CS229 Lecture notes

原作者：[Andrew Ng](http://cs229.stanford.edu/)（[吴恩达](http://open.163.com/movie/2008/1/M/C/M6SGF6VB4_M6SGHFBMC.html)）

翻译：[CycleUser](https://zhuanlan.zhihu.com/python-kivy)

Part XII

Independent Components Analysis

Our next topic is Independent Components Analysis (ICA). Similar to PCA, this will find a new basis in which to represent our data. However, the goal is very different.

As a motivating example, consider the “cocktail party problem.” Here, n speakers are speaking simultaneously at a party, and any microphone placed in the room records only an overlapping combination of the n speakers’ voices. But let’s say we have n different microphones placed in the room, and be- cause each microphone is a different distance from each of the speakers, it records a different combination of the speakers’ voices. Using these micro- phone recordings, can we separate out the original n speakers’ speech signals?

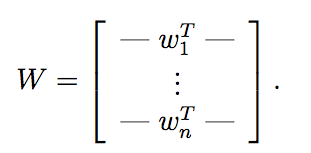
To formalize this problem, we imagine that there is some data s ∈ Rn that is generated via n independent sources. What we observe is

x = As,

where A is an unknown square matrix called the mixing matrix. Repeated observations gives us a dataset {x(i) ; i = 1, . . . , m}, and our goal is to recover the sources s(i) that had generated our data (x(i) = As(i)).

In our cocktail party problem, s(i) is an n-dimensional vector, and s(i) is the sound that speaker j was uttering at time i. Also, x(i) in an n-dimensional vector, and x j (i) is the acoustic reading recorded by microphone j at time i.

Let W = A−1 be the unmixing matrix. Our goal is to find W, so that given our microphone recordings x(i), we can recover the sources by computing s(i) = Wx(i). For notational convenience, we also let wiT denote the i-th row of W, so that

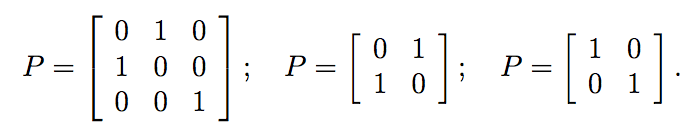


Thus, wi ∈ Rn, and the j-th source can be recovered by computing s(i) = w jT x ( i ) .

# 1 ICA ambiguities

To what degree can W = A−1 be recovered? If we have no prior knowledge about the sources and the mixing matrix, it is not hard to see that there are some inherent ambiguities in A that are impossible to recover, given only the x(i) ’s.

Specifically, let P be any n-by-n permutation matrix. This means that each row and each column of P has exactly one “1.” Here’re some examples of permutation matrices:



If z is a vector, then Pz is another vector that’s contains a permuted version of z’s coordinates. Given only the x(i)’s, there will be no way to distinguish between W and PW. Specifically, the permutation of the original sources is ambiguous, which should be no surprise. Fortunately, this does not matter for most applications.

Further, there is no way to recover the correct scaling of the wi’s. For instance, if A were replaced with 2A, and every s(i) were replaced with (0.5)s(i), then our observed x(i) = 2A · (0.5)s(i) would still be the same. More broadly, if a single column of A were scaled by a factor of α, and the corresponding source were scaled by a factor of 1/α, then there is again no way, given only the x(i)’s to determine that this had happened. Thus, we cannot recover the “correct” scaling of the sources. However, for the applications that we are concerned with—including the cocktail party problem—this ambiguity also does not matter. Specifically, scaling a speaker’s speech signal s(i) by some positive factor α affects only the volume of that speaker’s speech. Also, sign changes do not matter, and sj (i) and −sj (i) sound identical when played on a speaker. Thus, if the wi found by an algorithm is scaled by any non-zero real number, the corresponding recovered source si = wiT x will be scaled by the same factor; but this usually does not matter. (These comments also apply to ICA for the brain/MEG data that we talked about in class.)

Are these the only sources of ambiguity in ICA? It turns out that they are, so long as the sources si are non-Gaussian. To see what the difficulty is with Gaussian data, consider an example in which n = 2, and s ∼ N(0,I). Here, I is the 2x2 identity matrix. Note that the contours of the density of the standard normal distribution N(0,I) are circles centered on the origin, and the density is rotationally symmetric.

Now, suppose we observe some x = As, where A is our mixing matrix. The distribution of x will also be Gaussian, with zero mean and covariance E[xxT ] = E[AssT AT ] = AAT . Now, let R be an arbitrary orthogonal (less formally, a rotation/reflection) matrix, so that RRT = RTR = I, and let A′ = AR. Then if the data had been mixed according to A′ instead of A, we would have instead observed x′ = A′s. The distribution of x′ is also Gaussian, with zero mean and covariance E[x′(x′)T ] = E[A′ssT (A′)T ] = E[ARssT (AR)T ] = ARRT AT = AAT . Hence, whether the mixing matrix is A or A′ , we would observe data from a N (0, AAT ) distribution. Thus, there is no way to tell if the sources were mixed using A and A′. So, there is an arbitrary rotational component in the mixing matrix that cannot be determined from the data, and we cannot recover the original sources.

Our argument above was based on the fact that the multivariate standard normal distribution is rotationally symmetric. Despite the bleak picture that this paints for ICA on Gaussian data, it turns out that, so long as the data is not Gaussian, it is possible, given enough data, to recover the n independent sources.

# 2 Densities and linear transformations

Before moving on to derive the ICA algorithm proper, we first digress briefly to talk about the effect of linear transformations on densities.

Suppose we have a random variable s drawn according to some density ps(s). For simplicity, let us say for now that s ∈ R is a real number. Now, let the random variable x be defined according to x = As (here, x ∈ R, A ∈ R). Let px be the density of x. What is px?

Let W = A−1. To calculate the “probability” of a particular value of x, it is tempting to compute s = Wx, then evaluate ps at that point, and conclude that “px(x) = ps(Wx).” However, this is incorrect. For example, let s ∼ Uniform[0, 1], so that s’s density is ps(s) = 1{0 ≤ s ≤ 1}. Now, let A = 2, so that x = 2s. Clearly, x is distributed uniformly in the interval [0,2]. Thus, its density is given by px(x) = (0.5)1{0 ≤ x ≤ 2}. This does not equal ps (W x), where W = 0.5 = A−1 . Instead, the correct formula is px(x) = ps(W x)|W |.

More generally, if s is a vector-valued distribution with density ps, and x = As for a square, invertible matrix A, then the density of x is given by

px(x) = ps(Wx) · |W|,

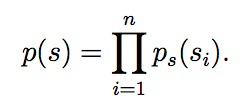
where W = A−1.

**Remark**. If you’ve seen the result that A maps [0, 1]n to a set of volume |A|, then here’s another way to remember the formula for px given above, that also generalizes our previous 1-dimensional example. Specifically, let A ∈ Rn×n be given, and let W = A−1 as usual. Also let C1 = [0, 1]n be the n-dimensional hypercube, and define C2 ={As:s∈C1}⊆Rn to be the image of C1 under the mapping given by A. Then it is a standard result in linear algebra (and, indeed, one of the ways of defining determinants) that the volume of C2 is given by |A|. Now, suppose s is uniformly distributed in [0, 1]n, so its density is ps(s) = 1{s ∈ C1}. Then clearly x will be uniformly distributed in C2. Its density is therefore found to be px(x) = 1{x ∈ C2}/vol(C2) (since it must integrate over C2 to 1). But using the fact that the determinant of the inverse of a matrix is just the inverse of the determinant, we have 1/vol(C2) = 1/|A| = |A−1| = |W|. Thus, px(x) = 1{x ∈ C2}|W| = 1{Wx ∈ C1}|W | = ps(W x)|W |.

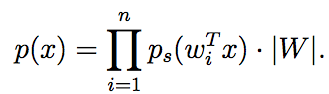
# 3 ICA algorithm

We are now ready to derive an ICA algorithm. The algorithm we describe is due to Bell and Sejnowski, and the interpretation we give will be of their algorithm as a method for maximum likelihood estimation. (This is different from their original interpretation, which involved a complicated idea called the infomax principal, that is no longer necessary in the derivation given the modern understanding of ICA.)

We suppose that the distribution of each source si is given by a density ps, and that the joint distribution of the sources s is given by



Note that by modeling the joint distribution as a product of the marginal, we capture the assumption that the sources are independent. Using our formulas from the previous section, this implies the following density on x = As = W−1s:

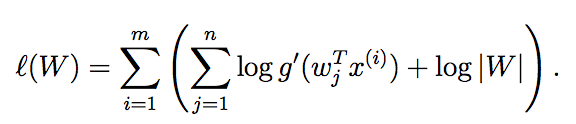


All that remains is to specify a density for the individual sources ps. Recall that, given a real-valued random variable z, its cumulative distribution function (cdf) F is defined by 

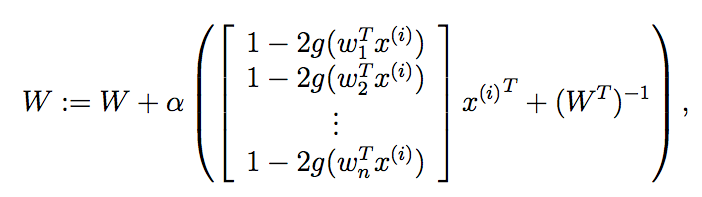
Also, the density of z can be found from the cdf by taking its derivative: pz(z) = F′(z).

Thus, to specify a density for the si’s, all we need to do is to specify some cdf for it. A cdf has to be a monotonic function that increases from zero to one. Following our previous discussion, we cannot choose the cdf to be the cdf of the Gaussian, as ICA doesn’t work on Gaussian data. What we’ll choose instead for the cdf, as a reasonable “default” function that slowly increases from 0 to 1, is the sigmoid function g(s) = 1/(1 + e−s). Hence, ps(s) = g′(s).1

The square matrix W is the parameter in our model. Given a training set {x(i);i = 1,...,m}, the log likelihood is given by



We would like to maximize this in terms W . By taking derivatives and using the fact (from the first set of notes) that ∇W|W| = |W|(W−1)T, we easily derive a stochastic gradient ascent learning rule. For a training example x(i), the update rule is:



where α is the learning rate. After the algorithm converges, we then compute s(i) = Wx(i) to recover the original sources.

**Remark.** When writing down the likelihood of the data, we implicitly assumed that the x(i)’s were independent of each other (for different values of i; note this issue is different from whether the different coordinates of x(i) are independent), so that the likelihood of the training set was given by . This assumption is clearly incorrect for speech data and other time series where the x(i)’s are dependent, but it can be shown that having correlated training examples will not hurt the performance of the algorithm if we have sufficient data. But, for problems where successive training examples are correlated, when implementing stochastic gradient ascent, it also sometimes helps accelerate convergence if we visit training examples in a randomly permuted order. (I.e., run stochastic gradient ascent on a randomly shuffled copy of the training set.)

1If you have prior knowledge that the sources’ densities take a certain form, then it is a good idea to substitute that in here. But in the absence of such knowledge, the sigmoid function can be thought of as a reasonable default that seems to work well for many problems. Also, the presentation here assumes that either the data x(i) has been preprocessed to have zero mean, or that it can naturally be expected to have zero mean (such as acoustic signals). This is necessary because our assumption that ps(s) = g′(s) implies E[s] = 0 (the derivative of the logistic function is a symmetric function, and hence gives a density corresponding to a random variable with zero mean), which implies E[x] = E[As] = 0.