Voice Command Classification with PCA

Introduction

Data Collection and Processing

In order for S1XT33N to be able to differentiate between the commands you give it, the voice command signals must be linearly independent. This is easy — any real, recorded audio signal necessarily has some amount of approximately white noise, and therefore it is highly unlikely that you will be able to generate two linearly dependent signals. However, this is not actually a good thing for our purposes: it is nearly impossible to generate the exact same signal by recording a voice command twice, due to variations in your intonation, timing, and ambient noise. So, we want to choose commands that generate signals that are as close to orthogonal as possible.

To do this, we will use PCA, which is an application of the SVD. It highlights strong patterns in the dataset on which it is applied by projecting the dataset onto a basis that emphasizes variation. This way, when we project our collected signals on the PCA basis we generate from our training samples, the signals corresponding to each command should fall in a clear cluster, allowing S1XT33N to easily classify the commands.

Review of SVD and PCA

Singular Value Decomposition

The SVD decomposes any *m* x *n*, rank *r* matrix *A* into a sum of *r* rank-1 matrices:

$$A = \sigma_1 \vec{u}_1 \vec{v}_1^T + \sigma_2 \vec{u}_2 \vec{v}_2^T + ... + \sigma_r \vec{u}_r \vec{v}_r^T$$

such that:

1. The set of vectors $\{\vec{u}_1, \vec{u}_2, ..., \vec{u}_r\}$ is orthonormal.
2. The set of vectors $\{\vec{v}_1, \vec{v}_2, ..., \vec{v}_r\}$ is orthonormal.
3. $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r \geq 0$

The last restriction is imposed to ensure the uniqueness of the set $\{\sigma_1, \sigma_2, ..., \sigma_r\}$. We will discuss this further later.

We can express this in matrix form as follows:

$$A = U \Sigma V^T$$

. Let's develop this expression from the initial sum $A = \sigma_1 \vec{u}_1 \vec{v}_1^T + \sigma_2 \vec{u}_2 \vec{v}_2^T + ... + \sigma_r \vec{u}_r \vec{v}_r^T$.

Let U_1 be the matrix formed by joining the column vectors $\{\vec{u}_1, \vec{u}_2, ..., \vec{u}_r\}$, and let V_1 be the matrix formed by joining the column vectors $\{\vec{v}_1, \vec{v}_2, ..., \vec{v}_r\}$. We can now define a diagonal matrix S, where the values $\{\sigma_1, \sigma_2, ..., \sigma_r\}$ are on the diagonal. S, therefore, must be an r x r matrix (since it is diagonal and we have r values σ). Now, we can rephrase the sum

$$A = \sigma_1 \vec{u}_1 \vec{v}_1^T + \sigma_2 \vec{u}_2 \vec{v}_2^T + ... + \sigma_r \vec{u}_r \vec{v}_r^T$$

as the matrix multiplication

$$A = U_1 S V_1^T = \begin{bmatrix} \vec{u}_1 & \vec{u}_2 & ... & \vec{u}_r \end{bmatrix} \begin{bmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_r \end{bmatrix} \begin{bmatrix} \vec{v}_1 & \vec{v}_2 & ... & \vec{v}_r \end{bmatrix}^T, \; U_1 \text{ is } m \times r, \; V_1^T \text{ is } r \times n$$

Since $\{\vec{v}_1, \vec{v}_2, ..., \vec{v}_r\}$ and $\{\vec{u}_1, \vec{u}_2, ..., \vec{u}_r\}$ are orthonormal, $U_1^T U_1 = V_1^T V_1 = I_{r \times r}$.

This, however, is still not the full matrix form of the SVD. An advantage of the SVD is that it provides compression when the matrix A we are trying to decompose is not full rank (this is one of its primary use cases — this includes any time the A is not a square matrix). Since $r \leq m$, U_1 will not necessarily span the full m-dimensional space (nor will V_1 necessarily span the full n-dimensional space). The full matrix form of SVD is as follows:

$$A = U \Sigma V^T = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \left[\begin{array}{c|c} S & 0 \\ \hline 0 & 0 \end{array}\right] \begin{bmatrix} V_1 & V_2 \end{bmatrix}^T$$

where U_1 and V_1 are the matrices we defined earlier.

So, what are U_2 and V_2? To motivate this, let's find $\{\vec{u}_1, \vec{u}_2, ..., \vec{u}_r\}$ and $\{\vec{v}_1, \vec{v}_2, ..., \vec{v}_r\}$ that satisfy the requirements we posed earlier.

Each rank-1 matrix is given by $\vec{u}\vec{v}^T$, where $\vec{u}$ is a column vector and $\vec{v}$ is a row vector. The set of column vectors $\{\vec{u}_1, \vec{u}_2, ..., \vec{u}_r\}$ is the set of eigenvectors of AA^T and the set of row vectors $\{\vec{v}_1, \vec{v}_2, ..., \vec{v}_r\}$ is the set of eigenvectors of A^TA.

While A can be any shape, AA^T and A^TA are both square, symmetric, and positive semidefinite. We can use these properties to our advantage: the symmetry of AA^T and A^TA ensures that both sets of eigenvectors $\vec{u}_i$ and $\vec{v}_i$ are orthogonal.

Let's begin our review of the SVD with a toy example: the SVD for 2x2 matrices. Please consult this excellent article for a walkthrough if you feel you need it.

Principal Component Analysis

PCA Let X and Y be m x n matrices related by a linear transformation P, where X is the original dataset and Y is a re-representation of that dataset. P, therefore, is a change-of-basis matrix: the rows of P define a new basis for the columns of X.

$$PX = Y$$

We assume that P is orthonormal.

The fundamental question PCA asks is: *How do we construct the "best" new basis for our dataset from a linear combination of its original basis vectors?* Our judgment of the "best" basis clearly depends on which features we would like the re-expressed dataset to exhibit. We want to make recordings of different words look as different as possible to the MSP, so we

Let's consider the limitations of our signal-processing system (the combination of your mic board, front-end circuit, and MSP). We know the MSP has very limited memory, and that your mic board isn't exactly hi-fidelity. Therefore, we would like to **minimize noise and redundancy** so that we can keep the minimum amount of data in the MSP's memory that will still allow us to classify our commands successfully.

Let's address our desire to eliminate as much redundancy as possible from our dataset. We can quantify the redundancy within our dataset by calculating the *covariance matrix.*

Variance Consider two signal vectors $A = \{a_1, a_2, ..., a_n\}$ and $B = \{b_1, b_2, ...b_n\}$. The individual variances of A and B are defined as follows:

$$\sigma_A^2 = \langle A, A\rangle \qquad \sigma_B^2 = \langle B, B\rangle$$

Covariance The basic properties of covariance define a valid inner product on the vector space L^2. Hence, the covariance of two random variables can be expressed as the inner product of the corresponding *centered* (a.k.a., demeaned) variables. This is why we need to demean our dataset; otherwise, the inner product operation implicit in the matrix multiplication would not be equivalent to finding the covariance of each pair of vectors.

$$\text{cov}(X,Y) = \langle X - \mathbb{E}(X), Y - \mathbb{E}(Y)\rangle$$

So, returning to our vectors A and B, we can define their covariance $\sigma_{A,B}^2$ as

$$\sigma_{A,B}^2 = \frac{1}{n-1}\langle A, B\rangle$$

, where the term $\frac{1}{n-1}$ is a normalizing factor[a].

[a] We use $\frac{1}{n-1}$ instead of $\frac{1}{n}$ to normalize because the latter provides a biased estimation of variance for small n. The proof that $\frac{1}{n-1}$ is the correct normalization for the unbiased estimator is out-of-scope, but we encourage you to look it up if you like.

Let's construct our A matrix by horizontally stacking our signal vectors (so that each row represents one recording, and each column represents one timestep). Now, using the definition of covariance we defined above, we can construct the covariance matrix S_A as follows:

$$S_A = \frac{1}{n-1} AA^T$$

Note how the ij^{th} element of S_A is the covariance between recording i and recording j (or the variance, if $i = j$). Thus, the terms on the diagonal of the matrix are the variances, while the off-diagonal terms are covariances. A large covariance between a pair of recordings corresponds to a high degree of redundancy, while a covariance of zero corresponds to orthogonality (no redundancy). S_A describes all relationships between pairs of recordings.

Something to ponder: *how can we manipulate S_A to further reduce redundancy?*

In order to reduce redundancy, we want to minimize the covariance between the measurements. Optimally, all the covariances would be zero. We want to find a change of basis that gets us this result. Recall the structure we observed in the previous paragraph: if all the covariances are zero, the matrix S_A will be **diagonal**. So, we will now diagonalize S_A while taking into account the following assumptions made by PCA:

- PCA assumes that all basis vectors are orthonormal.
- PCA assumes that the directions corresponding to the largest variances are the most important.

You can probably guess what comes next: what's an easy way to diagonalize a matrix that also yields an orthonormal basis? Yep — we're going to use the eigendecomposition again.

Note that S_A was generated by finding AA^T, which implies that S_A is symmetric (try to prove this). Since we know a symmetric matrix is diagonalized by a matrix of its orthonormal eigenvectors, we can assert that for a symmetric matrix S_A,

$$S_A = EDE^T \qquad (1)$$

where D is a diagonal matrix and E is a matrix whose columns are the eigenvectors of S_A.

We will now construct P so that each row is an eigenvector of AA^T. From this construction, it follows that $P = E^T$. Substituting this P into equation (1) yields

$$S_A = P^T DP$$

.

PCA Procedure:

1. Construct the initial data matrix A such that the rows of A represent different recordings, and the columns of A represent different timesteps.
2. Demean the matrix A along its columns.
3. Calculate the SVD or the eigenvectors of the covariance matrix.

References

Slides by Miki Lustig (https://inst.eecs.berkeley.edu/ ee16b/fa18/lectures/Lecture9A.pdf)
Linear Algebra, Fourth Edition. Friedberg, Insel, and Spence.
An Article on SVD and its Applications. (http://www.ams.org/publicoutreach/feature-column/fcarc-svd)
Vector Spaces of Random Variables. (https://www.randomservices.org/random/expect/Spaces.html)

Notes written by Mia Mirkovic (2019)