Sparse Coding

Sparse coding is a class of unsupervised methods for learning sets of over-complete bases to represent data efficiently. The aim of sparse coding is to find a set of basis vectors ϕ_i such that we can represent an input vector $\mathbf{x}$ as a linear combination of these basis vectors:

$$\mathbf{x} = \sum_{i=1}^{k} a_i \phi_i$$

While techniques such as Principal Component Analysis (PCA) allow us to learn a complete set of basis vectors efficiently, we wish to learn an **over-complete** set of basis vectors to represent input vectors $\mathbf{x} \in \mathbb{R}^n$ (i.e. such that $k > n$). The advantage of having an over-complete basis is that our basis vectors are better able to capture structures and patterns inherent in the input data. However, with an over-complete basis, the coefficients a_i are no longer uniquely determined by the input vector $\mathbf{x}$. Therefore, in sparse coding, we introduce the additional criterion of **sparsity** to resolve the degeneracy introduced by over-completeness.

Here, we define sparsity as having few non-zero components or having few components not close to zero. The requirement that our coefficients a_i be sparse means that given a input vector, we would like as few of our coefficients to be far from zero as possible. The choice of sparsity as a desired characteristic of our representation of the input data can be motivated by the observation that most sensory data such as natural images may be described as the superposition of a small number of atomic elements such as surfaces or edges. Other justifications such as comparisons to the properties of the primary visual cortex have also been advanced.

We define the sparse coding cost function on a set of m input vectors as

$$\text{minimize}_{a_i^{(j)},\phi_i} \sum_{j=1}^{m} \left|\left| \mathbf{x}^{(j)} - \sum_{i=1}^{k} a_i^{(j)} \phi_i \right|\right|^2 + \lambda \sum_{i=1}^{k} S(a_i^{(j)})$$

where $S(.)$ is a sparsity cost function which penalizes a_i for being far from zero. We can interpret the first term of the sparse coding objective as a reconstruction term which tries to force the algorithm to provide a good representation of $\mathbf{x}$ and the second term as a sparsity penalty which forces our representation of $\mathbf{x}$ to be sparse. The constant λ is a scaling constant to determine the relative importance of these two contributions.

Although the most direct measure of sparsity is the "L_0" norm ($S(a_i) = \mathbf{1}(|a_i| > 0)$), it is non-differentiable and difficult to optimize in general. In practice, common choices for the sparsity cost $S(.)$ are the L_1 penalty $S(a_i) = |a_i|_1$ and the log penalty $S(a_i) = \log(1 + a_i^2)$.

In addition, it is also possible to make the sparsity penalty arbitrarily small by scaling down a_i and scaling ϕ_i up by some large constant. To prevent this from happening, we will constrain $||\phi||^2$ to be less than some constant C. The full sparse coding cost function including our constraint on ϕ is

$$\begin{array}{ll} \text{minimize}_{a_i^{(j)},\phi_i} & \sum_{j=1}^{m} \left|\left| \mathbf{x}^{(j)} - \sum_{i=1}^{k} a_i^{(j)} \phi_i \right|\right|^2 + \lambda \sum_{i=1}^{k} S(a_i^{(j)}) \\ \text{subject to} & ||\phi_i||^2 \leq C, \forall i = 1, \ldots, k \end{array}$$

Probabilistic Interpretation

So far, we have considered sparse coding in the context of finding a sparse, over-complete set of basis vectors to span our input space. Alternatively, we may also approach sparse coding from a probabilistic perspective as a generative model.

Consider the problem of modelling natural images as the linear superposition of k independent source features ϕ_i with some additive noise ν:

$$\mathbf{x} = \sum_{i=1}^{k} a_i \phi_i + \nu(\mathbf{x})$$

Our goal is to find a set of basis feature vectors ϕ such that the distribution of images $P(\mathbf{x} \mid \phi)$ is as close as possible to the empirical distribution of our input data $P^*(\mathbf{x})$. One method of doing so is to minimize the KL divergence between $P^*(\mathbf{x})$ and $P(\mathbf{x} \mid \phi)$ where the KL divergence is defined as:

$$D(P^*(\mathbf{x})||P(\mathbf{x} \mid \phi)) = \int P^*(\mathbf{x}) \log \left(\frac{P^*(\mathbf{x})}{P(\mathbf{x} \mid \phi)} \right) d\mathbf{x}$$

Since the empirical distribution $P^*(\mathbf{x})$ is constant across our choice of ϕ, this is equivalent to maximizing the log-likelihood of $P(\mathbf{x} \mid \phi)$.

Assuming ν is Gaussian white noise with variance σ^2, we have that

$$P(\mathbf{x} \mid \mathbf{a}, \phi) = \frac{1}{Z} \exp \left(-\frac{(\mathbf{x} - \sum_{i=1}^{k} a_i \phi_i)^2}{2\sigma^2} \right)$$

In order to determine the distribution $P(\mathbf{x} \mid \phi)$, we also need to specify the prior distribution $P(\mathbf{a})$. Assuming the independence of our source features, we can factorize our prior probability as

$$P(\mathbf{a}) = \prod_{i=1}^{k} P(a_i)$$

At this point, we would like to incorporate our sparsity assumption – the assumption that any single image is likely to be the product of relatively few source features. Therefore, we would like the probability distribution of a_i to be peaked at zero and have high kurtosis. A convenient parameterization of the prior distribution is

$$P(a_i) = \frac{1}{Z}\exp(-\beta S(a_i))$$

Where $S(a_i)$ is a function determining the shape of the prior distribution.

Having defined $P(\mathbf{x} \mid \mathbf{a}, \phi)$ and $P(\mathbf{a})$, we can write the probability of the data $\mathbf{x}$ under the model defined by ϕ as

$$P(\mathbf{x} \mid \phi) = \int P(\mathbf{x} \mid \mathbf{a}, \phi) P(\mathbf{a}) d\mathbf{a}$$

and our problem reduces to finding

$$\phi^* = \text{argmax}_\phi E\left[\log(P(\mathbf{x} \mid \phi))\right]$$

Where $E\left[\cdot\right]$ denotes expectation over our input data.

Unfortunately, the integral over $\mathbf{a}$ to obtain $P(\mathbf{x} \mid \phi)$ is generally intractable. We note though that if the distribution of $P(\mathbf{x} \mid \phi)$ is sufficiently peaked (w.r.t. $\mathbf{a}$), we can approximate its integral with the maximum value of $P(\mathbf{x} \mid \phi)$ and obtain a approximate solution

$$\phi^{*'} = \text{argmax}_\phi E\left[\max_{\mathbf{a}} \log(P(\mathbf{x} \mid \phi))\right]$$

As before, we may increase the estimated probability by scaling down a_i and scaling up ϕ (since $P(a_i)$ peaks about zero) , we therefore impose a norm constraint on our features ϕ to prevent this.

Finally, we can recover our original cost function by defining the energy function of this linear generative model

$$\begin{aligned} E\left(\mathbf{x}, \mathbf{a} \mid \phi\right) &:= -\log\left(P(\mathbf{x} \mid \phi, \mathbf{a}) P(\mathbf{a})\right) \\ &= \sum_{j=1}^{m} \left|\left|\mathbf{x}^{(j)} - \sum_{i=1}^{k} a_i^{(j)} \phi_i\right|\right|^2 + \lambda \sum_{i=1}^{k} S(a_i^{(j)}) \end{aligned}$$

where $\lambda = 2\sigma^2\beta$ and irrelevant constants have been hidden. Since maximizing the log-likelihood is equivalent to minimizing the energy function, we recover the original optimization problem:

$$\phi^*, \mathbf{a}^* = \text{argmin}_{\phi, \mathbf{a}} \sum_{j=1}^{m} \left|\left|\mathbf{x}^{(j)} - \sum_{i=1}^{k} a_i^{(j)} \phi_i\right|\right|^2 + \lambda \sum_{i=1}^{k} S(a_i^{(j)})$$

Using a probabilistic approach, it can also be seen that the choices of the L_1 penalty $|a_i|_1$ and the log penalty $\log(1 + a_i^2)$ for $S(.)$ correspond to the use of the Laplacian $P(a_i) \propto \exp\left(-\beta|a_i|\right)$ and the Cauchy prior $P(a_i) \propto \frac{\beta}{1+a_i^2}$ respectively.

Learning

Learning a set of basis vectors ϕ using sparse coding consists of performing two separate optimizations, the first being an optimization over coefficients a_i for each training example $\mathbf{x}$ and the second an optimization over basis vectors ϕ across many training examples at once.

Assuming an L_1 sparsity penalty, learning $a_i^{(j)}$ reduces to solving a L_1 regularized least squares problem which is convex in $a_i^{(j)}$ for which several techniques have been developed (convex optimization software such as CVX can also be used to perform L1 regularized least squares). Assuming a differentiable $S(.)$ such as the log penalty, gradient-based methods such as conjugate gradient methods can also be used.

Learning a set of basis vectors with a L_2 norm constraint also reduces to a least squares problem with quadratic constraints which is convex in ϕ. Standard convex optimization software (e.g. CVX) or other iterative methods can be used to solve for ϕ although significantly more efficient methods such as solving the Lagrange dual have also been developed.

As described above, a significant limitation of sparse coding is that even after a set of basis vectors have been learnt, in order to “encode” a new data example, optimization must be performed to obtain the required coefficients. This significant “runtime” cost means that sparse coding is computationally expensive to implement even at test time especially compared to typical feedforward architectures.

Self-Taught Learning

Self-Taught Learning (http://ufldl.stanford.edu/tutorial/selftaughtlearning/SelfTaughtLearning)

Exercise: Self-Taught Learning (http://ufldl.stanford.edu/tutorial/selftaughtlearning/ExerciseSelfTaughtLearning)