximations. In this lab assignment, we will implement the variational method described in Bishop99. Chapters 10 and 12 of the PRML book contain additional material that may be useful when doing this exercise.

- [Bishop99] Variational Principal Components, C. Bishop, ICANN 1999 - http://research.microsoft.com/pubs/67241/bishop-vpca-icann-99.pdf

Below, you will find some code to get you started.

In [8]:
```
%pylab inline
import scipy.special as sp

class BayesianPCA(object):

    def __init__(self, d, N, a_alpha=10e-3, b_alpha=10e-3, a_tau=10e-3, b_tau=10e-3, b
        """
        """
        self.d = d # number of dimensions
        self.N = N # number of data points

        # Hyperparameters
        self.a_alpha = a_alpha
        self.b_alpha = b_alpha
        self.a_tau = a_tau
        self.b_tau = b_tau
        self.beta = beta

        # Variational parameters
        self.means_z = np.random.randn(d, N) # called x in bishop99
        self.sigma_z = np.random.randn(d, d)
        self.mean_mu = np.random.randn(d, 1)
        self.sigma_mu = np.random.randn(d, d)
        self.means_w = np.random.randn(d, d)
        self.sigma_w = np.random.randn(d, d)
        self.a_alpha_tilde = np.abs(np.random.randn(1))
        self.bs_alpha_tilde = np.abs(np.random.randn(d, 1))
        self.a_tau_tilde = np.abs(np.random.randn(1))
        self.b_tau_tilde = np.abs(np.random.randn(1))

    def __update_z(self, X):
        pass

    def __update_mu(self):
        pass

    def __update_w(self, X):
        pass

    def __update_alpha(self):
        pass

    def __update_tau(self, X):
        pass

    def L(self, X):
        L = 0.0
        return L

    def fit(self, X):
        pass
```

```
Populating the interactive namespace from numpy and matplotlib
```

1. The Q-distribution (5 points)

In variational Bayes, we introduce a distribution $Q(\Theta)$ over parameters / latent variables in order to make inference tractable. We can think of Q as being an approximation of a certain distribution. What function does Q approximate, $p(D|\Theta)$, $p(\Theta|D)$, $p(D,\Theta)$, $p(\Theta)$, or $p(D)$, and how do you see that from the equation $\ln p(D) = \mathcal{L}(Q) + \mathrm{KL}(Q||P)$?

2. The mean-field approximation (15 points)

Equation 13 from [Bishop99] is a very powerful result: assuming only that $Q(\Theta)$ factorizes in a certain way (no assumptions on the functional form of the factors Q_i!), we get a set of coupled equations for the Q_i.

However, the expression given in eq. 13 for Q_i contains a small mistake. Starting with the expression for the lower bound $\mathcal{L}(Q)$, derive the correct expression (and include your derivation). You can proceed as follows: first, substitute the factorization of Q (eq. 12) into the definition of $\mathcal{L}(Q)$ and separate $\mathcal{L}(Q)$ into Q_i-dependent and Q_i-independent terms. At this point, you should be able to spot the expectations $\langle\cdot\rangle_{k\neq i}$ over the other Q-distributions that appear in Bishop's solution (eq. 13). Now, keeping all $Q_k, k \neq i$ fixed, maximize the expression with respect to Q_i. You should be able to spot the form of the optimal lnQ_i, from which Q_i can easily be obtained.

3. The log-probability (10 points)

Write down the log-prob of data and parameters, $\ln p(\mathbf{X}, \mathbf{Z}, \mathbf{W}, \alpha, \tau, \mu)$, in full detail (where $\mathbf{X}$ are observed, $\mathbf{Z}$ is latent; this is different from [Bishop99] who uses $\mathbf{T}$ and $\mathbf{X}$ respectively, but $\mathbf{X}$ and $\mathbf{Z}$ are consistent with the PRML book and are more common nowadays). Could we use this to assess the convergence of the variational Bayesian PCA algorithm? If yes, how? If no, why not?

4. The lower bound $\mathcal{L}(Q)$ (25 points)

Derive an expression for the lower bound $\mathcal{L}(Q)$ of the log-prob $\ln p(X)$ for Bayesian PCA, making use of the factorization (eq. 12) and the form of the Q-distributions (eq. 16-20) as listed in [Bishop99]. Show your steps. Implement this function.

The following result may be useful:

For $x \sim \Gamma(a, b)$, we have $\langle \ln x \rangle = \ln b + \psi(a)$, where $\psi(a) = \frac{\Gamma'(a)}{\Gamma(a)}$ is the digamma function (which is implemented in numpy.special).

5. Optimize variational parameters (50 points)

Implement the update equations for the Q-distributions, in the __update_XXX methods. Each update function should re-estimate the variational parameters of the Q-distribution corresponding to one group of variables (i.e. either Z, μ, W, α or τ).

6. Learning algorithm (10 points)

Implement the learning algorithm described in [Bishop99], i.e. iteratively optimize each of the Q-distributions holding the others fixed.

What would be a good way to track convergence of the algorithm? Implement your suggestion.

Test the algorithm on some test data drawn from a Gaussian with different variances in orthogonal directions.

7. PCA Representation of MNIST (10 points)

Download the MNIST dataset from here http://deeplearning.net/tutorial/gettingstarted.html (the page contains python code for loading the data). Run your algorithm on (part of) this dataset, and visualize the results.