10 Other Topics

10.1 Ranking and Social Choice

Combining feedback from multiple users to rank a collection of items is an important task. We rank movies, restaurants, web pages, and many other items. Ranking has become a multi-billion dollar industry as organizations try to raise the position of their web pages in the results returned by search engines to relevant queries. Developing a method of ranking that cannot be easily gamed by those involved is an important task.

A ranking of a collection of items is defined as a complete ordering. For every pair of items a and b, either a is preferred to b or b is preferred to a. Furthermore, a ranking is transitive in that $a > b$ and $b > c$ implies $a > c$.

One problem of interest in ranking is that of combining many individual rankings into one global ranking. However, merging ranked lists in a meaningful way is nontrivial as the following example illustrates.

Example: Suppose there are three individuals who rank items a, b, and c as illustrated in the following table.

individual	first item	second item	third item
1	a	b	c
2	b	c	a
3	c	a	b

Suppose our algorithm tried to rank the items by first comparing a to b and then comparing b to c. In comparing a to b, two of the three individuals prefer a to b and thus we conclude a is preferable to b. In comparing b to c, again two of the three individuals prefer b to c and we conclude that b is preferable to c. Now by transitivity one would expect that the individuals would prefer a to c, but such is not the case, only one of the individuals prefers a to c and thus c is preferable to a. We come to the illogical conclusion that a is preferable to b, b is preferable to c, and c is preferable to a. ■

Suppose there are a number of individuals or voters and a set of candidates to be ranked. Each voter produces a ranked list of the candidates. From the set of ranked lists can one construct a reasonable single ranking of the candidates? Assume the method of producing a global ranking is required to satisfy the following three axioms.

Nondictatorship – The algorithm cannot always simply select one individual's ranking to use as the global ranking.

Unanimity – If every individual prefers a to b, then the global ranking must prefer a to b.

Independent of irrelevant alternatives – If individuals modify their rankings but keep the order of a and b unchanged, then the global order of a and b should not change.

Arrow showed that it is not possible to satisfy all three of the above axioms. We begin with a technical lemma.

Lemma 10.1 *For a set of rankings in which each individual ranks an item b either first or last (some individuals may rank b first and others may rank b last), a global ranking satisfying the above axioms must put b first or last.*

Proof: Let a, b, and c be distinct items. Suppose to the contrary that b is not first or last in the global ranking. Then there exist a and c where the global ranking puts $a > b$ and $b > c$. By transitivity, the global ranking puts $a > c$. Note that all individuals can move c above a without affecting the order of b and a or the order of b and c since b was first or last on each list. Thus, by independence of irrelevant alternatives, the global ranking would continue to rank $a > b$ and $b > c$ even if all individuals moved c above a since that would not change the individuals relative order of a and b or the individuals relative order of b and c. But then by unanimity, the global ranking would need to put $c > a$, a contradiction. We conclude that the global ranking puts b first or last. ∎

Theorem 10.2 *(**Arrow**) Any deterministic algorithm for creating a global ranking from individual rankings of three or more elements in which the global ranking satisfies unanimity and independence of irrelevant alternatives is a dictatorship.*

Proof: Let a, b, and c be distinct items. Consider a set of rankings in which every individual ranks b last. By unanimity, the global ranking must also rank b last. Let the individuals, one by one, move b from bottom to top leaving the other rankings in place. By unanimity, the global ranking must eventually move b from the bottom all the way to the top. When b first moves, it must move all the way to the top by Lemma 10.1.

Let v be the first individual whose change causes the global ranking of b to change. We argue that v is a dictator. First, we argue that v is a dictator for any pair ac not involving b. We will refer to the three rankings of v in Figure 10.1. The first ranking of v is the ranking prior to v moving b from the bottom to the top and the second is the ranking just after v has moved b to the top. Choose any pair ac where a is above c in v's ranking. The third ranking of v is obtained by moving a above b in the second ranking so that $a > b > c$ in v's ranking. By independence of irrelevant alternatives, the global ranking after v has switched to the third ranking puts $a > b$ since all individual ab votes are the same as in the first ranking, where the global ranking placed $a > b$. Similarly $b > c$ in the global ranking since all individual bc votes are the same as in the second ranking, in which b was at the top of the global ranking. By transitivity the global ranking must put $a > c$ and thus the global ranking of a and c agrees with v.

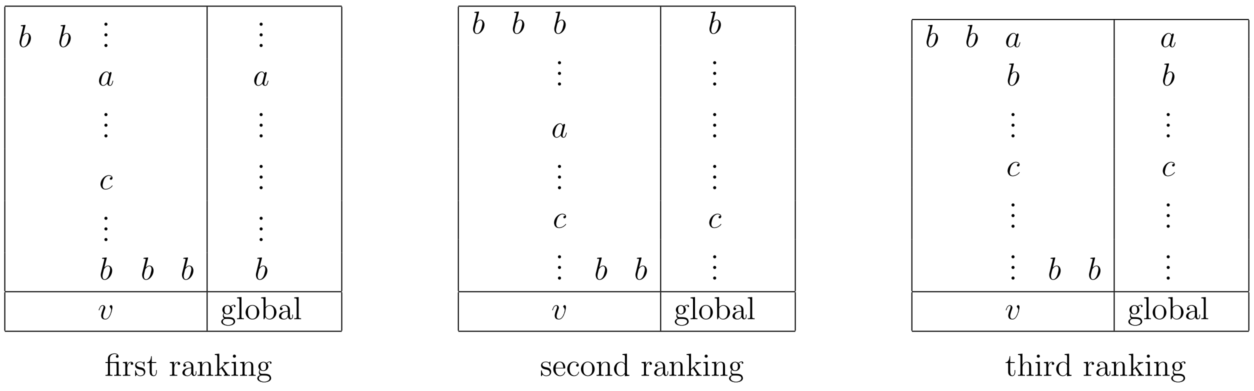

Figure 10.1: The three rankings that are used in the proof of Theorem 10.2.

Now all individuals except v can modify their rankings arbitrarily while leaving b in its extreme position and by independence of irrelevant alternatives, this does not affect the global ranking of $a > b$ or of $b > c$. Thus, by transitivity this does not affect the global ranking of a and c. Next, all individuals except v can move b to any position without affecting the global ranking of a and c.

At this point we have argued that independent of other individuals' rankings, the global ranking of a and c will agree with v's ranking. Now v can change its ranking arbitrarily, provided it maintains the order of a and c, and by independence of irrelevant alternatives the global ranking of a and c will not change and hence will agree with v. Thus, we conclude that for all a and c, the global ranking agrees with v independent of the other rankings except for the placement of b. But other rankings can move b without changing the global order of other elements. Thus, v is a dictator for the ranking of any pair of elements not involving b.

Note that v changed the relative order of a and b in the global ranking when it moved b from the bottom to the top in the previous argument. We will use this in a moment.

The individual v is also a dictator over every pair ab. Repeat the construction showing that v is a dictator for every pair ac not involving b only this time place c at the bottom. There must be an individual v_c who is a dictator for any pair such as ab not involving c. Since both v and v_c can affect the global ranking of a and b independent of each other, it must be that v_c is actually v. Thus, the global ranking agrees with v no matter how the other voters modify their rankings. ■

10.1.1 Randomization

An interesting randomized algorithm that satisfies unanimity and independence of irrelevant alternatives is to pick a random individual and use that individual's ranking as the output. This is called the “random dictator” rule because it is a randomization over dictatorships. An analogous scheme in the context of voting would be to select a winner with probability proportional to the number of votes for that candidate, because this is the same as selecting a random voter and telling that voter to determine the winner. Note that this method has the appealing property that as a voter, there is never any reason to strategize, e.g., voting for candidate a rather than your preferred candidate b because you think b is unlikely to win and you don’t want to throw away your vote. With this method, you should always vote for your preferred candidate.

10.1.2 Examples

Borda Count: Suppose we view each individual’s ranking as giving each item a score: putting an item in last place gives it one point, putting it in second-to-last place gives it two points, third-to-last place is three points, and so on. In this case, one simple way to combine rankings is to sum up the total number of points received by each item and then sort by total points. This is called the extended Borda Count method.

Let’s examine which axioms are satisfied by this approach. It is easy to see that it is a nondictatorship. It also satisfies unanimity: if every individual prefers a to b, then every individual gives more points to a than to b, and so a will receive a higher total than b. By Arrow’s theorem, the approach must fail independence of irrelevant alternatives, and indeed this is the case. Here is a simple example with three voters and four items $\{a, b, c, d\}$ where the independence of irrelevant alternatives axiom fails:

individual	ranking
1	*abcd*
2	*abcd*
3	*bacd*

In this example, a receives 11 points and is ranked first, b receives 10 points and is ranked second, c receives 6 points and is ranked third, and d receives 3 points and is ranked fourth. However, if individual 3 changes his ranking to *bcda*, then this reduces the total number of points received by a to 9, and so b is now ranked first overall. Thus, even though individual 3’s relative order of b and a did not change, and indeed no individual’s relative order of b and a changed, the global order of b and a did change.

Hare voting: An interesting system for voting is to have everyone vote for their favorite candidate. If some candidate receives a majority of the votes, he or she is declared the winner. If no candidate receives a majority of votes, the candidate with the fewest votes is dropped from the slate and the process is repeated.

The Hare system implements this method by asking each voter to rank all the candidates. Then one counts how many voters ranked each candidate as number one. If no candidate receives a majority, the candidate with the fewest number one votes is dropped from each voters ranking. If the dropped candidate was number one on some voters list, then the number two candidate becomes that voter's number one choice. The process of counting the number one rankings is then repeated.

We can convert the Hare voting system into a ranking method in the following way. Whichever candidate is dropped first is put in last place, whichever is dropped second is put in second-to-last place, and so on, until the system selects a winner, which is put in first place. The candidates remaining, if any, are placed between the first-place candidate and the candidates who were dropped, in an order determined by running this procedure recursively on just those remaining candidates.

As with Borda Count, the Hare system also fails to satisfy independence of irrelevant alternatives. Consider the following situation in which there are 21 voters that fall into four categories. Voters within a category rank individuals in the same order.

Category	Number of voters in category	Preference order
1	7	abcd
2	6	bacd
3	5	cbad
4	3	dcba

The Hare system would first eliminate d since d gets only three rank one votes. Then it would eliminate b since b gets only six rank one votes whereas a gets seven and c gets eight. At this point a is declared the winner since a has thirteen votes to c's eight votes. So, the final ranking is *acbd*.

Now assume that Category 4 voters who prefer b to a move b up to first place. This keeps their order of a and b unchanged, but it reverses the global order of a and b. In particular, d is first eliminated since it gets no rank one votes. Then c with five votes is eliminated. Finally, b is declared the winner with 14 votes, so the final ranking is *bacd*.

Interestingly, Category 4 voters who dislike a and have ranked a last could prevent a from winning by moving a up to first. Ironically this results in eliminating d, then c, with five votes and declaring b the winner with 11 votes. Note that by moving a up, category 4 voters were able to deny a the election and get b to win, whom they prefer over a.

10.2 Compressed Sensing and Sparse Vectors

Define a *signal* to be a vector $\mathbf{x}$ of length d, and define a *measurement* of $\mathbf{x}$ to be a dot-product of $\mathbf{x}$ with some known vector $\mathbf{a_i}$. If we wish to uniquely reconstruct $\mathbf{x}$ without any assumptions, then d linearly-independent measurements are necessary and sufficient.

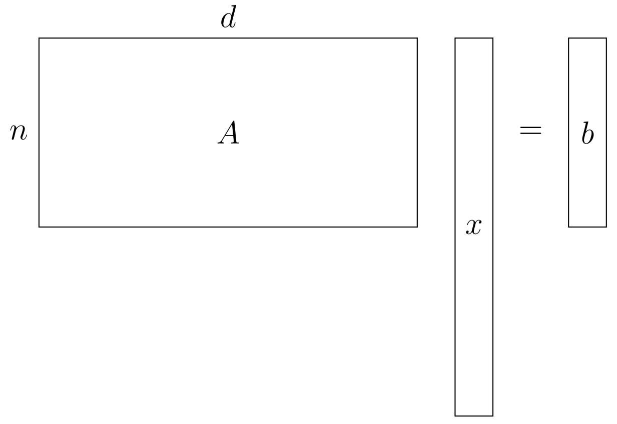

Figure 10.2: $A\mathbf{x} = \mathbf{b}$ has a vector space of solutions but possibly only one sparse solution. If the columns of A are unit length vectors that are pairwise nearly orthogonal, then the system has a unique sparse solution.

Given $\mathbf{b} = A\mathbf{x}$ where A is known and invertible, we can reconstruct $\mathbf{x}$ as $\mathbf{x} = A^{-1}\mathbf{b}$. In the case where there are fewer than d independent measurements and the rank of A is less than d, there will be multiple solutions. However, if we knew that $\mathbf{x}$ is sparse with $s \ll d$ nonzero elements, then we might be able to reconstruct $\mathbf{x}$ with far fewer measurements using a matrix A with $n \ll d$ rows. See Figure 10.2. In particular, it turns out that a matrix A whose columns are nearly orthogonal, such as a matrix of random Gaussian entries, will be especially well-suited to this task. This is the idea of compressed sensing. Note that we cannot make the columns of A be completely orthogonal since A has more columns than rows.

Compressed sensing has found many applications, including reducing the number of sensors needed in photography, using the fact that images tend to be sparse in the wavelet domain, and in speeding up magnetic resonance imaging in medicine.

10.2.1 Unique Reconstruction of a Sparse Vector

A vector is said to be s-sparse if it has at most s nonzero elements. Let $\mathbf{x}$ be a d-dimensional, s-sparse vector with $s \ll d$. Consider solving $A\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$ where A is an $n \times d$ matrix with $n < d$. The set of solutions to $A\mathbf{x} = \mathbf{b}$ is a subspace. However, if we restrict ourselves to sparse solutions, under certain conditions on A there is a unique s-sparse solution. Suppose that there were two s-sparse solutions, $\mathbf{x_1}$ and $\mathbf{x_2}$. Then $\mathbf{x_1} - \mathbf{x_2}$ would be a $2s$-sparse solution to the homogeneous system $A\mathbf{x} = \mathbf{0}$. A $2s$-sparse solution to the homogeneous equation $A\mathbf{x} = \mathbf{0}$ requires that some $2s$ columns of A be linearly dependent. Unless A has $2s$ linearly dependent columns there can be only one s-sparse solution.

The solution to the reconstruction problem is simple. If the matrix A has at least $2s$

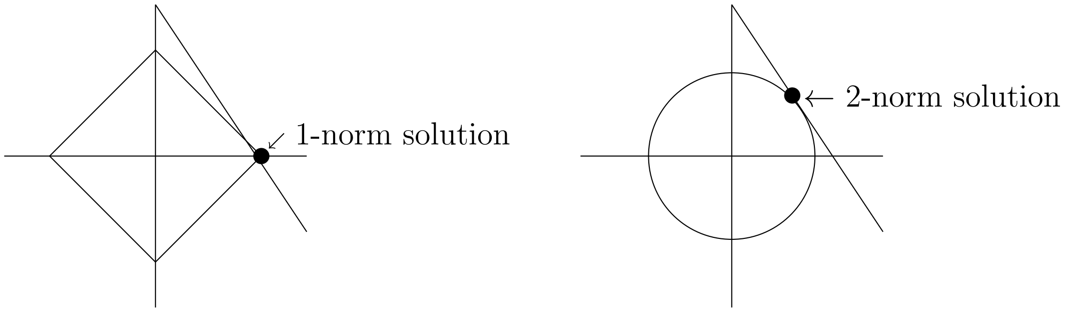

Figure 10.3: Illustration of minimum 1-norm and 2-norm solutions.

rows and the entries of A were selected at random from a standard Gaussian, then with probability one, no set of $2s$ columns will be linearly dependent. We can see this by noting that if we first fix a subset of $2s$ columns and then choose the entries at random, the probability that this specific subset is linearly dependent is the same as the probability that $2s$ random Gaussian vectors in a $2s$-dimensional space are linearly dependent, which is zero.[42] So, taking the union bound over all $\binom{d}{2s}$ subsets, the probability that any one of them is linearly dependent is zero.

The above argument shows that if we choose $n = 2s$ and pick entries of A randomly from a Gaussian, with probability one there will be a unique s-sparse solution. Thus, to solve for $\mathbf{x}$ we could try all $\binom{d}{s}$ possible locations for the nonzero elements in $\mathbf{x}$ and aim to solve $A\mathbf{x} = \mathbf{b}$ over just those s columns of A: any one of these that gives a solution will be the correct answer. However, this takes time $\Omega(d^s)$ which is exponential in s. We turn next to the topic of efficient algorithms, describing a polynomial-time optimization procedure that will find the desired solution when n is sufficiently large and A is constructed appropriately.

10.2.2 Efficiently Finding the Unique Sparse Solution

To find a sparse solution to $A\mathbf{x} = \mathbf{b}$, one would like to minimize the zero norm $\|\mathbf{x}\|_0$ over $\{\mathbf{x}|A\mathbf{x} = \mathbf{b}\}$, i.e., minimize the number of nonzero entries. This is a computationally hard problem. There are techniques to minimize a convex function over a convex set, but $||\mathbf{x}||_0$ is not a convex function, and with no further assumptions, it is NP-hard. With this in mind, we use the one-norm as a proxy for the zero-norm and minimize the one-norm $\|\mathbf{x}\|_1 = \sum_i |x_i|$ over $\{\mathbf{x}|A\mathbf{x} = \mathbf{b}\}$. Although this problem appears to be nonlinear, it can be solved by linear programming by writing $\mathbf{x} = \mathbf{u} - \mathbf{v}$, $\mathbf{u} \geq 0$, and $\mathbf{v} \geq 0$, and minimizing the linear function $\sum\limits_i u_i + \sum\limits_i v_i$ subject to $A\mathbf{u}$-$A\mathbf{v}$=$\mathbf{b}$, $\mathbf{u} \geq 0$, and $\mathbf{v} \geq 0$.

[42]This can be seen by selecting the vectors one at a time. The probability that the i^{th} new vector lies fully in the lower dimensional subspace spanned by the previous $i-1$ vectors is zero, and so by the union bound the overall probability is zero.

We now show if the columns of the n by d matrix A are unit length almost orthogonal vectors with pairwise dot products in the range $(-\frac{1}{2s}, \frac{1}{2s})$ that minimizing $\|\mathbf{x}\|_1$ over $\{\mathbf{x}|A\mathbf{x} = \mathbf{b}\}$ recovers the unique s-sparse solution to $A\mathbf{x}$=$\mathbf{b}$. The ij^{th} element of the matrix A^TA is the cosine of the angle between the i^{th} and j^{th} columns of A. If the columns of A are unit length and almost orthogonal, A^TA will have ones on its diagonal and all off diagonal elements will be small. By Theorem 2.8, if A has $n = s^2 \log d$ rows and each column is a random unit-length n-dimensional vector, with high probability all pairwise dot-products will have magnitude less than $\frac{1}{2s}$ as desired.[43] Here, we use $s^2 \log d$, a larger value of n compared to the existence argument in Section 10.2.1, but now the algorithm is computationally efficient.

Let $\mathbf{x_0}$ denote the unique s-sparse solution to $A\mathbf{x} = \mathbf{b}$ and let $\mathbf{x_1}$ be a solution of smallest possible one-norm. Let $\mathbf{z} = \mathbf{x_1} - \mathbf{x_0}$. We now prove that $\mathbf{z} = \mathbf{0}$ implying that $\mathbf{x_1} = \mathbf{x_0}$. First, $A\mathbf{z} = A\mathbf{x_1} - A\mathbf{x_0} = \mathbf{b} - \mathbf{b} = \mathbf{0}$. This implies that $A^TA\mathbf{z} = \mathbf{0}$. Since each column of A is unit length, the matrix A^TA has ones on its diagonal. Since every pair of distinct columns of A has dot-product in the range $(-\frac{1}{2s}, \frac{1}{2s})$, each off-diagonal entry in A^TA is in the range $(-\frac{1}{2s}, \frac{1}{2s})$. These two facts imply that unless $\mathbf{z} = \mathbf{0}$, every entry in $\mathbf{z}$ must have absolute value less than $\frac{1}{2s}||\mathbf{z}||_1$. If the j^{th} entry in $\mathbf{z}$ had absolute value greater than or equal to $\frac{1}{2s}||\mathbf{z}||_1$, it would not be possible for the j^{th} entry of $A^TA\mathbf{z}$ to equal 0 unless $||\mathbf{z}||_1 = 0$.

Finally let S denote the support of $\mathbf{x_0}$, where $|S| \leq s$. We now argue that $\mathbf{z}$ must have at least half of its ℓ_1 norm inside of S, i.e., $\sum_{j\in S} |z_j| \geq \frac{1}{2}||\mathbf{z}||_1$. This will complete the argument because it implies that the average value of $|z_j|$ for $j \in S$ is at least $\frac{1}{2s}||\mathbf{z}||_1$, which as shown above is only possible if $||\mathbf{z}||_1 = 0$. Let t_{in} denote the sum of the absolute values of the entries of $\mathbf{x_1}$ in the set S, and let t_{out} denote the sum of the absolute values of the entries of $\mathbf{x_1}$ outside of S. So, $t_{in} + t_{out} = ||\mathbf{x_1}||_1$. Let t_0 be the one-norm of $\mathbf{x_0}$. Since $\mathbf{x_1}$ is the minimum one norm solution, $t_0 \geq t_{in} + t_{out}$, or equivalently $t_0 - t_{in} \geq t_{out}$. But $\sum_{j\in S} |z_j| \geq t_0 - t_{in}$ and $\sum_{j\notin S} |z_j| = t_{out}$. This implies that $\sum_{j\in S} |z_j| \geq \sum_{j\notin S} |z_j|$, or equivalently, $\sum_{j\in S} |z_j| \geq \frac{1}{2}||\mathbf{z}||_1$, which as noted above implies that $||\mathbf{z}||_1 = 0$, as desired.

■

To summarize, we have shown the following theorem and corollary.

Theorem 10.3 *If matrix A has unit-length columns $\mathbf{a_1}, \ldots, \mathbf{a_d}$ and the property that $|\mathbf{a_i} \cdot \mathbf{a_j}| < \frac{1}{2s}$ for all $i \neq j$, then if the equation $A\mathbf{x} = \mathbf{b}$ has a solution with at most s nonzero coordinates, this solution is the unique minimum 1-norm solution to $A\mathbf{x} = \mathbf{b}$.*

Corollary 10.4 *For some absolute constant c, if A has n rows for $n \geq cs^2 \log d$ and each column of A is chosen to be a random unit-length n-dimensional vector, then with high probability A satisfies the conditions of Theorem 10.3 and therefore if the equation $A\mathbf{x} = \mathbf{b}$ has a solution with at most s nonzero coordinates, this solution is the unique minimum 1-norm solution to $A\mathbf{x} = \mathbf{b}$.*

[43] Note that the roles of "n" and "d" are reversed here compared to Theorem 2.8.

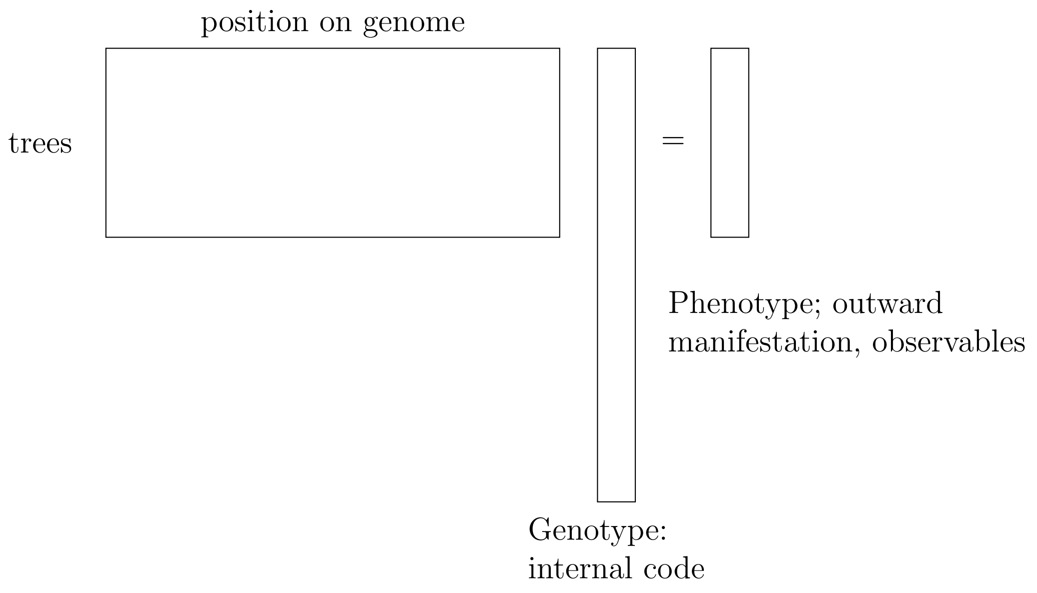

Figure 10.4: The system of linear equations used to find the internal code for some observable phenomenon.

The condition of Theorem 10.3 is often called *incoherence* of the matrix A. Other more involved arguments show that it is possible to recover the sparse solution using one-norm minimization for a number of rows n as small as $O(s\log(ds))$.

10.3 Applications

10.3.1 Biological

There are many areas where linear systems arise in which a sparse solution is unique. One is in plant breeding. Consider a breeder who has a number of apple trees and for each tree observes the strength of some desirable feature. He wishes to determine which genes are responsible for the feature so he can crossbreed to obtain a tree that better expresses the desirable feature. This gives rise to a set of equations $A\mathbf{x} = \mathbf{b}$ where each row of the matrix A corresponds to a tree and each column to a position on the genone. See Figure 10.4. The vector $\mathbf{b}$ corresponds to the strength of the desired feature in each tree. The solution $\mathbf{x}$ tells us the position on the genone corresponding to the genes that account for the feature. It would be surprising if there were two small independent sets of genes that accounted for the desired feature. Thus, the matrix should have a property that allows only one sparse solution.

10.3.2 Low Rank Matrices

Suppose L is a low rank matrix that has been corrupted by noise. That is, $A = L + R$. If the R is Gaussian, then principal component analysis will recover L from A. However, if L has been corrupted by several missing entries or several entries have a large noise added to them and they become outliers, then principal component analysis may be far off. However, if L is low rank and R is sparse, then L can be recovered effectively from $L+R$. To do this, find the L and R that minimize $\|L\|_* + \lambda \|R\|_1$.[44] Here the nuclear norm $\|L\|_*$ is the 1-norm of the vector of singular values of L and $||R||_1$ is the entrywise 1-norm $\sum_{ij} |r_{ij}|$. A small value of $\|L\|_*$ indicates a sparse vector of singular values and hence a low rank matrix. Minimizing $\|L\|_* + \lambda\|R\|_1$ subject to $L + R = A$ is a complex problem and there has been much work on it. The reader is referred to **Add references** Notice that we do not need to know the rank of L or the elements that were corrupted. All we need is that the low rank matrix L is not sparse and that the sparse matrix R is not low rank. We leave the proof as an exercise.

If A is a small matrix one method to find L and R by minimizing $||L||_* + ||R||_1$ is to find the singular value decomposition $A = U\Sigma V^T$ and minimize $||\Sigma||_1 + ||R||_1$ subject to $A = L + R$ and $U\Sigma V^T$ being the singular value decomposition of A. This can be done using Lagrange multipliers (**??**). Write $R = R^+ + R^-$ where $R^+ \geq 0$ and $R^- \geq 0$. Let

$$f(\sigma_i, r_{ij}) = \sum_{i=1}^{n} \sigma_i + \sum_{ij} r_{ij}^+ + \sum_{ij} r_{ij}^-.$$

Write the Lagrange formula

$$l = f(\sigma_i, r_{ij}) + \sigma_i \lambda_i g_i$$

where the g_i are the required constraints

1. $r_{ij}^+ \geq 0$
2. $r_{ij}^- \geq 0$
3. $\sigma_i \geq 0$
4. $a_{ij} = l_{ij} + r_{ij}$
5. $u_i^T u_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$
6. $v_i^T v_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$
7. $l_i = \sum u_i \sigma_i v_j^T$

[44] To minimize the absolute value of x write $x = u - v$ and using linear programming minimize $u + v$ subject to $u \geq 0$ and $v \geq 0$.

Conditions (5) and (6) insure that $U\Sigma V^T$ is the svd of some matrix. The solution is obtained when $\nabla(l) = 0$ which can be found by gradient descent using $\nabla^2(l)$.

An example where low rank matrices that have been corrupted might occur is aerial photographs of an intersection. Given a long sequence of such photographs, they will be the same except for cars and people. If each photo is converted to a vector and the vector used to make a column of a matrix, then the matrix will be low rank corrupted by the traffic. Finding the original low rank matrix will separate the cars and people from the back ground.

10.4 An Uncertainty Principle

Given a function $x(t)$, one can represent the function by the composition of sinusoidal functions. Basically one is representing the time function by its frequency components. The transformation from the time representation of a function to it frequency representation is accomplished by a Fourier transform. The Fourier transform of a function $x(t)$ is given by

$$f(\omega) = \int x(t)e^{-2\pi\omega t}dt$$

Converting the frequency representation back to the time representation is done by the inverse Fourier transformation

$$x(t) = \int f(\omega)e^{2\pi\omega t}d\omega$$

In the discrete case, $\mathbf{x} = [x_0, x_1, \ldots, x_{n-1}]$ and $\mathbf{f} = [f_0, f_1, \ldots, f_{n-1}]$. The Fourier transform is $\mathbf{f} = A\mathbf{x}$ with $a_{ij} = \frac{1}{\sqrt{n}}\omega^{ij}$ where ω is the principal n^{th} root of unity. The inverse transform is $\mathbf{x} = B\mathbf{f}$ where $B = A^{-1}$ has the simple form $b_{ij} = \frac{1}{\sqrt{n}}\omega^{-ij}$.

There are many other transforms such as the Laplace, wavelets, chirplets, etc. In fact, any nonsingular $n \times n$ matrix can be used as a transform.

10.4.1 Sparse Vector in Some Coordinate Basis

Consider $A\mathbf{x} = \mathbf{b}$ where A is a square $n \times n$ matrix. The vectors $\mathbf{x}$ and $\mathbf{b}$ can be considered as two representations of the same quantity. For example, $\mathbf{x}$ might be a discrete time sequence, $\mathbf{b}$ the frequency spectrum of $\mathbf{x}$, and the matrix A the Fourier transform. The quantity $\mathbf{x}$ can be represented in the time domain by $\mathbf{x}$ and in the frequency domain by its Fourier transform $\mathbf{b}$.

Any orthonormal matrix can be thought of as a transformation and there are many important transformations other than the Fourier transformation. Consider a transformation A and a signal $\mathbf{x}$ in some standard representation. Then $\mathbf{y} = A\mathbf{x}$ transforms the signal $\mathbf{x}$ to another representation $\mathbf{y}$. If A spreads any sparse signal $\mathbf{x}$ out so that the information contained in each coordinate in the standard basis is spread out to all coordinates in the second basis, then the two representations are said to be *incoherent*. A signal and its Fourier transform are one example of incoherent vectors. This suggests that if $\mathbf{x}$ is sparse, only a few randomly selected coordinates of its Fourier transform are needed to reconstruct $\mathbf{x}$. Below, we show that a signal cannot be too sparse in both its time domain and its frequency domain.

10.4.2 A Representation Cannot be Sparse in Both Time and Frequency Domains

There is an uncertainty principle that states that a time signal cannot be sparse in both the time domain and the frequency domain. If the signal is of length n, then the product of the number of nonzero coordinates in the time domain and the number of nonzero coordinates in the frequency domain must be at least n. This is the mathematical version of Heisenberg's uncertainty principle. Before proving the uncertainty principle we first prove a technical lemma.

In dealing with the Fourier transform it is convenient for indices to run from 0 to $n-1$ rather than from 1 to n. Let $x_0, x_1, \ldots, x_{n-1}$ be a sequence and let $f_0, f_1, \ldots, f_{n-1}$ be its discrete Fourier transform. Let $i = \sqrt{-1}$. Then $f_j = \frac{1}{\sqrt{n}} \sum\limits_{k=0}^{n-1} x_k e^{-\frac{2\pi i}{n}jk}, \quad j = 0, \ldots, n-1.$ In matrix form $\mathbf{f} = Z\mathbf{x}$ where $z_{jk} = e^{-\frac{2\pi i}{n}jk}$.

$$\begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{n-1} \end{pmatrix} = \frac{1}{\sqrt{n}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{-\frac{2\pi i}{n}} & e^{-\frac{2\pi i}{n}2} & \cdots & e^{-\frac{2\pi i}{n}(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & e^{-\frac{2\pi i}{n}(n-1)} & e^{-\frac{2\pi i}{n}2(n-1)} & \cdots & e^{-\frac{2\pi i}{n}(n-1)^2} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix}$$

If some of the elements of $\mathbf{x}$ are zero, delete the zero elements of $\mathbf{x}$ and the corresponding columns of the matrix. To maintain a square matrix, let n_x be the number of nonzero elements in $\mathbf{x}$ and select n_x consecutive rows of the matrix. Normalize the columns of the resulting submatrix by dividing each element in a column by the column element in the first row. The resulting submatrix is a Vandermonde matrix that looks like

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ a & b & c & d \\ a^2 & b^2 & c^2 & d^2 \\ a^3 & b^3 & c^3 & d^3 \end{pmatrix}$$

and is nonsingular.

Lemma 10.5 *If $x_0, x_1, \ldots, x_{n-1}$ has n_x nonzero elements, then $f_0, f_1, \ldots, f_{n-1}$ cannot have n_x consecutive zeros.*

$$\frac{1}{3}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & z & z^2 & z^3 & z^4 & z^5 & z^6 & z^7 & z^8 \\ 1 & z^2 & z^4 & z^6 & z^8 & z & z^3 & z^5 & z^7 \\ 1 & z^3 & z^6 & 1 & z^3 & z^6 & 1 & z^3 & z^6 \\ 1 & z^4 & z^8 & z^3 & z^7 & z^2 & z^6 & z & z^5 \\ 1 & z^5 & z & z^6 & z^2 & z^7 & z^3 & z^8 & z^4 \\ 1 & z^6 & z^3 & 1 & z^6 & z^3 & 1 & z^6 & z^3 \\ 1 & z^7 & z^5 & z^3 & z & z^8 & z^6 & z^4 & z^2 \\ 1 & z^8 & z^7 & z^6 & z^5 & z^4 & z^3 & z^2 & z \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{3}\begin{pmatrix} 3 \\ 1+z^3+z^6 \\ 1+z^6+z^3 \\ 3 \\ 1+z^3+z^6 \\ 1+z^6+z^3 \\ 3 \\ 1+z^3+z^6 \\ 1+z^6+z^3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Figure 10.5: The transform of the sequence 100100100.

Proof: Let $i_1, i_2, \dots, i_{n_x}$ be the indices of the nonzero elements of $\mathbf{x}$. Then the elements of the Fourier transform in the range $k = m+1, m+2, \dots, m+n_x$ are

$$f_k = \tfrac{1}{\sqrt{n}} \sum_{j=1}^{n_x} x_{i_j} e^{\frac{-2\pi i}{n} k i_j}$$

Note the use of i as $\sqrt{-1}$ and the multiplication of the exponent by i_j to account for the actual location of the element in the sequence. Normally, if every element in the sequence was included, we would just multiply by the index of summation.

Convert the equation to matrix form by defining $z_{kj} = \frac{1}{\sqrt{n}} \exp(-\frac{2\pi i}{n} k i_j)$ and write $\mathbf{f} = Z\mathbf{x}$ where now $\mathbf{x}$ is the vector consisting of the nonzero elements of the original $\mathbf{x}$. By its definition, $\mathbf{x} \neq 0$. To prove the lemma we need to show that $\mathbf{f}$ is nonzero. This will be true provided Z is nonsingular since $\mathbf{x} = Z^{-1}\mathbf{f}$. If we rescale Z by dividing each column by its leading entry we get the Vandermonde determinant which is nonsingular. ■

Theorem 10.6 *Let n_x be the number of nonzero elements in $\mathbf{x}$ and let n_f be the number of nonzero elements in the Fourier transform of $\mathbf{x}$. Let n_x divide n. Then $n_x n_f \geq n$.*

Proof: If $\mathbf{x}$ has n_x nonzero elements, $\mathbf{f}$ cannot have a consecutive block of n_x zeros. Since n_x divides n there are $\frac{n}{n_x}$ blocks each containing at least one nonzero element. Thus, the product of nonzero elements in $\mathbf{x}$ and $\mathbf{f}$ is at least n. ■

The Fourier transform of spikes proves that above bound is tight

To show that the bound in Theorem 10.6 is tight we show that the Fourier transform of the sequence of length n consisting of $\sqrt{n}$ ones, each one separated by $\sqrt{n}-1$ zeros, is the sequence itself. For example, the Fourier transform of the sequence 100100100 is 100100100. Thus, for this class of sequences, $n_x n_f = n$.

Theorem 10.7 *Let $S(\sqrt{n}, \sqrt{n})$ be the sequence of 1's and 0's with $\sqrt{n}$ 1's spaced $\sqrt{n}$ apart. The Fourier transform of $S(\sqrt{n}, \sqrt{n})$ is itself.*

Proof: Consider the columns $0, \sqrt{n}, 2\sqrt{n}, \ldots, (\sqrt{n}-1)\sqrt{n}$. These are the columns for which $S(\sqrt{n}, \sqrt{n})$ has value 1. The element of the matrix Z in the row $j\sqrt{n}$ of column $k\sqrt{n}$, $0 \le k < \sqrt{n}$ is $z^{nkj} = 1$. Thus, the product of these rows of Z times the vector $S(\sqrt{n}, \sqrt{n})$ equals $\sqrt{n}$ and the $1/\sqrt{n}$ normalization yields $f_{j\sqrt{n}} = 1$.

For rows whose index is not of the form $j\sqrt{n}$, the row b, $b \neq j\sqrt{n}$, $j \in \{0, \sqrt{n}, \ldots, \sqrt{n}-1\}$, the elements in row b in the columns $0, \sqrt{n}, 2\sqrt{n}, \ldots, (\sqrt{n}-1)\sqrt{n}$ are $1, z^b, z^{2b}, \ldots, z^{(\sqrt{n}-1)b}$ and thus $f_b = \frac{1}{\sqrt{n}}\left(1 + z^b + z^{2b} \cdots + z^{(\sqrt{n}-1)b}\right) = \frac{1}{\sqrt{n}} \frac{z^{\sqrt{n}b}-1}{z^b-1} = 0$ since $z^{b\sqrt{n}} = 1$ and $z^b \neq 1$. ■

ms better suited to perhaps a homework question

10.5 Gradient

The gradient of a function $f(\mathbf{x})$ of d variables, $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, at a point $\mathbf{x_0}$ is denoted $\bigtriangledown f(\mathbf{x_0})$. It is a d-dimensional vector with components $\frac{\partial f(\mathbf{x_0})}{\partial x_1}, \frac{\partial f(\mathbf{x_0})}{\partial x_2}, \ldots, \frac{\partial f(\mathbf{x_0})}{\partial x_d}$, where $\frac{\partial f}{\partial x_i}$ are partial derivatives. Without explicitly stating, we assume that the derivatives referred to exist. The rate of increase of the function f as we move from $\mathbf{x_0}$ in a direction $\mathbf{u}$ is $\bigtriangledown f(\mathbf{x_0}) \cdot \mathbf{u}$. So the direction of steepest descent is $-\bigtriangledown f(\mathbf{x_0})$; this is a natural direction to move to minimize f. But by how much should we move? A large move may overshoot the minimum. See Figure 10.6. A simple fix is to minimize f on the line from $\mathbf{x_0}$ in the direction of steepest descent by solving a one dimensional minimization problem. This gives us the next iterate $\mathbf{x_1}$ and we repeat. We do not discuss the issue of step-size any further. Instead, we focus on infinitesimal gradient descent, where, the algorithm makes infinitesimal moves in the $-\bigtriangledown f(\mathbf{x_0})$ direction. Whenever $\bigtriangledown \mathbf{f}$ is not the zero vector, we strictly decrease the function in the direction $-\bigtriangledown \mathbf{f}$, so the current point is not a minimum of the function f. Conversely, a point $\mathbf{x}$ where $\bigtriangledown \mathbf{f} = \mathbf{0}$ is called a *first-order local optimum* of f. A first-order local optimum may be a local minimum, local maximum, or a saddle point. We ignore saddle points since numerical error is likely to prevent gradient descent from stoping at a saddle point. In general, local minima do not have to be global minima, see Figure 10.6, and gradient descent may converge to a local minimum that is not a global minimum. When the function f is convex, this is not the case.

A function f of a single variable x is said to be convex if for any two points a and b, the line joining $f(a)$ and $f(b)$ is above the curve $f(\cdot)$. A function of many variables is convex if on any line segment in its domain, it acts as a convex function of one variable on the line segment.

Definition 10.1 *A function f over a convex domain is a convex function if for any two points $\mathbf{x}$ and $\mathbf{y}$ in the domain, and any λ in $[0,1]$ we have*

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \le \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}).$$

The function is concave if the inequality is satisfied with $\ge$ instead of $\le$.

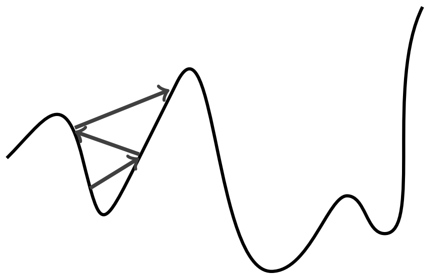

Figure 10.6: Gradient descent overshooting minimum

Theorem 10.8 *Suppose f is a convex, differentiable function defined on a closed bounded convex domain. Then any first-order local minimum is also a global minimum. Infinitesimal gradient descent always reaches the global minimum.*

Proof: We will prove that if $\mathbf{x}$ is a local minimum, then it must be a global minimum. If not, consider a global minimum point $\mathbf{y} \neq \mathbf{x}$. On the line joining $\mathbf{x}$ and $\mathbf{y}$, the function must not go above the line joining $f(\mathbf{x})$ and $f(\mathbf{y})$. This means for an infinitesimal $\varepsilon > 0$, moving distance ε from $\mathbf{x}$ towards $\mathbf{y}$, the function must decrease, so $\bigtriangledown \mathbf{f}(\mathbf{x})$ is not $\mathbf{0}$, contradicting the assumption that $\mathbf{x}$ is a local minimum. ■

The second derivatives $\frac{\partial^2}{\partial x_i \partial x_j}$ form a matrix, called the Hessian, denoted $H(f(\mathbf{x}))$. The Hessian of f at $\mathbf{x}$ is a symmetric $d \times d$ matrix with ij^{th} entry $\frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x})$. The second derivative of f at $\mathbf{x}$ in the direction $\mathbf{u}$ is the rate of change of the first derivative in the direction $\mathbf{u}$ from $\mathbf{x}$. It is easy to see that it equals

$$\mathbf{u}^T \, H(f(\mathbf{x}))\mathbf{u}.$$

To see this, note that the second derivative of f along the unit vector $\mathbf{u}$ is

$$\begin{aligned}
&\sum_j u_j \frac{\partial}{\partial x_j} (\bigtriangledown f(\mathbf{x}) \cdot \mathbf{u}) = \sum_j u_j \sum_i \frac{\partial}{\partial x_j} \left(u_i \frac{\partial f(\mathbf{x})}{\partial x_i} \right) \\
&= \sum_{j,i} u_j u_i \frac{\partial^2 f(\mathbf{x})}{\partial x_j \partial x_i}.
\end{aligned}$$

Theorem 10.9 *Suppose f is a function from a closed convex domain D in $\mathbf{R}^d$ to the reals and the Hessian of f exists everywhere in D. Then f is convex (concave) on D if and only if the Hessian of f is positive (negative) semi-definite everywhere on D.*

Gradient descent requires the gradient to exist. But, even if the gradient is not always defined, one can minimize a convex function over a convex domain efficiently, i.e., in polynomial time. Technically, one can only find an approximate minimum and the time depends on the error parameter as well as the presentation of the convex set. We do not go into these details. But, in principle we can minimize a convex function over a convex domain. We can also maximize a concave function over a concave domain. However, in general, we do not have efficient procedures to maximize a convex function over a convex domain. It is easy to see that at a first-order local minimum of a possibly non-convex function, the gradient vanishes. But second-order local decrease of the function may be possible. The steepest second-order decrease is in the direction of $\pm\mathbf{v}$, where, $\mathbf{v}$ is the eigenvector of the Hessian corresponding to the largest absolute valued eigenvalue.

10.6 Linear Programming

Linear programming is an optimization problem that has been carefully studied and is immensely useful. We consider linear programming problem in the following form where A is an $m \times n$ matrix, $m \le n$, of rank m, $\mathbf{c}$ is $1 \times n$, $\mathbf{b}$ is $m \times 1$, and $\mathbf{x}$ is $n \times 1$:

$$\max\ \mathbf{c} \cdot \mathbf{x} \quad \text{subject to} \quad A\mathbf{x} = \mathbf{b},\ \mathbf{x} \ge 0.$$

Inequality constraints can be converted to this form by adding slack variables. Also, we can do Gaussian elimination on A and if it does not have rank m, we either find that the system of equations has no solution, whence we may stop or we can find and discard redundant equations. After this preprocessing, we may assume that A 's rows are independent.

The simplex algorithm is a classical method to solve linear programming problems. It is a vast subject and is well discussed in many texts. Here, we will discuss the ellipsoid algorithm which is in a sense based more on continuous mathematics and is closer to the spirit of this book.

10.6.1 The Ellipsoid Algorithm

The first polynomial time algorithm for linear programming[45] was developed by Khachiyan based on work of Iudin, Nemirovsky and Shor and is called the ellipsoid algorithm. The algorithm is best stated for the seemingly simpler problem of determining whether there is a solution to $A\mathbf{x} \le \mathbf{b}$ and if so finding one. The ellipsoid algorithm starts with a large ball in d-space which is guaranteed to contain the polyhedron $A\mathbf{x} \le \mathbf{b}$. Even though we do not yet know if the polyhedron is empty or non-empty, such a ball can be found. The algorithm checks if the center of the ball is in the polyhedron, if it is, we have achieved our objective. If not, we know from the Separating Hyperplane Theorem of convex geometry that there is a hyperplane called the separating hyperplane through the center of the ball

[45]Although there are examples where the simplex algorithm requires exponential time, it was shown by Shanghua Teng and Dan Spielman that the expected running time of the simplex algorithm on an instance produced by taking an arbitrary instance and then adding small Gaussian perturbations to it is polynomial.

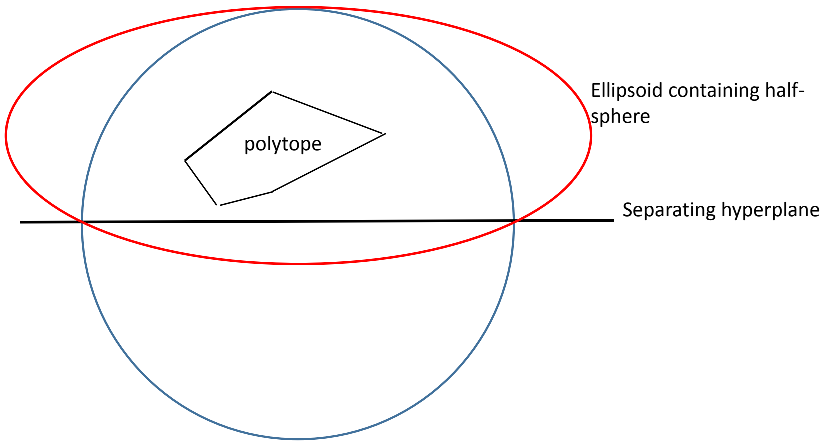

Figure 10.7: Ellipsoid Algorithm

such that the whole polytope lies in one of the half spaces.

We then find an ellipsoid which contains the ball intersected with this half-space. See Figure 10.7. The ellipsoid is guaranteed to contain $Ax \leq b$ as was the ball earlier. If the center of the ellipsoid does not satisfy the inequalities, then again there is a separating hyper plane and we repeat the process. After a suitable number of steps, either we find a solution to the original $A\mathbf{x} \leq \mathbf{b}$ or we end up with a very small ellipsoid. If the original A and $\mathbf{b}$ had integer entries, one can ensure that the set $A\mathbf{x} \leq \mathbf{b}$, after a slight perturbation which preserves its emptiness/non-emptiness, has a volume of at least some $\epsilon > 0$. If our ellipsoid has shrunk to a volume of less than this ϵ, then there is no solution. Clearly this must happen within $\log_\rho V_0/\epsilon = O(V_0 d/\epsilon)$ steps, where V_0 is an upper bound on the initial volume and ρ is the factor by which the volume shrinks in each step. We do not go into details of how to get a value for V_0, but the important points are that (i) only the logarithm of V_0 appears in the bound on the number of steps, and (ii) the dependence on d is linear. These features ensure a polynomial time algorithm.

The main difficulty in proving fast convergence is to show that the volume of the ellipsoid shrinks by a certain factor in each step. Thus, the question can be phrased as suppose E is an ellipsoid with center $\mathbf{x_0}$ and consider the half-ellipsoid E' defined by

$$E' = \{\mathbf{x} | \mathbf{x} \in E, \ \ \mathbf{a} \cdot (\mathbf{x} - \mathbf{x_0}) \geq 0\}$$

where $\mathbf{a}$ is some unit length vector. Let $\hat{E}$ be the smallest volume ellipsoid containing E'.

Show that

$$\frac{\mathrm{Vol}(\hat{E})}{\mathrm{Vol}(E)} \leq 1 - \rho$$

for some $\rho > 0$. A sequence of geometric reductions transforms this into a simple problem. Translate and then rotate the coordinate system so that $\mathbf{x_0} = 0$ and $\mathbf{a} = (1, 0, 0, \ldots, 0)$. Finally, apply a nonsingular linear transformation τ so that $\tau E = B = \{\mathbf{x}|\ |\mathbf{x}| = 1\}$, the unit sphere. The important point is that a nonsingular linear transformation τ multiplies the volumes of all sets by $|\det(\tau)|$, so that $\frac{\mathrm{Vol}(\hat{E})}{\mathrm{Vol}(E)} = \frac{\mathrm{Vol}(\tau(\hat{E}))}{\mathrm{Vol}(\tau(E))}$. The following lemma answers the question raised.

Lemma 10.10 *Consider the half-sphere $B' = \{\mathbf{x}|x_1 \geq 0, \quad |\mathbf{x}| \leq 1\}$. The following ellipsoid $\hat{E}$ contains B':*

$$\hat{E} = \left\{ \mathbf{x} \middle| \left(\frac{d+1}{d}\right)^2 \left(x_1 - \frac{1}{d+1}\right)^2 + \left(\frac{d^2-1}{d^2}\right)\left(x_2^2 + x_3^2 + \ldots + x_d^2\right) \leq 1 \right\}.$$

Further,

$$\frac{Vol(\hat{E})}{Vol(B)} = \left(\frac{d}{d+1}\right)\left(\frac{d^2}{d^2-1}\right)^{(d-1)/2} \leq 1 - \frac{1}{4d}.$$

The proof is left as an exercise (Exercise 10.27).

10.7 Integer Optimization

The problem of maximizing a linear function subject to linear inequality constraints, but with the variables constrained to be integers is called integer programming.

$$\text{Max } \mathbf{c} \cdot \mathbf{x} \qquad \text{subject to } A\mathbf{x} \leq \mathbf{b} \text{ with } x_i \text{ integers}$$

This problem is NP-hard. One way to handle the hardness is to relax the integer constraints, solve the linear program in polynomial time, and round the fractional values to integers. The simplest rounding, round each variable which is 1/2 or more to 1, the rest to 0, yields sensible results in some cases. The vertex cover problem is one of them. The problem is to choose a subset of vertices so that each edge is covered with at least one of its end points in the subset. The integer program is:

$$\text{Min} \sum_i x_i \quad \text{subject to } x_i + x_j \geq 1 \ \forall \text{ edges } (i,j); \ \ x_i \text{ integers }.$$

Solve the linear program. At least one variable for each edge must be at least 1/2 and the simple rounding converts it to one. The integer solution is still feasible. It clearly at most doubles the objective function from the linear programming solution and since the LP solution value is at most the optimal integer programming solution value, we are within a factor of two of the optimal.

10.8 Semi-Definite Programming

Semi-definite programs are special cases of convex programs. Recall that an $n \times n$ matrix A is positive semi-definite if and only if A is symmetric and for all $\mathbf{x} \in \mathbf{R}^n$, $\mathbf{x}^T A\mathbf{x} \geq 0$. There are many equivalent characterizations of positive semi-definite matrices. We mention one. A symmetric matrix A is positive semi-definite if and only if it can be expressed as $A = BB^T$ for a possibly rectangular matrix B.

A semi-definite program (SDP) is the problem of minimizing a linear function $\mathbf{c}^T\mathbf{x}$ subject to a constraint that $F = F_0 + F_1x_1 + F_2x_2 + \cdots + F_dx_d$ is positive semi-definite. Here $F_0, F_1, \ldots, F_d$ are given symmetric matrices.

This is a convex program since the set of $\mathbf{x}$ satisfying the constraint is a convex set. To see this, note that if $F(\mathbf{x}) = F_0 + F_1x_1 + F_2x_2 + \cdots + F_dx_d$ and $F(\mathbf{y}) = F_0 + F_1y_1 + F_2y_2 + \cdots + F_dy_d$ are positive semi-definite, then so is $F\big(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}\big)$ for $0 \leq \alpha \leq 1$. In principle, SDP's can be solved in polynomial time. It turns out that there are more efficient algorithms for SDP's than general convex programs and that many interesting problems can be formulated as SDP's. We discuss the latter aspect here.

Linear programs are special cases of SDP's. For any vector $\mathbf{v}$, let diag($\mathbf{v}$) denote a diagonal matrix with the components of $\mathbf{v}$ on the diagonal. Then it is easy to see that the constraints $\mathbf{v} \geq \mathbf{0}$ are equivalent to the constraint diag($\mathbf{v}$) is positive semi-definite. Consider the linear program:

Minimize $\mathbf{c}^T\mathbf{x}$ subject to $A\mathbf{x} = \mathbf{b}$; $\mathbf{x} \geq \mathbf{0}$.

Rewrite $A\mathbf{x} = \mathbf{b}$ as $A\mathbf{x} - \mathbf{b} \geq \mathbf{0}$ and $\mathbf{b} - A\mathbf{x} \geq \mathbf{0}$ and use the idea of diagonal matrices above to formulate this as an SDP.

A second interesting example is that of quadratic programs of the form:

Minimize $\frac{\left(\mathbf{c}^T\mathbf{x}\right)^2}{\mathbf{d}^T\mathbf{x}}$ subject to $A\mathbf{x} + \mathbf{b} \geq \mathbf{0}$.

This is equivalent to

Minimize t subject to $A\mathbf{x} + \mathbf{b} \geq \mathbf{0}$ and $t \geq \frac{\left(\mathbf{c}^T\mathbf{x}\right)^2}{\mathbf{d}^T\mathbf{x}}$.

This is in turn equivalent to the SDP

Minimize t subject to the following matrix being positive semi-definite:

$$\begin{pmatrix} \text{diag}(A\mathbf{x} + \mathbf{b}) & 0 & 0 \\ 0 & t & \mathbf{c}^T\mathbf{x} \\ 0 & \mathbf{c}^T\mathbf{x} & \mathbf{d}^T\mathbf{x} \end{pmatrix}.$$

Application to approximation algorithms.

An exciting area of application of SDP is in finding near-optimal solutions to some integer problems. The central idea is best illustrated by its early application in a breakthrough due to Goemans and Williamson [GW95] for the maximum cut problem which given a graph $G(V, E)$ asks for the cut $S, \bar{S}$ maximizing the number of edges going across the cut from S to $\bar{S}$. For each $i \in V$, let x_i be an integer variable assuming values ± 1 depending on whether $i \in S$ or $i \in \bar{S}$ respectively. Then the max-cut problem can be posed as

$$\text{Maximize} \sum_{(i,j)\in E} (1 - x_i x_j) \text{ subject to the constraints } x_i \in \{-1, +1\}.$$

The integrality constraint on the x_i makes the problem NP-hard. Instead replace the integer constraints by allowing the $\mathbf{x_i}$ to be unit length vectors. This enlarges the set of feasible solutions since ± 1 are just 1-dimensional vectors of length 1. The relaxed problem is an SDP and can be solved in polynomial time. To see that it is an SDP, consider $\mathbf{x_i}$ as the rows of a matrix X. The variables of our SDP are not X, but actually $Y = XX^T$, which is a positive semi-definite matrix. The SDP is

$$\text{Maximize} \sum_{(i,j)\in E} (1 - y_{ij}) \text{ subject to } Y \text{ positive semi-definite,}$$

which can be solved in polynomial time. From the solution Y, find X satisfying $Y = XX^T$. Now, instead of a ± 1 label on each vertex, we have vector labels, namely the rows of X. We need to round the vectors to ± 1 to get an S. One natural way to do this is to pick a random vector $\mathbf{v}$ and if for vertex i, $\mathbf{x_i} \cdot \mathbf{v}$ is positive, put i in S, otherwise put it in $\bar{S}$. Goemans and Wiiliamson showed that this method produces a cut guaranteed to be at least 0.878 times the maximum. The .878 factor is a big improvement on the previous best factor of 0.5 which is easy to get by putting each vertex into S with probability 1/2.

Application to machine learning.

As discussed in Chapter 5, kernel functions are a powerful tool in machine learning. They allow one to apply algorithms that learn linear classifiers, such as Perceptron and Support Vector Machines, to problems where the positive and negative examples might have a more complicated separating curve.

More specifically, a kernel K is a function from pairs of examples to reals such that for some implicit function ϕ from examples to $\Re^N$, we have $K(a, a') = \phi(a)^T \phi(a')$. (We are using "$a$" and "$a'$" to refer to examples, rather than x and x', in order to not conflict with the notation used earlier in this chapter.) Notice that this means that for any set of examples $\{a_1, a_2, \ldots, a_n\}$, the matrix A whose ij entry equals $K(a_i, a_j)$ is positive semi-definite. Specifically, $A = BB^T$ where the i^{th} row of B equals $\phi(a_i)$.

Given that a kernel corresponds to a positive semi-definite matrix, it is not surprising that there is a related use of semi-definite programming in machine learning. In particular, suppose that one does not want to specify up-front exactly which kernel an algorithm should use. In that case, a natural idea is instead to specify a space of kernel functions and allow the algorithm to select the best one from that space for the given data. Specifically, given some labeled training data and some unlabeled test data, one could solve for the matrix A over the combined data set that is positive semi-definite (so that it is a legal kernel function) and optimizes some given objective. This objective might correspond to separating the positive and negative examples in the labeled data while keeping the kernel simple so that it does not over-fit. If this objective is linear in the coefficients of A along with possibly additional linear constraints on A, then this is an SDP. This is the high-level idea of kernel learning, first proposed in [LCB$^+$04].

10.9 Bibliographic Notes

Arrow's impossibility theorem, stating that any ranking of three or more items satisfying unanimity and independence of irrelevant alternatives must be a dictatorship, is from [Arr50]. For extensions to Arrow's theorem on the manipulability of voting rules, see Gibbard [Gib73] and Satterthwaite [Sat75]. A good discussion of issues in social choice appears in [Lis13]. The results presented in Section 10.2.2 on compressed sensing are due to Donoho and Elad [DE03] and Gribonval and Nielsen [GN03]. See [Don06] for more details on issues in compressed sensing. The ellipsoid algorithm for linear programming is due to Khachiyan [Kha79] based on work of Shor [Sho70] and Iudin and Nemirovski [IN77]. For more information on the ellipsoid algorithm and on semi-definite programming, see the book of Grötschel, Lovász, and Schrijver [GLS12]. The use of SDPs for approximating the max-cut problem is due to Goemans and Williamson[GW95], and the use of SDPs for learning a kernel function is due to [LCB$^+$04].

10.10 Exercises

Exercise 10.1 *Select a method that you believe is good for combining individual rankings into a global ranking. Consider a set of rankings where each individual ranks b last. One by one move b from the bottom to the top leaving the other rankings in place. Does there exist a v as in Theorem 10.2 where v is the ranking that causes b to move from the bottom to the top in the global ranking. If not, does your method of combing individual rankings satisfy the axioms of unanimity and independence of irrelevant alternatives.*

Exercise 10.2 *Show that for the three axioms: non dictator, unanimity, and independence of irrelevant alternatives, it is possible to satisfy any two of the three.*

Exercise 10.3 *Does the axiom of independence of irrelevant alternatives make sense? What if there were three rankings of five items. In the first two rankings, A is number one and B is number two. In the third ranking, B is number one and A is number five. One might compute an average score where a low score is good. A gets a score of 1+1+5=7 and B gets a score of 2+2+1=5 and B is ranked number one in the global ranking. Now if the third ranker moves A up to the second position, A's score becomes 1+1+2=4 and the global ranking of A and B changes even though no individual ranking of A and B changed. Is there some alternative axiom to replace independence of irrelevant alternatives? Write a paragraph on your thoughts on this issue.*

Exercise 10.4 *Prove that in the proof of Theorem 10.2, the global ranking agrees with column v even if item b is moved down through the column.*

Exercise 10.5 *Let A be an m by n matrix with elements from a zero mean, unit variance Gaussian. How large must n be for there to be two or more sparse solutions to $A\mathbf{x} = \mathbf{b}$ with high probability. You will need to define how small s should be for a solution with at most s nonzero elements to be sparse.*

Exercise 10.6 *Section 10.2.1 showed that if A is an $n \times d$ matrix with entries selected at random from a standard Gaussian, and $n \geq 2s$, then with probability one there will be a unique s-sparse solution to $A\mathbf{x} = \mathbf{b}$. Show that if $n \leq s$, then with probability one there will* not *be a unique s-sparse solution. Assume $d > s$.*

Exercise 10.7 *Section 10.2.2 used the fact that $n = O(s^2 \log d)$ rows is sufficient so that if each column of A is a random unit-length n-dimensional vector, then with high probability all pairwise dot-products of columns will have magnitude less than $\frac{1}{2s}$. Here, we show that $n = \Omega(\log d)$ rows is necessary as well. To make the notation less confusing for this argument, we will use "m" instead of "d".*

Specifically, prove that for $m > 3^n$, it is not possible to have m unit-length n-dimensional vectors such that all pairwise dot-products of those vectors are less than $\frac{1}{2}$.

Some hints: (1) note that if two unit-length vectors $\mathbf{u}$ and $\mathbf{v}$ have dot-product greater than or equal to $\frac{1}{2}$ then $|\mathbf{u} - \mathbf{v}| \leq 1$ (if their dot-product is equal to $\frac{1}{2}$ then $\mathbf{u}$, $\mathbf{v}$, and the origin form an equilateral triangle). So, it is enough to prove that $m > 3^n$ unit-length vectors in $\Re^n$ cannot all have distance at least 1 from each other. (2) use the fact that the volume of a ball of radius r in $\Re^n$ is proportional to r^n.

Exercise 10.8 *Create a random 100 by 100 orthonormal matrix A and a sparse 100-dimensional vector $\mathbf{x}$. Compute $A\mathbf{x} = \mathbf{b}$. Randomly select a few coordinates of $\mathbf{b}$ and reconstruct $\mathbf{x}$ from the samples of $\mathbf{b}$ using the minimization of 1-norm technique of Section 10.2.2. Did you get $\mathbf{x}$ back?*

Exercise 10.9 *Let A be a low rank $n \times m$ matrix. Let r be the rank of A. Let $\tilde{A}$ be A corrupted by Gaussian noise. Prove that the rank r SVD approximation to $\tilde{A}$ minimizes $\left|A - \tilde{A}\right|_F^2$.*

Exercise 10.10 *Prove that minimizing $||x||_0$ subject to $Ax = b$ is NP-complete.*

Exercise 10.11 *When one wants to minimize $||x||_0$ subject to some constraint the problem is often NP-hard and one uses the 1-norm as a proxy for the 0-norm. To get an insite into this issue consider minimizing $||x||_0$ subject to the constraint that x lies in a convex region. For simplicity assume the convex region is a sphere with center more than the radius of the circle from the origin. Explore sparsity of solution when minimizing the 1-norm for values of x in the circular region with regards to location of the center.*

Exercise 10.12 *Express the matrix*

$$\begin{array}{ccccc} 2 & 17 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 9 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 13 & 2 & 2 & 2 & 2 \end{array}$$

as the sum of a low rank matrix plus a sparse matrix. To simplify the computation assume you want the low rank matrix to be symmetric so that its singular valued decomposition will be $V\Sigma V^T$.

Exercise 10.13 *Generate 100×100 matrices of rank 20, 40, 60 80, and 100. In each matrix randomly delete 50, 100, 200, or 400 entries. In each case try to recover the original matrix. How well do you do?*

Exercise 10.14 *Repeat the previous exercise but instead of deleting elements, corrupt the elements by adding a reasonable size corruption to the randomly selected matrix entries.*

End of sparse solutions, start of Uncertainty principle

Exercise 10.15 *Compute the Fourier transform of the sequence 1000010000.*

Exercise 10.16 *What is the Fourier transform of a Gaussian?*

Exercise 10.17 *What is the Fourier transform of a cyclic shift of a sequence?*

Exercise 10.18 *Let $S(i,j)$ be the sequence of i blocks each of length j where each block of symbols is a 1 followed by $j-1$ 0's. The number n=6 is factorable but not a perfect square. What is Fourier transform of $S\,(2,3)=$ 100100?*

Exercise 10.19 *Let Z be the n root of unity. Prove that $\left\{z^{bi}|0 \le i < n\right\} = \{z^i|0 \le i < n\}$ provide that b does not divide n.*

Exercise 10.20 *Show that if the elements in the second row of the $n \times n$ Vandermonde matrix*

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ a & b & \cdots & c \\ a^2 & b^2 & \cdots & c^2 \\ \vdots & \vdots & & \vdots \\ a^{n-1} & b^{n-1} & \cdots & c^{n-1} \end{pmatrix}$$

are distinct, then the Vandermonde matrix is nonsingular by expressing the determinant of the matrix as an $n-1$ degree polynomial in a.

Exercise 10.21 *Show that the following two statements are equivalent.*

1. *If the elements in the second row of the $n \times n$ Vandermonde matrix*

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ a & b & \cdots & c \\ a^2 & b^2 & \cdots & c^2 \\ \vdots & \vdots & & \vdots \\ a^{n-1} & b^{n-1} & \cdots & c^{n-1} \end{pmatrix}$$

   are distinct, then the Vandermonde matrix is nonsingular.

2. *Specifying the value of an n^{th} degree polynomial at $n+1$ points uniquely determines the polynomial.*

Exercise 10.22 *Many problems can be formulated as finding $\mathbf{x}$ satisfying $A\mathbf{x} = \mathbf{b}$ where A has more columns than rows and there is a subspace of solutions. If one knows that the solution is sparse but some error in the measurement $\mathbf{b}$ may prevent finding the sparse solution, they might add some residual error to $\mathbf{b}$ and reformulate the problem as solving for $\mathbf{x}$ and $\mathbf{r}$ subject to $A\mathbf{x} = \mathbf{b} + \mathbf{r}$ where $\mathbf{r}$ is the residual error. Discuss the advantages and disadvantages of each of the following three versions of the problem.*

1. *Set $\mathbf{r}$=0 and find $\mathbf{x}$= argmin $\|\mathbf{x}\|_1$ satisfying $A\mathbf{x} = \mathbf{b}$*

2. *Lasso: find $\mathbf{x}$= argmin $\left(\|\mathbf{x}\|_1 + \alpha \|\mathbf{r}\|_2^2\right)$ satisfying $A\mathbf{x} = \mathbf{b} + \mathbf{r}$*

3. *find $\underline{x}$=argmin $\|\mathbf{x}\|_1$ such that $\|\mathbf{r}\|_2^2 < \varepsilon$*

Exercise 10.23 *Let $M = L+R$ where L is a low rank matrix corrupted by a sparse noise matrix R. Why can we not recover L from M if R is low rank or if L is sparse?*

Exercise 10.24

1. *Suppose for a univariate convex function f and a finite interval D, $|f''(x)| \leq \delta |f'(x)|$ for every x. Then, what is a good step size to choose for gradient descent? Derive a bound on the number of steps needed to get an approximate minimum of f in terms of as few parameters as possible.*

2. *Generalize the statement and proof to convex functions of d variables.*

Exercise 10.25 *Prove that the maximum of a convex function over a polytope is attained at one of its vertices.*

Exercise 10.26 *Create a convex function and a convex region where the maximization problem has local maximuns.*

Exercise 10.27 *Prove Lemma 10.10.*

Exercise 10.28 *Consider the following symmetric matrix A:*

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & -1 \\ 1 & 1 & 2 & 0 \\ 1 & -1 & 0 & 2 \end{pmatrix}$$

Find four vectors $\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}, \mathbf{v_4}$ such that $a_{ij} = \mathbf{v_i}^T \mathbf{v_j}$ for all $1 \leq i, j \leq 4$. Also, find a matrix B such that $A = BB^T$.

Exercise 10.29 *Prove that if A_1 and A_2 are positive semi-definite matrices, then so is $A_1 + A_2$.*

7. Smoothed Analysis of Algorithms: The Simplex Algorithm Usually Takes a Polynomial Number of Steps, Journal of the Association for Computing Machinery (JACM), 51 (3) pp: 385463, May 2004. Conference Version: the Annual ACM Symposium on Theory of Computing, pages 296-305, 2001 (with Dan Spielman).