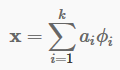
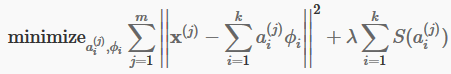
Sparse coding

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 *ϕi* such that we can represent an input vector **x** as a linear combination of these basis vectors:



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 (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 *ai* are no longer uniquely determined by the input vector **x**. Therefore, in sparse coding, we introduce the additional criterion of **sparsity** to resolve the degeneracy introduced by over-completeness. 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.

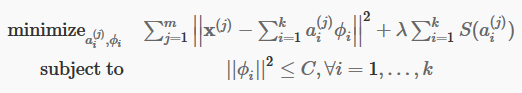
We define the sparse coding cost function on a set of *m* input vectors as



where  is a sparsity cost function which penalizes  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 **x** and the second term as a sparsity penalty which forces our representation of **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 , it is non-differentiable and difficult to optimize in general. In practice, common choices for the sparsity cost are the *L*1 penalty  and the log penalty .

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



# Learning

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

Assuming an *L*1 sparsity penalty, learning  reduces to solving a *L*1 regularized least squares problem which is convex in  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  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.