CSC2535 2011
ASSIGNMENT 2

Geoffrey Hinton, Ilya Sutskever and Ryan Adams

February 17, 2011

Assignments are due by 3.00pm on wednesday March 2. Printed copies are required. This assignment is worth 20% of the final grade and is marked out of 40 points.

Late assignments will have 20% 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 3.10pm the next day.

Problem One

One of the critical properties of a valid Markov chain Monte Carlo transition operator is that it leave the target distribution invariant. That is, if the target distribution is $f_X(x)$, then the operator $T(x' \leftarrow x)$ must satisfy

$$f_X(x') = \int f_X(x)\, T(x' \leftarrow x)\, \mathrm{d}x.$$

A stronger property that is a sufficient condition for $T(x' \leftarrow x)$ to leave $f_X(x)$ invariant is *detailed balance*:

$$f_X(x')\, T(x \leftarrow x') = f_X(x)\, T(x' \leftarrow x).$$

Prove that the Metropolis–Hastings algorithm satisfies detailed balance. **(5 points)**

Problem Two

Here is a simple hierarchical model that looks like a ten-dimensional "funnel":

$$\begin{aligned} v &\sim \mathcal{N}(\mu = 0, \sigma^2 = 3^2) \\ x_k &\sim \mathcal{N}(\mu = 0, \sigma^2 = e^v) \text{ for } k = 1, 2, \ldots 9 \end{aligned}$$

which leads to a joint distribution

$$p(v, \{x_k\}_{k=1}^9) = \mathcal{N}(v \,|\, 0, 9) \prod_{k=1}^{9} \mathcal{N}(x_k \,|\, 0, e^v). \tag{1}$$

- Out of 50,000 samples from this joint distribution, about how many will have $v < -5$? **(1 point)**

- Write a function `funnel_directed` that generates 50,000 samples from this joint distribution, exploiting knowledge of the hierarchical model. Plot a histogram of v. How many samples had $v < -5$? **(1 point)**
- Pretend that you don't know about the directed structure and that you just have the joint distribution in Equation 1. Write a function `funnel_mh` that generates 50,000 samples from this joint distribution using Metropolis–Hastings. Use a ten-dimensional zero-mean Gaussian proposal with identity covariance. Start from $v = 0$ and $x_k = 1$. Plot a histogram of v. How many samples had $v < -5$? **(2 points)**
- Write a function `funnel_slice` that implements slice sampling for the joint distribution in Equation 1. Start from the same initialization as the Metropolis–Hastings sampler. Plot a histogram of v. How many samples had $v < -5$? **(2 points)**

Problem 3

(8 points)

Assume that we have a family of probability distributions $P(V, H)$ (e.g., all possible instantiations of a Boltzmann Machine with a fixed architecture), and that we wish to maximize the log probability of a single observed datapoint, $\log P(V)$. The variational approximation provides a lower-bound to $\log P(V)$ for each non-zero distribution $Q(H)$:

$$\begin{aligned}
\log P(V) &= \log \sum_H P(V, H) \\
&= \log \sum_H Q(H) \frac{P(V, H)}{Q(H)} \\
&\geq \sum_H Q(H) \log \frac{P(V, H)}{Q(H)} \\
&=_{\text{def}} \mathcal{L}(P, Q)
\end{aligned}$$

where we used Jensen's inequality.

Furthermore, a manipulation of the equations suggests the equality

$$\log P(V) - \mathcal{L}(P, Q) = KL(Q(\cdot) \| P(\cdot|V)),$$

implying that the lower bound $\mathcal{L}(P, Q)$ reaches equality when $Q(H) = P(H|V)$. Thus a maximization of $\mathcal{L}(P, Q)$ with respect to Q is equivalent to computing the posterior $P(H|V)$, and a partial maximization of $\mathcal{L}(P, Q)$ with respect to Q lets us obtain an approximation to the posterior.

The EM algorithm maximizes $\mathcal{L}$ with respect to Q (the E-step) and with respect to P (the M-step) iteratively, and is useful in because both operations are tractable in many models of interest. By increasing $\mathcal{L}$, we are maximizing a lower bound to the log probability $\log P(V)$ which is a proxy to our real desire — to maximize $\log P(V)$.

The key principle of the variational methods is that, if we wish to maximize $\log P(V)$ for a highly

complex model where full maximization of $\mathcal{L}(P, Q)$ with respect to Q is infeasible [1][2], we could simply restrict the set of allowable Q, denoted $\mathcal{Q}$, so that the problem

$$\max_{Q \in \mathcal{Q}} \mathcal{L}(P, Q) \tag{2}$$

is feasible. By restricting Q to be from $\mathcal{Q}$, learning proceeds by repeated partial maximizations of $\mathcal{L}$ with respect to Q and with respect to P, which maximizes a lower bound to the quantity we care about. Of course, by not using the optimal Q we learn suboptimal models, but this is much better than not using the model at all.

We will only consider Q's that can be represented in the following manner:

$$Q(H) = Q(H_1, \ldots, H_N) = \prod_j Q_j(H_j) \tag{3}$$

Here each Q_j is a simple distribution over a small number of configurations. The hope is that maximizing $\mathcal{L}(P, Q)$ with respect to the parameters of Q_j is going to be a much easier task than an unrestricted maximization with respect to Q, and the goal of the assignment is to verify that this is the case.

This exercise is on the Boltzmann Machine (BM), which defines a distribution over pairs of binary vectors (V, H) by the equation

$$P(V, H) = \frac{\exp(X^\top W X)}{Z} \tag{4}$$

where $X = (V, H)$ is the concatenation of V and H and W are the parameters of P. Given that we restrict ourselves to the distribution Q expressible by eq. 3 and that each H_j is binary, $Q_j(H_j) = q_j^{H_j} \cdot (1 - q_j)^{1 - H_j}$ for some q_j, so Q is represented with the set of scalars $\{q_j\}$:

$$Q(H) = \prod_j q_j^{H_j} \cdot (1 - q_j)^{1 - H_j} \tag{5}$$

1. Your first task is to show how to maximize $\mathcal{L}(P, Q)$ with respect to the parameters of Q (which are the q_j). Specifically, given V, obtain the value of q_j that maximizes $\mathcal{L}(P, Q)$ while all other $q_{i \neq j}$ are held constant. Use single-variable calculus and set a derivative to zero. Describe a very simple method for finding a sensible approximation $Q(H)$ to $P(H|V)$. **(2 points)**

2. Compare the optimal q_j to the probability $P(H_j = 1 | V, H_1, H_2, \ldots, H_{j-1}, H_{j+1}, \ldots)$. **(1 point)**

3. The learning rule of the Boltzmann Machine has the form

$$\Delta W_{ij} \propto \langle X_i X_j \rangle_{P(H|V)} - \langle X_i X_j \rangle_{P(V,H)} \tag{6}$$

   where V is a data point and $X = (V, H)$. Derive the gradient of $\mathcal{L}(P, Q)$ with respect to parameters W. **(1 point)**

[1] Optimizing $\mathcal{L}$ with respect to Q is challenging *only* due to the large size of H — for instance, if H is the set of large binary vectors, there are exponentially many configurations of H. Since a distribution $Q(H)$ assigns a probability to every configuration, a generic Q is represented with exponentially many real numbers, so there is no hope in solving an optimization problem that involves objects of such size. In contrast, simpler models, such as the Mixture of Gaussians, has a very limited number of configurations of the hidden state (the number of Gaussians, k), so it is possible to write down the optimal Q, which is the posterior $P(H|V)$. But if we chose to approximately represent $Q(H)$ with a set of samples, we could maximize $\mathcal{L}$ with respect to Q by simply running a Markov chain with $P(H|V)$ as its equilibrium distribution, but this approach is too slow to be useful.

[2] Maximizing $\mathcal{L}(P, Q)$ with respect to P is trivial for directed graphical model but is nontrivial for Boltzmann Machines.

4. Briefly explain how the algorithm for training Deep Boltzmann Machines maximizes $\mathcal{L}(P, Q)$ with respect to Q and P. How does it handle the partition function problem? **(2 points)**

5. We could use the method of part 1 to approximate $P(V, H)$ with a factorial distribution $Q(V, H)$ and use $\langle X_i X_j \rangle_{Q(V,H)}$ in place of the negative statistics of the gradient. Briefly explain why such an update will not increase a variational lower bound to the log probability. (Hint: the log partition function $\log Z$ can be lower-bounded using a variational lower bound $\mathcal{L}(Q(V, H))$, just like $\log P(V)$). **(2 points)**

Problem 4

(21 points)

In lecture 6b, we saw that the multiplicative RNN generates text with a significant amount of linguistic structure, indicating that the trained RNN "knows" at least a few aspects the English language. But what does its knowledge consist of? While it is impossible to completely describe its knowledge in a comprehensible manner, we can fruitfully ask questions of the form "does the RNN know X?", and answer them using empirical evidence.

Your goal in this problem is to:

1. Propose hypotheses about the knowledge of the RNN, and

2. Design and execute experiments that either support or disprove the proposed hypotheses.

More precisely, you should report exactly **three** experiments. At least one experiment should evaluate a syntax-related hypothesis and at least one should evaluate a semantics-related hypothesis. At the same time, at least one experiment should corroborate its hypothesis and at least one should refute it. So, for example, the first experiment could corroborate a syntax-related hypothesis, the second refute a semantics-related hypothesis, and the third could be anything you like.

You will need to provide reasonably convincing evidence for the validity or the invalidity of the proposed hypothesis. To do so, you will be provided with two functions: one which calculates the log probability of any sequence of characters, and another that samples from the distribution defined by the RNN conditioned on an initial string of characters.

To see what we have in mind, a very simple semantic hypothesis is the claim that the word "car" and the word "truck" tend to co-occur in the sentences generated by the RNN. This could be demonstrated by showing that the probability of the substring "truck" is, on average, greater in sentences that contain the word "car" than in average sentences. You could proceed by collecting 1000 random sentences from the internet containing the word "car", and 1000 random sentences containing both "car" and "truck", and measure the average log probability of the character-string "car" in the two groups of sentences.

An example of a syntactic hypothesis is the claim that the RNN understands plurality, which could be demonstrated by showing that the string "X eats" is more likely than the string "X eat "[3] when X represents a singular object, such as a given name ("Bob"), and vice-versa for plural Xs. This specific claim (about "eats") should be evaluated using a moderate number of sentences (500). The experiment would more interesting if it investigated the above for a number of verbs.

It is important that the hypothesis is not trivially true ("the RNN knows that a full-stop is always followed by a space") nor trivially false ("the RNN does not understand abstract metaphors"). Be

[3]Note that the two strings are of the same length; it is important that, whenever you are comparing the log probability of strings, that they are of the same length, simply because longer strings are less probable.

imaginative when designing the hypotheses. You may consider hypotheses related to numbers, geography, physics, or any other kind of common knowledge. The experimental proof or a disproof of a hypothesis should be convincing to a reader knowledgeable in statistics or machine learning. Given that the assignment is about formulating interesting hypotheses, do not use the example hypotheses in your experiments.

For each experiment, there are **3 points** for the choice of the hypothesis, and **4 points** for the quality of the experiments.

Start thinking about the hypotheses right away. We intend to write a technical report with all the experiments we find interesting, and you will be an author on the technical report if we choose yours.

Note: given that evaluating a sentence takes some time, do not evaluate more than 10,000 sentences for each experiment, so that students without access to a large computer cluster are not disadvantaged.

Another note: It is safer to evaluate a claim about the RNN's knowledge by evaluating the log probabilities of sentences you have collected yourself rather than relying on sampling from the RNNs model. Unconstrained samples are less convincing because the RNN exhibits a modest amount of overfitting, so the RNN may produce samples that are similar to the specific subset of wikipedia that was used for training. This point is illustrated by an extreme model that simply memorizes the whole training set. Its samples will be extremely good but it has definitely not learned anything of value from the data.

A final note: the code provides you with three RNNs — one trained on Wikipedia, one on Machine learning papers, and one on articles from the New York Times. You may use any RNN you like, although the Wikipedia RNN is preferred. It is very important that you *do not* test the Wikipedia RNN on sentences taken from the Wikipedia, because there is a possibility that the RNN has already encountered these sentences during training, which can make it seem more knowledgeable than it really is. The same applies to the other RNNs.

The code for problem 4 is obtained in the following URL:

`www.cs.utoronto.ca/~ilya/a2_rnn.tar.gz`

You will need numpy and python. Ilya is the TA for this assignment, so if you have difficulty with numpy and python, ask him what to do. His office hours are on the main webpage for csc2535.