

18.4 Score Matching and Ratio Matching

Score matching (Hyvärinen, 2005) provides another consistent means of training a model without estimating Z or its derivatives. The name *score matching* comes from terminology in which the derivatives of a log density with respect to its argument, $\nabla_{\mathbf{x}} \log p(\mathbf{x})$, are called its **score**. The strategy used by score matching is to minimize the expected squared difference between the derivatives of the model's log density with respect to the input and the derivatives of the data's log density with respect to the input:

$$L(\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{2} \|\nabla_{\mathbf{x}} \log p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta}) - \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})\|_2^2, \quad (18.22)$$

$$J(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} L(\mathbf{x}, \boldsymbol{\theta}), \quad (18.23)$$

$$\boldsymbol{\theta}^* = \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}). \quad (18.24)$$

This objective function avoids the difficulties associated with differentiating

the partition function Z because Z is not a function of \mathbf{x} and therefore $\nabla_{\mathbf{x}}Z = 0$. Initially, score matching appears to have a new difficulty: computing the score of the data distribution requires knowledge of the true distribution generating the training data, p_{data} . Fortunately, minimizing the expected value of $L(\mathbf{x}, \boldsymbol{\theta})$ is equivalent to minimizing the expected value of

$$\tilde{L}(\mathbf{x}, \boldsymbol{\theta}) = \sum_{j=1}^n \left(\frac{\partial^2}{\partial x_j^2} \log p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta}) + \frac{1}{2} \left(\frac{\partial}{\partial x_j} \log p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta}) \right)^2 \right), \quad (18.25)$$

where n is the dimensionality of \mathbf{x} .

Because score matching requires taking derivatives with respect to \mathbf{x} , it is not applicable to models of discrete data but the latent variables in the model may be discrete.

Like pseudolikelihood, score matching only works when we are able to evaluate $\log \tilde{p}(\mathbf{x})$ and its derivatives directly. It is not compatible with methods that provide only a lower bound on $\log \tilde{p}(\mathbf{x})$, because score matching requires the derivatives and second derivatives of $\log \tilde{p}(\mathbf{x})$, and a lower bound conveys no information about its derivatives. This means that score matching cannot be applied to estimating models with complicated interactions between the hidden units, such as sparse coding models or deep Boltzmann machines. While score matching can be used to pretrain the first hidden layer of a larger model, it has not been applied as a pretraining strategy for the deeper layers of a larger model. This is probably because the hidden layers of such models usually contain some discrete variables.

While score matching does not explicitly have a negative phase, it can be viewed as a version of contrastive divergence using a specific kind of Markov chain (Hyvärinen, 2007a). The Markov chain in this case is not Gibbs sampling, but rather a different approach that makes local moves guided by the gradient. Score matching is equivalent to CD with this type of Markov chain when the size of the local moves approaches zero.

Lyu (2009) generalized score matching to the discrete case (but made an error in the derivation that was corrected by Marlin *et al.* [2010]). Marlin *et al.* (2010) found that **generalized score matching** (GSM) does not work in high-dimensional discrete spaces where the observed probability of many events is 0.

A more successful approach to extending the basic ideas of score matching to discrete data is **ratio matching** (Hyvärinen, 2007b). Ratio matching applies specifically to binary data. Ratio matching consists of minimizing the average over

examples of the following objective function:

$$L^{(\text{RM})}(\mathbf{x}, \boldsymbol{\theta}) = \sum_{j=1}^n \left(\frac{1}{1 + \frac{p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta})}{p_{\text{model}}(f(\mathbf{x}, j); \boldsymbol{\theta})}} \right)^2, \quad (18.26)$$

where $f(\mathbf{x}, j)$ returns \mathbf{x} with the bit at position j flipped. Ratio matching avoids the partition function using the same trick as the pseudolikelihood estimator: in a ratio of two probabilities, the partition function cancels out. [Marlin *et al.* \(2010\)](#) found that ratio matching outperforms SML, pseudolikelihood and GSM in terms of the ability of models trained with ratio matching to denoise test set images.

Like the pseudolikelihood estimator, ratio matching requires n evaluations of \tilde{p} per data point, making its computational cost per update roughly n times higher than that of SML.

As with the pseudolikelihood estimator, ratio matching can be thought of as pushing down on all fantasy states that have only one variable different from a training example. Since ratio matching applies specifically to binary data, this means that it acts on all fantasy states within Hamming distance 1 of the data.

Ratio matching can also be useful as the basis for dealing with high-dimensional sparse data, such as word count vectors. This kind of data poses a challenge for MCMC-based methods because the data is extremely expensive to represent in dense format, yet the MCMC sampler does not yield sparse values until the model has learned to represent the sparsity in the data distribution. [Dauphin and Bengio \(2013\)](#) overcame this issue by designing an unbiased stochastic approximation to ratio matching. The approximation evaluates only a randomly selected subset of the terms of the objective and does not require the model to generate complete fantasy samples.

See [Marlin and de Freitas \(2011\)](#) for a theoretical analysis of the asymptotic efficiency of ratio matching.

18.5 Denoising Score Matching

In some cases we may wish to regularize score matching, by fitting a distribution

$$p_{\text{smoothed}}(\mathbf{x}) = \int p_{\text{data}}(\mathbf{y})q(\mathbf{x} | \mathbf{y})d\mathbf{y} \quad (18.27)$$

rather than the true p_{data} . The distribution $q(\mathbf{x} | \mathbf{y})$ is a corruption process, usually one that forms \mathbf{x} by adding a small amount of noise to \mathbf{y} .