Sparse coding and 'ICA'

Bruno A. Olshausen

November 18, 2008

Sparse coding

Sparse coding is based on the linear generative model:

$$\mathbf{x} = \mathbf{A}\,\mathbf{s} + \mathbf{n} \tag{1}$$

where $\mathbf{x}$ is a data vector (e.g., pixels from an image patch, or a sound waveform), $\mathbf{A}$ is a matrix of 'features' or basis functions (each column is a different basis function), $\mathbf{s}$ is a vector of coefficients, and $\mathbf{n}$ is a vector of Gaussian 'noise' (typically i.i.d.). The aim here is to find a set of basis functions $\mathbf{A}$ which allow the data $\mathbf{x}$ to be represented as a set of sparse coefficient values $\mathbf{s}$ (lots of zeros on average). Usually the basis function matrix is overcomplete (more columns than rows), and the noise term is small relative to $\mathbf{A}\,\mathbf{s}$ and is included to account for residual structure that is not well described by the basis function model.

The model distribution is given by

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{s})\, p_s(\mathbf{s})\, d\mathbf{s} \tag{2}$$

If we assume Gaussian i.i.d. noise $\mathbf{n}$ with variance σ_n^2 then the conditional distribution $p(\mathbf{x}|\mathbf{s})$ is given by

$$p(\mathbf{x}|\mathbf{s}) \propto e^{-\frac{|\mathbf{x}-\mathbf{A}\,\mathbf{s}|^2}{2\,\sigma_n^2}} \tag{3}$$

The prior $p_s(\mathbf{s})$ is parameterized as

$$p_s(\mathbf{s}) \propto e^{-\sum_i C(s_i)} \tag{4}$$

where $C(\,)$ specifies the shape of the distribution over each element s_i and is chosen to correspond to a 'sparse prior'—i.e., peaked at zero with heavy tails. For example, choosing $C(s_i) = |s_i|$ corresponds to a Laplacian prior, which has the desired shape. Note that we are also assuming a factorial prior on $\mathbf{s}$ for now, but that is mostly for convenience—it is not required for sparse coding in general.

Learning the basis function matrix $\mathbf{A}$ is accomplished by maximizing the average log-likelihood of the model via gradient ascent:

$$\Delta\mathbf{A} \propto \frac{\partial}{\partial\mathbf{A}} \langle \log p(\mathbf{x}) \rangle \tag{5}$$

where $\langle\,\rangle$ denotes averaging over the entire set of data vectors $\mathbf{x}$. This yields the learning rule:

$$\Delta\mathbf{A} \propto \left\langle \int [\mathbf{x} - \mathbf{A}\,\mathbf{s}]\,\mathbf{s}^T\, p(\mathbf{s}|\mathbf{x})\, d\mathbf{s} \right\rangle \tag{6}$$

Thus, learning the basis functions requires us to sample from the posterior $p(\mathbf{s}|\mathbf{x})$ and accumulate the average of the outer product $[\mathbf{x} - \mathbf{A}\,\mathbf{s}]\,\mathbf{s}^T$ from these samples.

Sampling from the posterior can be very slow, so in practice we take a single sample at the posterior maximum:

$$\hat{\mathbf{s}} = \arg\max_{\mathbf{s}} p(\mathbf{s}|\mathbf{x})$$

which is equivalent to minimizing the negative log-posterior:

$$\begin{aligned} \hat{\mathbf{s}} &= \arg\min_{\mathbf{s}} -\log p(\mathbf{s}|\mathbf{x}) \\ &= \arg\min_{\mathbf{s}} \left[\frac{\lambda_n}{2} |\mathbf{x} - \mathbf{A}\,\mathbf{s}|^2 + \sum_i C(s_i) \right] \end{aligned} \tag{7}$$

where $\lambda_n = 1/\sigma_n^2$. Note that there is a simple, intuitive interpretation of equation (7): minimize squared error of the reconstruction (first term) plus a cost function on coefficient activity (second term). Minimizing this function can be accomplished via gradient descent, yielding

$$\dot{\mathbf{s}} \propto \lambda_n\,[\mathbf{b} - \mathbf{G}\,\mathbf{s}] - \mathbf{z}(\mathbf{s}) \tag{8}$$

where $\mathbf{b} = \mathbf{A}^T\,\mathbf{x}$, $\mathbf{G} = \mathbf{A}^T\mathbf{A}$, and $\mathbf{z}(\mathbf{s})$ has elements

$$z_i = C'(s_i) \tag{9}$$

The solution $\hat{\mathbf{s}}$ is obtained when $\dot{\mathbf{s}} = 0$. Note that this differential equation can be implemented as a recurrent neural network with feedforward excitation $\mathbf{b}$, recurrent inhibition $\mathbf{G}\,\mathbf{s}$, and non-linear self-inhibition $\mathbf{z}(\mathbf{s})$.[1]

The overall learning procedure thus involves a fast inner loop in which the coefficients $\hat{\mathbf{s}}$ are computed for each data vector $\mathbf{x}$ via equation 8, and a slower outer loop in which the basis functions are adapted to the statistics of the entire dataset. The latter part is done by replacing the posterior $p(\mathbf{s}|\mathbf{x})$ in equation 6 with the posterior maximum computed in equation 8, yielding the learning rule:

$$\Delta\mathbf{A} \propto \left\langle [\mathbf{x} - \mathbf{A}\,\hat{\mathbf{s}}]\,\hat{\mathbf{s}}^T \right\rangle \tag{10}$$

This essentially amounts to Hebbian learning between the residual $[\mathbf{x} - \mathbf{A}\,\hat{\mathbf{s}}]$ and the inferred coefficients $\hat{\mathbf{s}}$.

If you want to try out this algorithm you can download the Matlab code from my webpage at `http://redwood.berkeley.edu/bruno/sparsenet`.

[1] This method has been superceded by the recently developed Locally Competitive Algorithm (LCA), which generally provides a more efficient solution that is also neurally plausible, see Rozell et al., *Neural Computation, 20*, 2526-2563.

ICA

ICA (independent component analysis), as it is typically applied, is a special case of sparse coding where the matrix $\mathbf{A}$ is square and of full rank, and $\mathbf{n} = 0$ (no noise). Thus the image model reduces to

$$\mathbf{x} = \mathbf{A}\,\mathbf{s} \tag{11}$$

Since the matrix is invertible we now have a simple way to compute the coefficients:

$$\mathbf{s} = \mathbf{A}^{-1}\mathbf{x} \tag{12}$$

In general the prior in ICA need only be non-Gaussian, although it is typically assumed to be sparse (e.g., Laplacian). As equation (12) shows, the prior now plays no role in determining the coefficients, but it does still play an important role in the learning as we shall see.

Since there is no noise in the ICA model, the conditional distribution $p(\mathbf{x}|\mathbf{s})$ collapses to a delta function

$$p(\mathbf{x}|\mathbf{s}) = \delta(\mathbf{x} - \mathbf{A}\,\mathbf{s})$$

and so the model distribution becomes

$$\begin{aligned} p(\mathbf{x}) &= \int \delta(\mathbf{x} - \mathbf{A}\,\mathbf{s})\, p_s(\mathbf{s})\, d\mathbf{s} \\ &= p_s(\mathbf{A}^{-1}\mathbf{x})/|\det \mathbf{A}| \end{aligned} \tag{13}$$

Thus, the log-likelihood of the model is now

$$\log p(\mathbf{x}) = -\sum_i C(s_i) - \log\det \mathbf{A} \tag{14}$$

where $s_i = (\mathbf{A}^{-1}\mathbf{x})_i$, as in equation (12) above. Computing the derivative with respect to $\mathbf{A}$ yields the learning rule:

$$\Delta\mathbf{A} \propto \langle [\mathbf{A}^T]^{-1}\,\mathbf{z}(\mathbf{s})\,\mathbf{s}^T - [\mathbf{A}^T]^{-1}\rangle \tag{15}$$

(Note: to obtain this result one must use the identity $\frac{\partial}{\partial A_{ij}}(\mathbf{A}^{-1})_{kl} = -(\mathbf{A}^{-1})_{ki}\,(\mathbf{A}^{-1})_{jl}$). Unfortunately this learning rule involves a matrix inverse which can make it unstable during learning. In practice one finds that the learning rule is made more efficient by pre-multiplying by $\mathbf{A}\,\mathbf{A}^T$, which cancels $[\mathbf{A}^T]^{-1}$, thus yielding the learning rule:

$$\Delta\mathbf{A} \propto \left\langle \mathbf{A}\,\mathbf{z}(\mathbf{s})\,\mathbf{s}^T - \mathbf{A}\right\rangle \tag{16}$$

(This is also known as the "natural gradient.") Since the basis functions $\mathbf{A}$ are changing slowly with respect to $\mathbf{s}$ and $\mathbf{z}$, we can bring them outside the averaging brackets, yielding

$$\Delta\mathbf{A} \propto \mathbf{A}\,\langle \mathbf{z}(\mathbf{s})\,\mathbf{s}^T\rangle - \mathbf{A} \tag{17}$$

The ICA learning algorithm (eq. 17) is simpler and faster than the sparse coding algorithm (eq. 10) because $\mathbf{s}$ can be computed from the data $\mathbf{x}$ in a single feedforward pass using equation (12). $\mathbf{z}$ is then computed directly from $\mathbf{s}$ via equation (9). The statistics $\langle \mathbf{z}\,\mathbf{s}^T\rangle$ are computed for a large chunk of data and the basis function matrix $\mathbf{A}$ is then updated using equation 17. This process is repeated until the basis functions converge to a solution.

Why 'ICA' is a misnomer

When most people say 'ICA,' they really mean *linear* ICA - i.e., using the linear generative model above. However this is a very simple model and it is not generally capable of yielding truly independent components for most real datasets (unless you are doing source separation of a small, finite number of truly independent, linearly-mixed signals). Usually the components recovered still exhibit statistical dependencies, and so one is left with the awkward task of modeling the "dependencies among the independent components."

Note that one can not even claim that ICA is trying to recover components that are "as independent as possible," because the independence assumption is just one of several properties of the ICA model. Overall there are three properties of the model which are being fit to the data:

1. linear superposition, $\mathbf{x} = \mathbf{A}\mathbf{s}$,
2. the shape of the prior over each of the components, $p_{s_i} \propto e^{C(s_i)}$, and
3. the joint prior over the entire set of components, which is factorial: $p_s(\mathbf{s}) \propto \prod_i p_{s_i}$.

The basis function matrix $\mathbf{A}$ is changing during gradient descent so as to satisfy all three of these assumptions as best as possible (as measured by the log-likelihood). To the extent any one of them is wrong, then the model will still try to satisfy the other assumptions. Thus to the extent that the data can not be described as a linear superposition of independent components, the model will still try to accommodate the shape of the prior over each component, p_{s_i}. If it can better fit the prior over each component by introducing more dependencies among them, it may do so. The final solution for the basis functions will likely be some compromise between satisfying the factorial assumption and the shape of the prior over each component. *There is nothing in the model that adjudicates in favor of the factorial assumption over the shape of the prior.*

The real power of ICA comes from the shape of the prior—i.e., the manner in which it is chosen to be non-Gaussian (positive or negative kurtosis)—rather than the fact that it is factorial per se. Indeed, some of the more recent extensions of ICA utilize sparse, non-factorial priors (e.g., Hyvarinen's "sub-space ICA"). Figuring out how to express and specify the non-factorial prior turns out to be one of the more important issues in these models.

The term 'ICA' is thus in some sense a misnomer. An alternative is to call this class of models *sparse component analysis (SCA)*, or *sparse, linear component analysis (SLiCA)*, the goal of which is to decompose data into sparse components so as to make it easier to model the dependencies among them.