CSC2535 Assignment 2
Due Feb 27, 1.00pm at START of class

Iain Murray, Geoffrey Hinton and Ruslan Salakhutdinov

February 14, 2008

Late assignments will have 25% subtracted from the total out of which they are graded for each day or part of a day that they are late. They will be more than one day late if they are not slipped under my office door (LP 290G) before 1.00pm the next day.

1 Simple Monte Carlo

(a) (*1 point*) Write some Matlab code to draw independent samples from a zero-mean, unit variance Gaussian distribution, truncated to lie within [-1,1]. That is, sample from

$$p(x) \propto \begin{cases} e^{-x^2/2} & -1 < x < 1 \\ 0 & \text{otherwise.} \end{cases}$$

There is more than one sensible approach to answering this question. You may use any of the functions that ship with Matlab.

Assume that we are interested in averages under a complex distribution:

$$p(x) = \frac{\tilde{p}(x)}{\mathcal{Z}_p}.$$

As we are unable to sample directly from $p(x)$ we decide to use a more tractable distribution:

$$q(x) = \frac{\tilde{q}(x)}{\mathcal{Z}_q}.$$

We need to be able to sample from q; we usually know $\mathcal{Z}_q$.

(b) (*1 point*) Write down an expression for the normalizing constant $\mathcal{Z}_p$.

(c) (*1 point*) As discussed in the lectures, the ratio of normalizing constants, $\mathcal{Z}_p/\mathcal{Z}_q$, is needed by importance sampling. Rewrite $\mathcal{Z}_p/\mathcal{Z}_q$ as an expectation under $q(x)$, involving only $\tilde{p}(x)$, $q(x)$ and $\tilde{q}(x)$. From this write down an unbiased, simple Monte Carlo estimate for the quantity $\mathcal{Z}_p/\mathcal{Z}_q$.

We now consider an example distribution over a D-dimensional hypercube:

$$\tilde{p}(\mathbf{x}) = \begin{cases} 9/(0.1)^D & \text{if } 0.9 < x_d < 1 \text{ for all } d \\ 1/(1-(0.1)^D) & \text{if } 0 < x_d < 1 \text{ for all } d, \text{ and if } x_d \leq 0.9 \text{ for any } d \\ 0 & \text{otherwise} \end{cases}$$

This distribution has a region of high probability in one corner of the cube, which contains $9/10$ of the probability mass. Distributions from probabilistic models often have extreme peaks: as soon as a few data points are observed, many parameters usually become very improbable.

(d) (*2 points*) From your answer to (c), we could find an unbiased estimate of $\mathcal{Z}_p$ by drawing samples uniformly inside the unit hypercube:

$$q(\mathbf{x}) = \begin{cases} 1 & \text{if } 0 < x_d < 1 \text{ for all } d \\ 0 & \text{otherwise} \end{cases}, \qquad (\mathcal{Z}_q = 1 \text{ is known}).$$

Keen to obtain accurate answers, we draw $S = 10,000$ samples from $q(\mathbf{x})$. Work out the probability that at least one of these samples lands in the corner of high probability as a function of D. Evaluate this probability for $D = 1, 2, 4, 8$. Explain how this will affect your estimates of $\mathcal{Z}_p$ for high-dimensional problems. Comment on the description that the estimator for $\mathcal{Z}_p$ is "unbiased".

2 Markov Chain Monte Carlo

(a) (*2 points*) Write a Matlab function

```
function samples = dumb_metropolis(x_start, p_tilde, sigma, steps)
```

that runs a Metropolis Markov chain starting at a vector `x_start` for `steps` iterations. The proposal distribution should be a simple spherical Gaussian with width `sigma` centered on the current location (hence *dumb* Metropolis). A function that evaluates a quantity proportional to the target distribution is passed in as `p_tilde`.

(b) (*2 points*) Test your function on a simple distribution. You may use

```
p_tilde = @(x) (x<1)*(x>-1)*exp(-0.5*x*x);
```

to sample from the truncated distribution from the first question. Or a more interesting distribution of your choice, for example one with more dimensions. Exercise 29.17 from Mackay's book

`http://www.inference.phy.cam.ac.uk/mackay/itila/book.html`

offers an *optional* small statistical problem.

(c) (*1 point*) If a Markov chain is initialized arbitrarily, the distribution over the position at each iteration, $q_t(\mathbf{x})$ is biased away from the target stationary distribution, $p(\mathbf{x})$. Consider a single Gibbs sampling update, which resamples one component of $\mathbf{x}$ from its conditional. Show that the distribution after this update, $q_{t+1}(\mathbf{x})$, is never further from $p(\mathbf{x})$ in KL divergence (measured both ways around) than the distribution before the update, $q_t(\mathbf{x})$.

3 Alternative ways of learning RBM's

(*10 points*) Empirically evaluate the following three methods of learning a restricted Boltzmann machine:

a) **CD1** This starts at the data and uses three half steps of alternating Gibbs to update the hidden units, then the visible units, then the hidden units again. So it uses one full step of alternating Gibbs to produce the "negative data" that are used for approximating the gradient of the partition function (see lecture notes).

b) **CD10** This uses 10 full steps of alternating Gibbs to produce the negative data.

c) **Persistent CD** After each weight update this runs one full step of alternating Gibbs starting with the current negative data to produce new negative data. Before learning starts the negative data is initialized at random data. When learning involves multiple mini-batches of data, the negative data is started at the negative data that was used for the previous mini-batch, so only one mini-batch of negative data is used.

To evaluate these three methods, you should find a dataset of binary data that is highly structured (so there is a significant amount of structure for the RBM to learn) and is not too high-dimensional (so you dont need too many parameters). For example, an RBM with 100 hidden units trained on 1000 binarized 16×16 images of handwritten twos would do, but you should attempt to find some other dataset. The hope is that different students will get results on different datasets. There is no need to try more than one dataset unless you decide that the first one you tried is inappropriate.

Some simple code for training an RBM with CD1 can be found at

```
http://www.cs.toronto.edu/~hinton/code/rbm.m
```

The learning rate and weightcost in this code may need to be changed for your dataset.

To compute the log probability that a learned RBM assigns to your test data, you will need to compute the negative free energy (the goodness) that the model assigns to each test vector and subtract the log of the partition function for that RBM. Matlab code for computing the log of the partition function is provided at:

```
http://www.cs.toronto.edu/~hinton/code/partition
```

The files are:

```
free_energy.m -- computes free energy for use in RBM_partition.m
logdiff.m and logsum.m  -- used for computing log(Z+-3std)
RBM_partition.m -- is the main engine for calculating Z
demo.m -- loads some data and an RBM model trained with CD25,
runs RBM_partition.m and returns log(Z),log(Z+3std),log(Z-3std).
Occasionally you may get logZZ_down=0.000000.
If this happens just try again.
```

Computing the partition function accurately can take a long time, so you may only be able to do it a few times while learning an RBM. You can get a noisy estimate more quickly by using fewer intermediate distributions, but 5000 is already a very small number.

We do not expect you to do a polished piece of research. It is sufficient to find a single sensible dataset, train the three methods using a sensible number of hidden units, and compare the log probabilities they assign to your test data at a few sensibly chosen points during the learning. Question 3 should take you between 8 and 16 hours. Your answer should consist of up to one page describing your dataset, about one page describing your experiments, and one or two graphs showing how the log probabilities that the three methods assigned to the training and test data varied as they learned.