We have come across two general techniques for estimating the probability of symbols in a sequence. One way to use conditional probabilities, $p(X_{j+1} \mid X_j,\ X_{j-1}, \ldots, X_{j-k+1})$, that is, a Markov model. The other way to take blocks of random variables and code blocks based on the marginal probabilites $p(X_1, \ldots, X_k), p(X_{k+1}, \ldots, X_{2k}), \ldots$. We do so by performing a linear transform on each block and encoding the transformed variables. This is called *transform coding*.

In what follows, we assume a stationary sequence of variables. The boundaries between blocks and the position of a variable within a block have no special significance. Typically we choose k to be a power of 2.

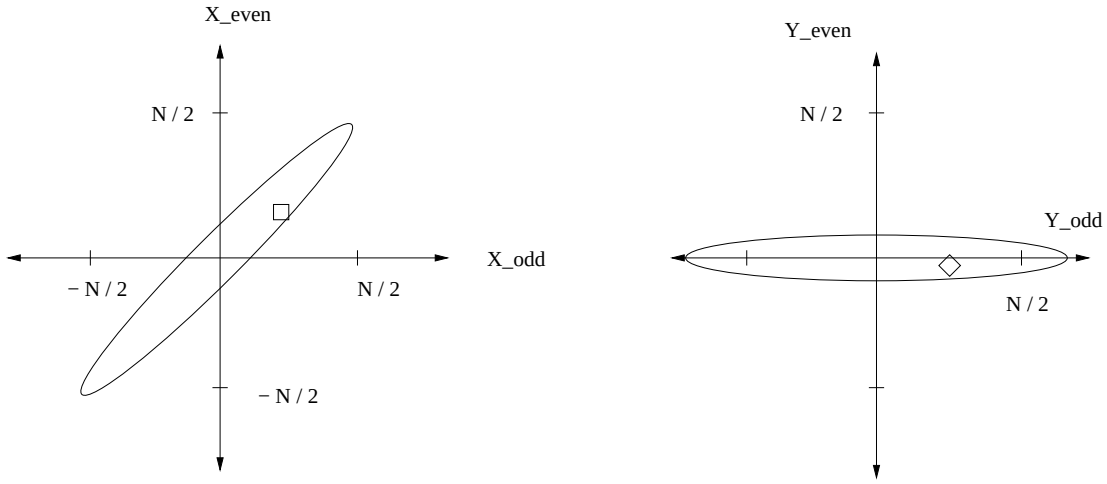
Transform Coding

When we transform code a sequence of blocks of size m, we are coding vectors in an m dimensional space. Take the case of $m = 2$. Let's plot the values (x_{2j-1}, x_{2j}) as points in a 2D space. If x_{2j-1} and x_{2j} tend to have similar values (which is the case for audio and image signals) then the plotted points are clustered near the diagonal line $X_{2j-1} = X_{2j}$.

We would like our code to make use of the fact that the joint probability $p(X_{2j-1}, X_{2j})$ is concentrated on the diagonal. The basic idea of how to do this is to rotate the axes by 45 degrees

$$\begin{bmatrix} Y_{2j-1} \\ Y_{2j} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} X_{2j-1} \\ X_{2j} \end{bmatrix}$$

and then code the Y values. Here is a sketch of the situation when the X_j have mean 0, as in the case of an audio file. (In the case of an image file, we can treat the intensities as being from -128 to 127. This doesn't ensure the mean is zero, but let's just pretend it is for now...)



Suppose the original X variables were quantized into N levels. For example, $N = 256$ for an image, or $N = 2^{16}$ for a high quality audio file. Suppose, for illustration purposes, that the data were distributed roughly uniformly for each X_j variable. If we were to encode the X_j individually, we would use $\log N$ bits for each value, or $2 \log N$ bits per pair (X_{2j-1}, X_{2j}). Along the major axis Y_{2j-1}, the data is distributed values in $[-\frac{\sqrt{2}}{2}N,\ \frac{\sqrt{2}}{2}\ N]$. Along the minor axis Y_{2j}, the data is

roughly uniformly distributed along a much smaller interval, say of length $N/8$, that is, roughly $[-N/16,\ N/16]$.

If we encode the Y values, then we would use about $\log(\sqrt{2}N) = \frac{1}{2} + \log N$ bits for each Y_{2j-1} and $\log \frac{N}{8} = \log(N) - 3$ bits for each Y_{2j}. Thus, we would save $2\frac{1}{2}$ bits for each two-sample block if we encode the Y's rather than the X's.

[ASIDE: For the moment, I ignore the fact that a little square in the X space maps to a little diamond in the Y space, and that this can lead to errors, i.e. lossy compression. I will come back to this a few lectures from now.]

Correlation

Let's take a more principled look at the transform coding problem. First, we need the following definition. Suppose we have two random variables X and X'. We say they are

- *positively correlated* if $\mathcal{E}(XX') > 0$
- *negatively correlated* if $\mathcal{E}(XX') < 0$
- *uncorrelated* if $\mathcal{E}(XX') = 0$.

If X, X' are always positive (e.g. image intensities) then they must have positive correlation. This is the case for image files. Negative correlations can only arise when at least one of the two variables has negative values.

We saw an example of correlation last lecture when we considered autocorrelation which is the correlation of samples of a stationary sequence. If we consider the expected squared difference of X_j and X_{j+m}, we get

$$\begin{aligned} \mathcal{E}(\ X_j - X_{j+m})^2 &= \mathcal{E}\ (X_j^2) + \mathcal{E}(X_{j+m})^2 - 2\mathcal{E}X_jX_{j+m}) \\ &= 2\mathcal{E}\ (X_j^2) - 2\mathcal{E}(X_jX_{j+m})). \end{aligned}$$

If the sequence is stationary, then the only quantity on the right side that depends on m is the autocorrelation. We expect the quantity on the left side to increase with m since samples further apart are less likely to have the same value. Thus we expect $\mathcal{E}(X_jX_{j+m})$ to *decrease* with m.

Definition of the transform coding problem

Let's use autocorrelation to do compression in a different way from last class. As in the example at the beginning of this lecture, we compress *blocks* of samples of a stationary sequence of variables by performing a linear transform of each block, such that the transformed variables are *uncorrelated*. We partition a stationary sequence of random variables $X_1,\ X_2,\ X_3,\ X_4,\ \ldots$ into k-tuples:

$$(X_1,\ \ldots, X_k),\ (X_{k+1}, \ldots,\ X_{2k}),\ (X_{2k+1}, \ldots, X_{3k}), \ldots$$

Let $\vec{X}$ be an k-tuple. We wish to transform each m-tuple $\vec{X}$ to a new k-tuple $\vec{Y}$ such that the elements of $\vec{Y}$ are uncorrelated. How to do this? Let

$$\vec{Y} \ \equiv\ \mathbf{U}^T\vec{X}$$

for some as yet unspecified matrix $\mathbf{U}$. Then,

$$\vec{Y}\ \vec{Y}^T = \mathbf{U}^T \vec{X} \vec{X}^T \mathbf{U}\ . \tag{1}$$

Take the expected value of both sides. Since the coefficients of matrix $\mathbf{U}$ are not random variables, we can pass the expectation operator to the inside. (Convince yourself by writing the whole thing out using summations.) This gives:

$$\mathcal{E}\{\vec{Y}\ \vec{Y}^T\}\ =\ \mathbf{U}^T\ \mathcal{E}\{\vec{X}\vec{X}^T\}\ \mathbf{U} \tag{2}$$

Our goal is to find a matrix $\mathbf{U}$ such that the Y_i and Y_j are uncorrelated where $i \neq j$. This means that our goal to find a $\mathbf{U}$ such that *the left hand side of Eq. (2) is a diagonal matrix.*

$\mathcal{E}\{\vec{X}\vec{X}^T\}$ is an $k \times k$ matrix whose elements are the expected value of products of two samples separated by some distance in the sequence. Since we are assuming the sequence is stationary, $\mathcal{E}\{X_i X_j\}$ depends only on $|i-j|$. We can estimate this correlation using the function $\hat{R}(i)$ from last lecture,

$$\hat{R}(\ i\ ) = \frac{1}{n-k} \sum_{j=k+1}^{n} x_j\ x_{j-i}$$

We define $k \times k$ matrix R with $R(0)$ on the diagonal, $R(1)$ on the first off diagonal, etc. For example if $k = 4$ then we would have:

$$\mathbf{R}\ \equiv\ \begin{bmatrix} R(0) & R(1) & R(2) & R(3) \\ R(1) & R(0) & R(1) & R(2) \\ R(2) & R(1) & R(0) & R(1) \\ R(3) & R(2) & R(1) & R(0) \end{bmatrix}$$

$\mathbf{R}$ is real and symmetric. Thus the eigenvalues are real and there exists a complete and orthogonal set of eigenvectors. (This just a basic fact of linear algebra which you should have seen in MATH 223. I include a proof of this fact in the Appendix below. You are not responsible for it. I include it here just for reference.)

Let $\mathbf{U}$ be a matrix whose columns are the eigenvectors of $\mathbf{R}$. We assume the columns are normalized, so that

$$\mathbf{U}^T\ \mathbf{U} = \mathbf{I}$$

$\mathbf{U}$ is an *orthonormal matrix*. Since the columns of $\mathbf{U}$ are the eigenvectors of $\mathbf{R}$, we have

$$\mathbf{U}\ \Lambda = \mathbf{R}\ \mathbf{U}$$

where Λ is the diagonal matrix of eigenvalues. Multiplying this eigenvector equation on the left by $\mathbf{U}^T$ yields

$$\Lambda\ =\ \mathbf{U}^T\ \mathbf{R}\ \mathbf{U}$$

But this has the same form as Eq. 2, where Λ is $\mathcal{E}\{\vec{Y}\ \vec{Y}^T\}$. In particular, the diagonal elements of Λ are the *eigenvalues* of $\mathbf{R}$.

Thus, we can decorrelate the variables in a block $\vec{X}$ by transforming:

$$\vec{Y}\ \equiv\ \mathbf{U}^T \vec{X}$$

This projects $\vec{X}$ onto the eigenvectors of the correlation matrix $\mathbf{R}$.

Appendix (you are not responsible for this)

If a matrix $\mathbf{R}$ is real and symmetric, then the eigenvalues are real and the matrix has an orthogonal set of eigenvectors.

Proof: First we show the eigenvalues are real. Let v be an eigenvector and λ its eigenvalue, i.e.

$$\mathbf{R}v = \lambda v \ .$$

Let v^H denote the Hermitian transpose of v i.e. the complex conjugate of the transpose. Then,

$$v^H(\mathbf{R}v) = v^H(\lambda v) = \lambda v^H v \ .$$

But since $\mathbf{R}$ is real and symmetric, $\mathbf{R} = \mathbf{R}^H$, and so

$$v^H\mathbf{R}v = v^H\mathbf{R}^H v = (\mathbf{R}v)^H v = \overline{\lambda}v^H \ v$$

Thus, $\lambda = \overline{\lambda}$ and so λ is real.

Next we show that there exists an orthogonal set of eigenvectors. Let v_1 and v_2 be two eigenvectors of $\mathbf{R}$ with eigenvalues, λ_1 and λ_2, respectively. Then

$$v_1^H(Rv_2) = (v_1^H R)v_2$$

The left side is $\lambda_2 v_1^H v_2$. The right side is $(R^H v_1)^T v_2$. But since $\mathbf{R}$ is real symmetric, the right side is $(Rv_1)^H v_2$ which is $\lambda_1 v_1^H v_2$. Thus,

$$\lambda_1 v_1^H v_2 = \lambda_2 v_1^H v_2$$

If $\lambda_1 \neq \lambda_2$ then $v_1^H v_2 = 0$, i.e. the eigenvectors are orthogonal.

If $\lambda_1 = \lambda_2$, then we can take linear combinations of v_1 and v_2 (Gram-Schmidt orthogonalization) to define two eigenvectors with eigenvalue $\lambda_1 = \lambda_2$ such that these eigenvectors are orthogonal.

If we have an orthogonal set of eigenvectors then we can trivially get an orthonormal set, just by dividing each vector by its length.