6
Sparse Coding

Many types of signals turn out to be sparse, either in their natural basis or in a hand-designed basis (e.g., a family of wavelets). But if we are given a collection of signals and we don't know the basis in which they are sparse, can we automatically learn it? This problem goes by various names, including sparse coding and dictionary learning. It was introduced in the context of neuroscience, where it was used to explain how neurons get the types of activation patterns they have. It also has applications to compression and deep learning. In this chapter, we will give algorithms for sparse coding that leverage convex programming relaxations as well as iterative algorithms where we will prove that greedy methods successfully minimize a nonconvex function in an appropriate stochastic model.

6.1 Introduction

Sparse coding was introduced by Olshausen and Field [117], who were neuroscientists interested in understanding properties of the mammalian visual cortex. They were able to measure the receptive field of neurons — essentially how neurons respond to various types of stimuli. But what they found surprised them. The response patterns were always

(a) *spatially localized*, which means that each neuron was sensitive only to light in a particular region of the image;
(b) *bandpass*, in the sense that adding high-frequency components had a negligible effect on the response; and
(c) *oriented*, in that rotating images with sharp edges produced responses only when the edge was within some range of angles.

https://doi.org/10.1017/9781316882177.007 Published online by Cambridge University Press

What's surprising is that if you took a collection of natural images and compressed them by finding a k-dimensional subspace to project them onto using principal component analysis, the directions you find wouldn't have any of these properties. So how are neurons learning the basis they are using to represent images?

What Olshausen and Field [117] proposed was revolutionary. First, what is better about the basis that neurons are using is that they produce sparse activation patterns. Or, in our language, neurons are representing the set of natural images in a basis in which they are sparse. Second, Olshausen and Field proposed that there are natural and biologically plausible rules for learning such a basis. They introduced a simple update rule, whereby neurons that fire together strengthen their connections to each other. This is called a Hebbian learning rule. And empirically, they showed that their iterative algorithm, when run on natural images, recovered a basis that met the above three properties. *Thus algorithms can explain the emergence of certain biological properties of the visual cortex.*

Since then, sparse coding and dictionary learning have become important problems in signal processing and machine learning. We will assume that we are given a collection of examples $b^{(1)}, b^{(2)}, \ldots, b^{(p)}$ that are sparse in a common basis. In particular, there is a matrix A and a set of representations $x^{(1)}, x^{(2)}, \ldots, x^{(p)}$ where $Ax^{(i)} = b^{(i)}$ and each $x^{(i)}$ is sparse. Let's discuss two popular approaches, called the method of optimal directions and k-SVD.

Method of Optimal Directions [68]

```
Input: Matrix $B$, whose columns can be jointly sparsely represented
Output: A basis $\widehat{A}$ and representation $\widehat{X}$

Guess $\widehat{A}$

Repeat until convergence:

    Given $\widehat{A}$, compute a column sparse $\widehat{X}$ so that $\widehat{A}\,\widehat{X} \approx B$ (using, e.g.,
    matching pursuit [111] or basis pursuit [50]).

    Given $\widehat{X}$, compute the $\widehat{A}$ that minimizes $\|\widehat{A}\,\widehat{X} - B\|_F$.
End
```

To simplify our notation, we have organized the observations $b^{(i)}$ as columns in a matrix B and will use the matrix $\widehat{X}$ to represent our estimated sparse representations. Another popular approach is the following:

K-SVD [5]

```
Input: Matrix $B$, whose columns can be jointly sparsely represented
Output: A basis $\widehat{A}$ and representation $\widehat{X}$
Guess $\widehat{A}$

Repeat until convergence:

    Given $\widehat{A}$, compute a column sparse $\widehat{X}$ so that $\widehat{A}\,\widehat{X} \approx B$ (using, e.g.,
    matching pursuit [111] or basis pursuit [50]).

    For each column $\widehat{A}_j$:

        Group all samples $b^{(i)}$ where $\widehat{x}^{(i)}$ has a nonzero at index $j$.
        Subtract off components in the other directions:

            $b^{(i)} - \sum_{j' \neq j} \widehat{A}_{j'} \widehat{x}^{(i)}_{j'}$

        Organize these vectors into a residual matrix and compute
        the top singular vector $v$ and update the column $\widehat{A}_j$ to $v$.

End
```

You should think about these algorithms as variants of the alternating minimization algorithm we gave for nonnegative matrix factorization. They follow the same style of heuristic. The difference is that k-SVD is more clever about how it corrects for the contribution of the other columns in our basis $\widehat{A}$ when performing an update, which makes it the heuristic of choice in practice. Empirically, both of these algorithms are sensitive to their initialization but work well aside from this issue.

We want algorithms with provable guarantees. Then it is natural to focus on the case where A is a basis for which we know how to solve sparse recovery problems. Thus we could consider both the undercomplete case, where A has full column rank, and the overcomplete case, where there are more columns than rows and A is either incoherent or has the restricted isometry property. That's exactly what we'll do in this chapter. We'll also assume a stochastic model for how the $x^{(i)}$'s are generated, which helps prevent lots of pathologies that can arise (e.g., a column in A is never represented).

6.2 The Undercomplete Case

In this section, we will give an algorithm for sparse coding when A has full column rank. Our approach will be based on a convex programming relaxation and many of the insights that we developed in the previous chapter. We will find our matrix X of sparse representations using the insight that its rows are the sparsest vectors in the row space of our matrix B of samples. More formally, the algorithm of Spielman et al. [131] works under the following natural generative model:

(a) There is an unknown dictionary A that is an $n \times m$ matrix and has full column rank.
(b) Each sample x has independent coordinates, which are nonzero with probability θ. If a coordinate is nonzero, its value is sample from a standard Gaussian.
(c) We observe b where $Ax = b$.

Thus all of our samples are sparse in an unknown basis. So we would like to find A, or equivalently find its left pseudoinverse A^+, which is a linear transformation that makes all of our samples sparse. The parameter θ governs the average sparsity of each representation x and it is required to be neither too big nor too small. More formally, we assume

$$\frac{1}{n} \leq \theta \frac{1}{n^{1/2} \log n}.$$

Spielman et al. [131] gave a polynomial time algorithm to recover A exactly. This is a stronger guarantee than the algorithms we will see later, which merely recover A approximately or to arbitrarily good precision, but require more and more samples to do so. However, the later algorithms will work in the overcomplete case and in the presence of noise. It's also important to note that, strictly speaking, if the coordinates of x_i were independent, we could recover A using algorithms for independent component analysis [74]. However, those algorithms are very sensitive to the independence assumption, and everything we do here will work even under weaker conditions (that are a mouthful to properly spell out).

We will make the simplifying assumption that A is invertible. This doesn't really cost us anything, but let's leave that as an exercise for the reader. In any case, the main insight that underlies the algorithm is contained in the following claims:

Claim 6.2.1 *The row span of B and the row span of $A^{-1}B = X$ are the same.*

Proof: The proof follows by observing that for any vector u,

$$u^T B = (u^T A)A^{-1}B = v^T X.$$

So we can represent any linear combination of the rows of B with a corresponding linear combination of the rows of X. We can obviously go in the reverse direction as well. ■

We will state the second claim informally for now:

Claim 6.2.2 *Given enough samples, with high probability the sparsest vectors in the row span of X are the rows of X.*

Hopefully this claim is intuitively obvious. The rows of X are independent random vectors whose average sparsity is θ. For our choice of θ, we will have few collisions, which means the sparsity of any two rows of X should be about twice the sparsity of one row.

Now we come to the need for a convex programming relaxation. We can't hope to directly find the sparsest vector in an arbitrary subspace. We proved that that problem is *NP*-hard in Theorem 5.1.5. But let's leverage our insights from sparse recovery and use a convex programming relaxation instead. Consider the following optimization problem:

$$(P_1) \qquad \min \|w^T B\|_1 \text{ s.t. } r^T w = 1$$

This is the usual trick of replacing the sparsity of a vector with its ℓ_1 norm. The constraint $r^T w = 1$ is needed just to fix a normalization, to prevent us from returning the all-zero vector as a solution. We will choose r to be a column in B for reasons that will become clear later. Our goal is to show that the optimal solution to (P_1) is a scaled row of X. In fact, we can transform the above linear program into a simpler one that will be easier to analyze:

$$(Q_1) \qquad \min \|z^T X\|_1 \text{ s.t. } c^T z = 1$$

Lemma 6.2.3 *Let $c = A^{-1}r$. Then there is a bijection between the solutions of (P_1), and the solutions of (Q_1) that preserves the objective value.*

Proof: Given a solution w to (P_1), we can set $z = A^T w$. The objective values are the same because

$$w^T B = w^T AX = z^T X$$

and the linear constraint is met, because again

$$1 = r^T w = r^T (A^T)^{-1} z = r^T (A^{-1})^T z = c^T z$$

and it is easy to check that you can go from a solution to (Q_1) to a solution to (P_1) in the analogous way. ■

The Minimizers Are Somewhat Sparse

Here we will establish a key step in the analysis. We will show that any optimal solution z_* has its support contained in the support of c. Remember, we chose r to be a column of B. We promised to give an explanation later, so now we can ask: Why did we do this? The point is that if r is a column of B, then the way our bijection between solutions to (P_1) and (Q_1) worked was that we set $c = A^{-1}r$, and so c is a column of X. In our model, c is sparse, so if we show that the support of z_* is contained in the support of c, we'll have proven that z_* is sparse too.

Now let's state and prove the main lemma in this subsection. In what follows, we will assert that certain things happen with high probability but will not dwell on the number of samples needed to make these things be true. Instead, we will give a heuristic argument why the concentration bounds ought to work out that way and focus on the analogy to sparse recovery. For full details, see Spielman et al. [131].

Lemma 6.2.4 *With high probability, any optimal solution z_* to (Q_1) satisfies $supp(z_*) \subseteq supp(c)$.*

Proof: Let's decompose z_* into two parts. Set $J = \text{supp}(c)$ and write $z_* = z_0 + z_1$ where z_0 is supported in J and z_1 is supported in $\overline{J}$. Then we have $c^T z_0 = c^T z_*$. What this means is that since z_* is a feasible solution to (Q_1), then z_0 is too. Our goal is to show that z_0 is a strictly better solution to (Q_1) than z_* is. More formally, we want to show:

$$\|z_0^T X\|_1 < \|z_*^T X\|_1.$$

Let S be the set of columns of X that have a nonzero entry in J. That is,

$$S = \{j | X_j^J \neq \vec{0}\}.$$

We now compute:

$$\begin{aligned}\|z_*^T X\|_1 &= \|z_*^T X_S\|_1 + \|z_*^T X_{\overline{S}}\|_1 \\ &\geq \|z_0^T X_S\|_1 - \|z_1^T X_S\|_1 + \|z_1^T X_{\overline{S}}\|_1 \\ &\geq \|z_0^T X\|_1 - 2\|z_1^T X_S\|_1 + \|z_1^T X\|_1\end{aligned}$$

For now, let's assume the following claim:

Claim 6.2.5 *With high probability, for any nonzero z_1 we have $\|z_1^T X\|_1 > 2\|z_1^T X_S\|_1$.*

With this claim, we have

$$\|z_*^T X\|_1 > \|z_0^T X\|_1$$

which completes the proof. ■

Now let's prove Claim 6.2.5:

Proof: For now, let's cheat and assume that z_1 is fixed and is a unit vector. Then S is a random set, and if we take p samples from the model we have

$$\mathbb{E}[\|z_1^T X_S\|_1] = \frac{|S|}{p}\,\mathbb{E}[\|z_1^T X\|_1].$$

The expected size of S is $p \times \mathbb{E}[|\text{supp}(x_i)|] \times \theta = \theta^2 np = o(p)$. Together, these imply that

$$\mathbb{E}[\|z_1^T X\|_1 - 2\|z_1^T X_S\|_1] = \left(1 - \frac{2\,\mathbb{E}[|S|]}{p}\right)\mathbb{E}[\|z_1^T X\|_1]$$

is bounded away from zero, thus proving our desired bound

$$\|z_1^T X\|_1 - 2\|z_1^T X_S\|_1 > 0$$

holds with high probability for any fixed z_1. We can take a union bound over an ϵ-net of all possible unit vectors z_1 and conclude by rescaling that the bound holds for all nonzero z_1's. ■

The Minimizers Are Rows of X

Now we know that the solutions to (Q_1) are somewhat sparse, because their support is contained in the support of c. But even sparse linear combinations of the rows of X will have few collisions, and so we should expect the ℓ_1 norm to be approximately preserved. More precisely:

Lemma 6.2.6 *With high probability, for any vector z supported in a set J of size at most $10\theta n \log n$, we have*

$$\|z_J^T X^J\|_1 = (1 \pm o(1))C\frac{p}{|J|}\|z_J\|_1$$

where C is the expected absolute value of a nonzero in X.

We will not prove this lemma here. See Spielman et al. [131] for full details. However, the intuition is easy. We should expect most of the columns of X_J to have at most one nonzero element. It is straightforward to analyze the expected contribution of these columns, and the remaining columns have only lower-order contributions. What this means for us is that instead of (Q_1), we can consider

$$(R_1) \qquad \min \|z\|_1 \text{ s.t. } c^T z = 1$$

because the feasible regions of (Q_1) and (R_1) are the same, and their objective value is nearly the same after rescaling. The final step is the following:

Lemma 6.2.7 *If c has a unique coordinate of maximum value c_i, then the unique optimal solution to (R_1) satisfies $z_i = 1/c_i$ and $z_j = 0$ for all other coordinates j.*

Now we can state the main theorem:

Theorem 6.2.8 *[131] Suppose A is an $n \times m$ matrix with full column rank and we are given a polynomial number of samples from the generative model. There is a polynomial time algorithm to recover A exactly (up to a permutation and rescaling of its columns) that succeeds with high probability.*

Proof: The theorem follows by putting together Lemma 6.2.4, Lemma 6.2.6, and Lemma 6.2.7. Using these and the bijection in Lemma 6.2.3, we conclude that for any optimal solution to (P_1), the vector that appears in the objective function is

$$w^T B = z^T X$$

where the only the i^{th} coordinate of z is nonzero. Hence it is a scaled copy of the i^{th} row of X. Now, since the generative model chooses the nonzero entries of x from a standard Gaussian, almost surely there is a coordinate that is the strictly largest in absolute value.

In fact, even more is true. For any fixed coordinate i, with high probability it will be the strictly largest coordinate in absolute value for some column of X. This means that if we repeatedly solve (P_1) by setting r to be different columns of B, then with high probability every row of X will show up. Now, once we know the rows of X, we can solve for A as follows. With high probability, if we take enough samples, then X will have a left pseudo-inverse and we can compute $A = BX^+$, which will recover A up to a permutation and rescaling of its columns. This completes the proof. ■

6.3 Gradient Descent

Gradient descent and its relatives are some of the most ubiquitous algorithms in machine learning. Traditionally, we are faced with the task of minimizing a convex function $f : \mathbb{R}^n \to \mathbb{R}$ either over all of space (the unconstrained case) or over some convex body K. The simplest possible algorithm you could think of — follow the direction of steepest descent — works. Actually, there are all sorts of convergence guarantees out there, depending on what you know about your function. Is it at least twice differentiable? Do its gradients smoothly vary? Can you fit a quadratic function under it? There are even accelerated methods that get faster rates by leveraging connections to physics, like momentum. You could write an entire book on iterative methods. And indeed there are many terrific sources, such as Nesterov [116] and Rockefellar [127].

In this section we will prove some basic results about gradient descent in the simplest setting, where f is twice differentiable, β-smooth, and α-strongly convex. We will show that the difference between the current value of our objective and the optimal value decays exponentially. Ultimately, our interest in gradient descent will be in applying it to nonconvex problems. Some of the most interesting problems, like fitting parameters in a deep network, are nonconvex. When faced with a nonconvex function f, you just run gradient descent anyway.

It is very challenging to prove guarantees about nonconvex optimization (except for things like being able to reach a local minimum). Nevertheless, our approach for overcomplete sparse coding will be based on an abstraction of the analysis of gradient descent. What is really going on under the hood is that the gradient always points you somewhat in the direction of the globally minimal solution. In nonconvex settings, we will still be able to get some mileage out of this intuition by showing that under the appropriate stochastic assumptions, even simple update rules make progress in a similar manner. In any case, let's now define gradient descent:

Gradient Descent

Given: A convex, differentiable function $f : \mathbb{R}^n \to \mathbb{R}$
Output: A point x_T that is an approximate minimizer of f

```
For $t = 1$ to $T$
    $x_{t+1} = x_t - \eta \nabla f(x_t)$
End
```

The parameter η is called the learning rate. You want to make it large, but not so large that you overshoot. Our analysis of gradient descent will hinge on multivariable calculus. A useful ingredient for us will be the following multivariate Taylor's theorem:

Theorem 6.3.1 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex, differentiable function. Then*

$$f(y) = f(x) + (\nabla f(x))^T (y - x) + \frac{1}{2}(y - x)^T \nabla^2 f(x)(y - x) + o(\|y - x\|^2).$$

Now let's precisely define the conditions on f that we will impose. First we need the gradient to not change too quickly:

Definition 6.3.2 *We will say that f is β-smooth if for all x and y, we have*

$$\|\nabla f(y) - \nabla f(x)\| \leq \beta \|y - x\|.$$

Alternatively, if f is twice differentiable, the condition above is equivalent to $\|\nabla^2 f(x)\| \leq \beta$ for all x.

Next we need to be able to fit a quadratic function underneath f. We need a condition like this to preclude the case where f is essentially flat for a long time but we need to move far to reach the global minimum. If you can fit a quadratic function underneath f, then you know that the global minimum cannot be too far away from where you currently are.

Definition 6.3.3 *We will say that a convex function f is α-strongly convex if for all x and y, we have*

$$(y - x)^T \nabla^2 f(x)(y - x) \geq \alpha \|y - x\|^2$$

or, equivalently, for all x and y, f satisfies

$$f(y) \geq f(x) + (\nabla f(x))^T (y - x) + \frac{\alpha}{2} \|y - x\|^2.$$

Now let's state the main result we will prove in this section:

Theorem 6.3.4 *Let f be twice differentiable, β-smooth, and α-strongly convex. Let x^* be the minimizer of f and $\eta \leq \frac{1}{\beta}$. Then gradient descent starting from x_1 satisfies*

$$f(x_t) - f(x^*) \leq \beta \Big(1 - \frac{\eta\alpha}{2}\Big)^{t-1} \|x_1 - x^*\|^2.$$

We will make use of the following helper lemma:

Lemma 6.3.5 *If f is twice differentiable, β-smooth, and α-strongly convex, then*

$$\nabla f(x_t)^T (x_t - x^*) \geq \frac{\alpha}{4} \|x_t - x^*\|^2 + \frac{1}{2\beta} \|\nabla f(x_t)\|^2.$$

Let's come back to its proof. For now, let's see how it can be used to establish Theorem 6.3.4:

Proof: Let $\alpha' = \frac{\alpha}{4}$ and $\beta' = \frac{1}{2\beta}$. Then we have

$$\begin{aligned}
\|x_{t+1} - x^*\|^2 &= \|x_t - x^* - \eta \nabla f(x_t)\|^2 \\
&= \|x_t - x^*\|^2 - 2\eta \nabla f(x_t)^T (x_t - x^*) + \eta^2 \|\nabla f(x_t)\|^2 \\
&\le \|x_t - x^*\|^2 - 2\eta(\alpha' \|x_t - x^*\|^2 + \beta' \|\nabla f(x_t)\|) \\
&= (1 - 2\eta\alpha')\|x_t - x^*\|^2 + (\eta^2 - 2\eta\beta')\|\nabla f(x_t)\|^2 \\
&\le (1 - 2\eta\alpha')\|x_t - x^*\|^2.
\end{aligned}$$

The first equality follows from the definition of gradient descent. The first inequality follows from Lemma 6.3.5 and the last inequality from the bound on the learning rate η. To complete the proof, note that

$$f(x_t) + \nabla f(x_t)^T (x^* - x_t) \le f(x^*).$$

Rearranging this inequality and invoking β-smoothness, we have

$$f(x_t) - f(x^*) \le \nabla f(x_t)^T (x_t - x^*) \le \beta \|x_t - x^*\|^2.$$

And putting it all together, we have

$$f(x_t) - f(x^*) \le \beta \Big(1 - 2\eta\alpha'\Big) \|x_t - x^*\|^2$$

which completes the proof. ■

Now let's tie up our loose ends and prove Lemma 6.3.5:

Proof: First, by strong convexity we have

$$f(x^*) \ge f(x) + \nabla f(x)^T (x^* - x) + \frac{\alpha}{2} \|x - x^*\|^2.$$

Using the fact that $f(x) \ge f(x^*)$ and rearranging, we get

$$\nabla f(x)^T (x - x^*) \ge \frac{\alpha}{2} \|x - x^*\|^2.$$

This is half of the lemma. Now let's relate the left-hand side to the norm of the gradient. Actually, we need a more convenient form of Theorem 6.3.1 that has Lagrange remainder:

Theorem 6.3.6 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice differentiable function. Then, for some $t \in [0, 1]$ and $x' = ty + (1 - t)x$, we have*

$$\nabla f(x) = \nabla f(y) + \nabla^2 f(x')(x - y).$$

This can be proven using a multivariate intermediate value theorem. In any case, by setting $y = x^*$ and observing that $\nabla f(x^*) = 0$, we get

$$\nabla f(x) = \nabla^2 f(x')(x - x^*)$$

from which we get

$$\nabla f(x)^T (\nabla^2 f(x'))^{-1} \nabla f(x) = \nabla f(x)^T (x - x^*)$$

for some $x' = tx + (1-t)x^*$. Now, β-smoothness implies that $(\nabla^2 f(x'))^{-1} \geq \frac{1}{\beta}\|\nabla f(x')\|^2$, which gives us

$$\|\nabla f(x)^T (x - x^*)\| \geq \frac{1}{\beta}.$$

Taking the average of the two main inequalities completes the proof. ■

Actually, our proof works even when the direction you move in is just an approximation to the gradient. This is an important shortcut when, for example, f is a loss function that depends on a very large number of training examples. Instead of computing the gradient of f, you can sample some training examples, compute your loss function on just those, and follow its gradient. This is called *stochastic gradient descent*. The direction it moves in is a random variable whose expectation is the gradient of f. The beauty of it is that the usual proofs of convergence for gradient descent carry over straightforwardly (provided your sample is large enough).

There is an even further abstraction we can make. What if the direction you move in isn't a stochastic approximation of the gradient, but is just some direction that satisfies the conditions shown in Lemma 6.3.5? Let's call this abstract gradient descent, just to give it a name:

Abstract Gradient Descent

```
Given: A function $f : \mathbb{R}^n \to \mathbb{R}$
Output: A point $x_T$ that is close to $x^*$

For $t = 1$ to $T$
        $x_{t+1} = x_t - \eta g_t$
End
```

Let's introduce the following key definition:

Definition 6.3.7 *We say that g_t is $(\alpha', \beta', \epsilon_t)$-correlated with a point x^* if for all t we have*

$$g_t^T (x_t - x^*) \geq \alpha' \|x_t - x^*\|^2 + \beta' \|g_t\|^2 - \epsilon_t.$$

We have already proven that if f is twice differentiable, β-smooth, and α-strongly convex, then the gradient is $(\frac{\alpha}{4}, \frac{1}{2\beta}, 0)$-correlated with the optimal solution x^*. It turns out that the proof we gave of Theorem 6.3.4 generalizes immediately to this more abstract setting:

Theorem 6.3.8 *Suppose that g_t is $(\alpha', \beta', \epsilon_t)$-correlated with a point x^* and, moreover, $\eta \leq 2\beta'$. Then abstract gradient descent starting from x_1 satisfies*

$$\|x_t - x^*\|^2 \leq \left(1 - \frac{\eta\alpha'}{2}\right)^{t-1} \|x_1 - x^*\|^2 + \frac{\max_t \epsilon_t}{\alpha'}.$$

Now we have the tools we need for overcomplete sparse coding. We'll prove convergence bounds for iterative methods in spite of the fact that the underlying function they are attempting to minimize is nonconvex. The key is to use the above framework and exploit the stochastic properties of our model.

6.4 The Overcomplete Case

In this section, we will give an algorithm for sparse coding that works for overcomplete dictionaries. As usual, we will work in a stochastic model. More formally, x is a random k-sparse vector generated according to the following procedure:

(a) The support of x is chosen uniformly at random from all size k subsets of $[m]$.
(b) If the j^{th} coordinate is nonzero, then its value is independently chosen to be $+1$ or -1 (with equal probability).

And we observe just the right-hand side of $Ax = b$. Our goal is to learn the columns of A given enough samples from the model. Actually, we've made some simplifying assumptions in the above model that we won't actually need. We don't really need the support of x to be chosen uniformly at random, or the coordinates to be independent. In fact, our algorithms will even be able to tolerate additive noise. Nevertheless, our model is easier to think about, so let's stick with it.

Now we come to the main conceptual insight. Usually we think of iterative algorithms as performing alternating minimization on a nonconvex objective function. For example, a popular energy function in the context of sparse coding is the following:

$$\mathcal{E}(\widehat{A}, \widehat{X}) = \sum_{i=1}^{p} \|b^{(i)} - \widehat{A}\widehat{x}^{(i)}\|^2 + \sum_{i=1}^{p} L(\widehat{x}^{(i)})$$

where $Ax^{(i)} = b^{(i)}$ are our observed samples. Moreover, L is a loss function that penalizes for vectors $\widehat{x}^{(i)}$ that are not k-sparse. You can think of this as being a hard penalty function that is infinite when x has more than k nonzero coordinates and is zero otherwise. It could also be your favorite sparsity-inducing soft penalty function.

Many iterative algorithms attempt to minimize an energy function like the one above that balances how well your basis explains each sample and how sparse each representation is. The trouble is that the function is nonconvex, so if you want to give provable guarantees, you would have to figure out all kinds of things, like why it doesn't get stuck in a local minimum or why it doesn't spend too much time moving slowly around saddle points.

Question 8 *Instead of viewing iterative methods as attempting to minimize a known nonconvex function, can we view them as minimizing an unknown convex function?*

What we mean is: What if, instead of the $\widehat{x}$'s, we plug in the true sparse representations x? Our energy function becomes

$$\mathcal{E}(\widehat{A}, X) = \sum_{i=1}^{p} \|b^{(i)} - \widehat{A}x^{(i)}\|^2$$

which is convex, because only the basis A is unknown. Moreover, it's natural to expect that in our stochastic model (and probably many others), the minimizer of $\mathcal{E}(\widehat{A}, X)$ converges to the true basis A. So now we have a convex function where there is a path from our initial solution to the optimal solution via gradient descent. The trouble is that we cannot evaluate or compute gradients of the function $\mathcal{E}(\widehat{A}, X)$, because X is unknown.

The path we will follow in this section is to show that simple, iterative algorithms for sparse coding move in a direction that is an approximation to the gradient of $\mathcal{E}(\widehat{A}, X)$. More precisely, we will show that under our stochastic model, the direction our update rule moves in meets the conditions in Definition 6.3.7. That's our plan of action. We will study the following iterative algorithm:

Hebbian Rule for Sparse Coding

```
Input: Samples $b = Ax$ and an estimate $\widehat{A}$
Output: An improved estimate $\widehat{A}$

For $t = 0$ to $T$

    Decode using the current dictionary:

        $\widehat{x}^{(i)} = \text{threshold}_{1/2}(\widehat{A}^T b^{(i)})$
        Update the dictionary:
            $\widehat{A} \leftarrow \widehat{A} + \eta \sum_{i=qt+1}^{q(t+1)} (b^{(i)} - \widehat{A}\widehat{x}^{(i)}) \operatorname{sign}(\widehat{x}^{(i)})^T$
End
```

We have used the following notation:

Definition 6.4.1 *Let* sign *denote the entrywise operation that sets positive coordinates to* $+1$*, negative coordinates to* -1*, and zero to zero. Also, let* threshold_C *denote the entrywise operation that zeros out coordinates whose absolute value is less than* $C/2$ *and keeps the rest of the coordinates the same.*

The update rule is also natural in another sense. In the context of neuroscience, the dictionary A often represents the connection weights between two adjacent layers of neurons. Then the update rule has the property that it strengthens the connections between pairs of neurons that fire together when you set up a neural network that computes the sparse representation. Recall, these are called *Hebbian* rules.

Now let's define the metric we will use to measure how close our estimate $\widehat{A}$ is to the true dictionary A. As usual, we cannot hope to recover which column is which or the correct sign, so we need to take this into account:

Definition 6.4.2 *We will say that two* $n \times m$ *matrices* $\widehat{A}$ *and* A*, whose columns are unit vectors, are* (δ, κ)*-close if there is a permutation and sign flip of the columns of* $\widehat{A}$ *that results in a matrix* B *that satisifes*

$$\|B_i - A_i\| \leq \delta$$

for all i, and furthermore $\|B - A\| \leq \kappa \|A\|$.

First let's analyze the decoding step of the algorithm:

Lemma 6.4.3 *Suppose that* A *is an* $n \times m$ *matrix that is* μ*-incoherent and that* $Ax = b$ *is generated from the stochastic model. Further suppose that*

$$k \leq \frac{1}{10\mu \log n}$$

and $\widehat{A}$ *is* $(1/\log n, 2)$*-close to* A*. Then decoding succeeds; i.e.,*

$$\operatorname{sign}(\text{threshold}_{1/2}(\widehat{A}^T b)) = \operatorname{sign}(x)$$

with high probability.

We will not prove this lemma here. The idea is that for any j, we can write

$$(\widehat{A}^T b)_j = A_j^T A_j x_j + (\widehat{A}_j - A_j)^T A_j x_j + \widehat{A}_j^T \sum_{i \in S \setminus \{j\}} A_i x_i$$

where $S = \text{supp}(x)$. The first term is x_j. The second term is at most $1/\log n$ in absolute value. And the third term is a random variable whose variance can be appropriately bounded. For the full details, see Arora et al. [16]. Keep in mind that for incoherent dictionaries, we think of $\mu = 1/\sqrt{n}$.

Let γ denote any vector whose norm is negligible (say $n^{-\omega(1)}$). We will use γ to collect various sorts of error terms that are small, without having to worry about what the final expression looks like. Consider the expected direction that our Hebbian update moves in when restricted to some column j. We have

$$g_j = \mathbb{E}[(b - \widehat{A}\widehat{x}\,)\ \text{sign}(\widehat{x}_j)]$$

where the expectation is over a sample $Ax = b$ from our model. This is a priori a complicated expression to analyze, because b is a random variable of our model and $\widehat{x}$ is a random variable that arises from our decoding rule. Our main lemma is the following:

Lemma 6.4.4 *Suppose that $\widehat{A}$ and A are $(1/\log n, 2)$-close. Then*

$$g_j = p_j q_j (I - \widehat{A}_j \widehat{A}_j^T) A_j + p_j \widehat{A}_{-j} Q \widehat{A}_{-j}^T A_j \pm \gamma$$

where $q_j = \mathbb{P}[j \in S]$, $q_{i,j} = \mathbb{P}[i, j \in S]$ and $p_j = \mathbb{E}[x_j \text{sign}(x_j) | j \in S]$. Moreover, $Q = \text{diag}(\{q_{i,j}\}_i)$.

Proof: Using the fact that the decoding step recovers the correct signs of x with high probability, we can play various tricks with the indicator variable for whether or not the decoding succeeds and be able to replace $\widehat{x}$'s with x's. For now, let's state the following claim, which we will prove later:

Claim 6.4.5 $g_j = \mathbb{E}[(I - \widehat{A}_S \widehat{A}_S^T) Ax\ sign(x_j))] \pm \gamma$

Now let $S = \text{supp}(x)$. We will imagine first sampling the support of x, then choosing the values of its nonzero entries. Thus we can rewrite the expectation using subconditioning as

$$\begin{aligned} g_j &= \mathop{\mathbb{E}}_S[\mathop{\mathbb{E}}_{x_S}[(I - \widehat{A}_S \widehat{A}_S^T) Ax\ \text{sign}(x_j))]|S] \pm \gamma \\ &= \mathop{\mathbb{E}}_S[\mathop{\mathbb{E}}_{x_S}[(I - \widehat{A}_S \widehat{A}_S^T) A_j x_j\ \text{sign}(x_j))]|S] \pm \gamma \\ &= p_j \mathop{\mathbb{E}}_S[(I - \widehat{A}_S \widehat{A}_S^T) A_j] \pm \gamma \\ &= p_j q_j (I - \widehat{A}_j \widehat{A}_j^T) A_j + p_j \widehat{A}_{-j} Q \widehat{A}_{-j}^T A_j \pm \gamma. \end{aligned}$$

The second equality uses the fact that the coordinates are uncorrelated, conditioned on the support S. The third equality uses the definition of p_j. The fourth equality follows from separating the contribution from j from all the other coordinates, where A_{-j} denotes the matrix we obtain by deleting the j^{th} column. This now completes the proof of the main lemma. ■

So why does this lemma tell us that our update rule meets the conditions in Definition 6.3.7? When $\widehat{A}$ and A are close, you should think of the expression as follows:

$$g_j = \underbrace{p_j q_j (I - \widehat{A}_j \widehat{A}_j^T) A_j}_{\approx p_j q_j (A_j - \widehat{A}_j)} + \underbrace{p_j \widehat{A}_{-j} Q \widehat{A}_{-j}^T A_j}_{\text{systemic error}} \pm \gamma$$

And so the expected direction that the update rule moves in is almost the ideal direction $A_j - \widehat{A}_j$, pointing toward the true solution. What this tells us is that sometimes the way to get around nonconvexity is to have a reasonable stochastic model. Even though in a worst-case sense you can still get stuck in a local minimum, in the average case you often make progress with each step you take. We have not discussed the issue of how to initialize it. But it turns out that there are simple spectral algorithms to find a good initialization. See Arora et al. [16] for the full details, as well as the guarantees of the overall algorithm.

Let's conclude by proving Claim 6.4.5:

Proof: Let F denote the event that decoding recovers the correct signs of x. From Lemma 6.4.3, we know that F holds with high probability. First let's use the indicator variable for event F to replace the $\widehat{x}$ inside the sign function with x at the expense of adding a negligible error term:

$$\begin{aligned} g_j &= \mathbb{E}[(b - \widehat{A}\widehat{x}) \operatorname{sign}(\widehat{x}_j) \mathbb{1}_F] + \mathbb{E}[(b - \widehat{A}\widehat{x}) \operatorname{sign}(\widehat{x}_j) \mathbb{1}_{\overline{F}}] \\ &= \mathbb{E}[(b - \widehat{A}\widehat{x}) \operatorname{sign}(x_j) \mathbb{1}_F] \pm \gamma \end{aligned}$$

The equality uses the fact that $\operatorname{sign}(\widehat{x}_j) = \operatorname{sign}(x_j)$ when event F occurs. Now let's substitute in for $\widehat{x}$:

$$\begin{aligned} g_j &= \mathbb{E}[(b - \widehat{A} \operatorname{threshold}_{1/2}(\widehat{A}^T b)) \operatorname{sign}(x_j) \mathbb{1}_F] \pm \gamma \\ &= \mathbb{E}[(b - \widehat{A}_S \widehat{A}_S^T b) \operatorname{sign}(x_j) \mathbb{1}_F] \pm \gamma \\ &= \mathbb{E}[(I - \widehat{A}_S \widehat{A}_S^T) b \operatorname{sign}(x_j) \mathbb{1}_F] \pm \gamma \end{aligned}$$

Here we have used the fact that $\operatorname{threshold}_{1/2}(\widehat{A}^T b)$ keeps all coordinates in S the same and zeros out the rest when event F occurs. Now we can play some more tricks with the indicator variable to get rid of it:

$$\begin{aligned} g_j &= \mathbb{E}[(I - \widehat{A}_S\widehat{A}_S^T)b \operatorname{sign}(x_j)] - \mathbb{E}[(I - \widehat{A}_S\widehat{A}_S^T)b \operatorname{sign}(x_j)\mathbb{1}_{\overline{F}}] \pm \gamma \\ &= \mathbb{E}[(I - \widehat{A}_S\widehat{A}_S^T)b \operatorname{sign}(x_j)] \pm \gamma \end{aligned}$$

which completes the proof of the claim. Line by line, the manipulations are trivial, but they yield a useful expression for the update rule. ■

There are other, earlier algorithms for overcomplete sparse coding. Arora et al. [15] gave an algorithm based on overlapping clustering that works for incoherent dictionaries almost up to the threshold where the sparse recovery problem has a unique solution, a la Lemma 5.2.3. Agarwal et al. [2, 3] gave algorithms for overcomplete, incoherent dictionaries that work up to thresholds that are worse by a polynomial factor. Barak et al. [25] gave algorithms based on the sum-of-squares hierarchy that work with nearly linear sparsity, but where the degree of the polynomial depends on the desired accuracy.

6.5 Exercises

Problem 6-1: Consider the sparse coding model $y = Ax$ where A is a fixed $n \times n$ matrix with orthonormal columns a_i, and x has i.i.d. coordinates drawn from the distribution

$$x_i = \begin{cases} +1 & \text{with probability } \alpha/2, \\ -1 & \text{with probability } \alpha/2, \\ 0 & \text{with probability } 1-\alpha. \end{cases}$$

The goal is to recover the columns of A (up to sign and permutation) given many independent samples y. Construct the matrix

$$M = \mathbb{E}_y\left[\langle y^{(1)}, y\rangle\langle y^{(2)}, y\rangle yy^T\right]$$

where $y^{(1)} = Ax^{(1)}$ and $y^{(2)} = Ax^{(2)}$ are two fixed samples from the sparse coding model, and the expectation is over a third sample y from the sparse coding model. Let $\hat{z}$ be the (unit-norm) eigenvector of M corresponding to the largest (in absolute value) eigenvalue.

(a) Write an expression for M in terms of $\alpha, x^{(1)}, x^{(2)}, \{a_i\}$.

(b) Assume for simplicity that $x^{(1)}$ and $x^{(2)}$ both have support size exactly αn and that their supports intersect at a single coordinate i^*. Show that $\langle\hat{z}, a_{i^*}\rangle^2 \geq 1 - O(\alpha^2 n)$ in the limit $\alpha \to 0$.

This method can be used to find a good starting point for alternating minimization.