Lecture Notes: Singular Value Decomposition, Principal Component Analysis, and Applications

Alice Tang* **and Shonushka Sawant†**
Team 5-5
CS 189/289 Final Project T

Contents

*alicetang@berkeley.edu
†shonushka.sen@berkeley.edu

1 Singular Value Decomposition

1.1 SVD Overview

Singular value decomposition (SVD) is a method of matrix decomposition that separates a rank-r matrix $A \in \mathbb{R}^{m\times n}$ into a sum of r rank-1 matrices, each written as a column times row. Specifically, we can find:

1) orthonormal vectors $\vec{u}_1, \ldots, \vec{u}_r \in \mathbb{R}^m$

2) orthonormal vectors $\vec{v}_1, \ldots, \vec{v}_r \in \mathbb{R}^n$

3) real, positive numbers $\sigma_1, \ldots, \sigma_r$ such that

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

The numbers $\sigma_1, \ldots, \sigma_r$ are called singular values and, by convention, we order them from the largest to smallest:

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$$

In its original form A has mn entries to be stored. In the SVD form each of the r terms is the product of a column of m entries with a row of n entries; therefore we need $r(m+n)$ numbers to store. This is an advantage when r is small relative to m and n, that is $r(m+n) \ll mn$. In a typical application the exact rank r may not be particularly small, but we may find that the first few singular values, say $\sigma_1, \ldots, \sigma_{\hat{r}}$ are much bigger than the rest, $\sigma_{\hat{r}+1}, \ldots, \sigma_r$. Then it is reasonable to discard the small singular values and approximate A as

$$A \approx \sigma_1 \vec{u}_1 \vec{v}_1^T + \sigma_2 \vec{u}_2 \vec{v}_2^T + \cdots + \sigma_{\hat{p}} \vec{u}_{\hat{p}} \vec{v}_{\hat{f}}^T$$

which has rank $= \hat{r}$, thus $\hat{r}(m+n) \ll mn$ numbers to store.

Besides enabling data compression, SVD allows us to extract important features of a data set as illustrated in the next example.

Example (Netflix): Suppose we have a $m \times n$ matrix that contains the ratings of m viewers for n movies. A truncated SVD as suggested above not only saves memory; it also gives insight into the preferences of each viewer. For example we can interpret each rank-1 $\text{matrix}\, \sigma_i \vec{u}_i \vec{v}_i^T$ to be due to a particular attribute, $e.g.$, comedy, action, sci-fi, or romance content. Then σ_i determines how strongly the ratings depend on the i th attribute, the entries of $\vec{v}_i^T$ score each movie with respect to this attribute, and the entries of $\vec{u}_i$ evaluate how much each viewer cares about this particular attribute. Then truncating the SVD as in (8) amounts to identifying a few key attributes that underlie the ratings.

This is useful, for example, in making movie recommendations.

Numerical Example: Finding the SVD

To find a SVD for A we use either the $n \times n$ matrix $A^T A$ or the $m\times\ m$ matrix AA^T. We will see later that these matrices have only real eigenvalues, r of which are positive and the remaining zero, and a complete set of orthonormal eigenvectors. For now we take this as a fact and outline the following procedure to find a SVD using $A^T A$:

1. Find the eigenvalues λ_i of $A^T A$ and order them from the largest to smallest, so that $\lambda_1 \geq \cdots \geq \lambda_r > 0$ and $\lambda_{r+1} = \cdots = \lambda_n = 0$

2. Find orthonormal eigenvectors $\vec{v}_i$, so that

$$A^T A \vec{v}_i = \lambda_i \vec{v}_i \quad i = 1, \ldots, r$$

3. Let $\sigma_i = \sqrt{\lambda_i}$ and obtain $\vec{u}_i$ from

$$A\vec{v}_i = \sigma_i \vec{u}_i \quad i = 1, \ldots, r$$

We will provide a justification for this procedure in the next lecture. For now we provide an example:
Example: Let

$$A = \begin{bmatrix} 1 & 2 \\ 1 & 2 \\ 1 & 2 \end{bmatrix}$$

since this matrix is rank-1 it is not difficult to write it as a column times row, but we will instead practice the general procedure above. Note that

$$A^T A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 1 & 2 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 6 & 12 \end{bmatrix}$$

and the eigenvalues of $A^T A$ are obtained from:

$$\det(\lambda I - A) = \det \begin{bmatrix} \lambda - 3 & -6 \\ -6 & \lambda - 12 \end{bmatrix} = \lambda^2 - 15\lambda = \lambda(\lambda - 15) = 0$$

Therefore, $\lambda_1 = 15$ and $\lambda_2 = 0$. Next we find an eigenvector $\vec{v}_1$ from

$$\begin{bmatrix} \lambda_1 - 3 & -6 \\ -6 & \lambda_1 - 12 \end{bmatrix} \vec{v}_1 = \begin{bmatrix} 12 & -6 \\ -6 & 3 \end{bmatrix} \vec{v}_1 = 0$$

with length normalized to one:

$$\vec{v}_1 = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

We compute the singular value from $\sigma_1 = \sqrt{\lambda_1} = \sqrt{15}$, and $\vec{u}_1$ from (10):

$$\vec{u}_1 = \frac{1}{\sigma_1} A \vec{v}_1 = \frac{1}{\sqrt{15}} \frac{1}{\sqrt{5}} \begin{bmatrix} 1 & 2 \\ 1 & 2 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \frac{1}{\sqrt{15}} \frac{1}{\sqrt{5}} \begin{bmatrix} 5 \\ 5 \\ 5 \end{bmatrix} = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Thus we have obtained the SVD:

$$A = \sigma_1 \vec{u}_1 \vec{v}_1^T = \sqrt{15} \begin{bmatrix} \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \end{bmatrix}$$

1.2 Applications of SVD

We can use SVD in computing the pseudoinverse, matrix approximation, and determining the rank, range and null space of a matrix.

2 Principal Component Analysis

2.1 Overview of PCA

PCA is an application of SVD in statistics that aims to find the most informative directions in a data set.

Suppose the $m \times n$ matrix A contains n measurements from m samples, for example n test scores for m students. If we subtract from each measurement the average over all samples, then each column of A is an m -vector with zero mean, and the $n \times n$ matrix

$$\frac{1}{m-1} A^T A$$

constitutes what is called the "covariance matrix" in statistics. Recall that the eigenvalues of this matrix are the singular values of A except for the scaling factor $m-1$, and its orthonormal eigenvectors correspond to $\vec{v}_{1,\ldots,\vec{v}_n}$ in the SVD of A

The vectors $\vec{v}_1, \vec{v}_2, \ldots$ corresponding to large singular values are called principal components and identify dominant directions in the data set along which the samples are clustered. The most significant direction is $\vec{v}_1$ corresponding to σ_1

As an illustration, the scatter plot below shows $n = 2$ midterm scores in a class of $m = 94$ students that I taught in the past. The data points are centered around zero because the class average is subtracted from the test scores. Each data point corresponds to a student and those in the first quadrant (both midterms ≥ 0) are those students who scored above average in each midterm. You can see that there were students who scored below average in the first and above average in the second, and vice versa. For this data set the covariance matrix is:

$$\frac{1}{93} A^T A = \begin{bmatrix} 297.69 & 202.53 \\ 202.53 & 292.07 \end{bmatrix}$$

where the diagonal entries correspond to the squares of the standard deviations 17.25 and 17.09 for Midterms 1 and 2, respectively. The positive sign of the (1,2) entry implies a positive correlation between the two midterm scores as one would expect.

The eigenvalues of $A^T A$, that is the singular values of A are $\sigma_1 = 215.08, \sigma_2 = 92.66$, and the corresponding eigenvectors of $A^T A$ are: $\vec{v}_1 = \begin{bmatrix} 0.7120 \\ 0.7022 \end{bmatrix} \quad \vec{v}_2 = \begin{bmatrix} -0.7022 \\ 0.7120 \end{bmatrix}$ The principal component $\vec{v}_1$ is superimposed on the scatter plot and we see that the data is indeed clustered around this line. Note that it makes an angle of $\tan^{-1}(0.7022/0.7120) \approx 44.6^\circ$ which is skewed slightly towards the Midterm 1 axis because the standard deviation in Midterm 1 was slightly higher than in Midterm 2. We may interpret the points above this line as students who performed better in Midterm 2 than in Midterm 1, as measured by their scores relative to the class average that are then compared against the factor $\tan(44.6^\circ)$ to account for the difference in standard deviations.

The $\vec{v}_2$ direction, which is perpendicular to $\vec{v}_1$, exhibits less variation than the $\vec{v}_1$ direction ($\sigma_2 = 92.66$ vs. $\sigma_1 = 215.08$)

3 Minimum Norm

Consider solving a linear system of equations in the form $Xw = y$.
$X \in \mathbb{R}^{nxd}$ is your data vector with n data points and d dimensions. w is your feature weights or coefficients. y are your observations or labels that you wish to predict or estimate.

With training data X_{train} and y_{train}, or any data X and y which you wish yo model, you want to be able to come up with an estimate for w in the form $\hat{y} = x^T w$ that minimizes the error or residual $\|Xw - y\|_2^2$.

3.1 4 Scenarios of Linear Equations

There are 4 scenarios in which we can determine a solution for w:

1. X is square ($n = d$) and full rank.
2. X is tall or over determined ($n > d$) and full rank.
3. X is wide or under determined ($n < d$) and full rank.
4. X is rectangular ($n \neq d$) and not full rank.

For (1.), to find w we can simply invert X to get our solution:

$$w = X^{-1}y$$

Note that $X^{-1}X = XX^{-1} = I_{nxn}$.

For (2.), we want to solve a linear regression problem where our estimate or prediction is of the form $\hat{y} = Xw$, where for each point we have $\hat{y}_i = x_i^T w$. We can try to find an approximate solution for w that will minimize the error or residual between $\hat{y}$ and y: $\min_w \|Xw - y\|_2^2$. This will be our least squares solution.

$$w = (X^T X)^{-1} X^T y$$

Note that $(X^T X)^{-1} X^T X = I_{nxn}$, making $(X^T X)^{-1} X^T$ a left inverse for X.

For (3.), our problem $Xw = y$ has infinite solutions. In such a scenario, we are interested in the minimum-norm solution, so we can reformulate our problem:

$$\min_w \|w\| \text{ s.t. } Xw = y$$

Generally, these type of problems are useful for constrained optimization and control problems, which you may have encountered in other engineering courses.

All of our solutions to this problem will have the form $\{w|Xw = y\} = \{w_r + w_n | w_r \in R(X), w_n \in N(X)\}$, where $R(X)$ is the range of the columns of X and $N(X)$ is the null space of the columns of X.

The least norm solution will be:

$$w = X^T (XX^T)^{-1} y$$

Note that $XX^T (XX^T)^{-1} = I_{nxn}$, making $X^T (XX^T)^{-1}$ a right inverse for X.

To see that w above is the least norm solution, let us consider another solution z, so we have $y = Xz = Xw$. Then $X(z - w) = 0$. Then we see

$$(z - w)^T w = (z - w)^T X^T (XX^T)^{-1} y = (X(z - w))^T (XX^T)^{-1} y = 0$$

So

$$\|z\|^2 = \|z + w - w\|^2 = \|w\|^2 + \|z - w\|^2 \geq \|w\|^2$$

Note that cross terms cancel. From this, we see that our least norm solution has the smallest norm.

3.2 SVD Perspective

We will now see that we can solve the systems of linear equations using an SVD perspective to give us a way to create a form of an inverse for our solution. This inverse is called the Moore Penrose Pseudo-Inverse.

3.2.1 Moore Penrose Pseudo-Inverse

When solving linear least squares problems in the form $Xw = y$ (X data matrix, w coefficients, y observations to predict). Often, X is not square or directly invertible.

The Moore-Penrose pseudoinverse X^+ gives us a generalization of the inverse matrix that can help us solve linear least squares solutions, $w = X^+y$.

A computationally simple and accurate of computing the pseudoinverse involves using the singular value decomposition. If $X = USV^T$, then the pseudo-inverse would be

$$X^+ = VS^+U^T$$

where S^+ of a diagonal matrix of singular values S is obtained by taking the reciprocal of the nonzero diagonal entries, then transposing that matrix $(S^{-1})^T$.

3.2.2 Solving linear equation with the Pseudoinverse

The pseudo-inverse works for the overdetermined and underdermined system of equations case. In Figure 1, we can see how we can use SVD to derive the Moore-Penrose Pseudoinverse to solve systems of linear equations

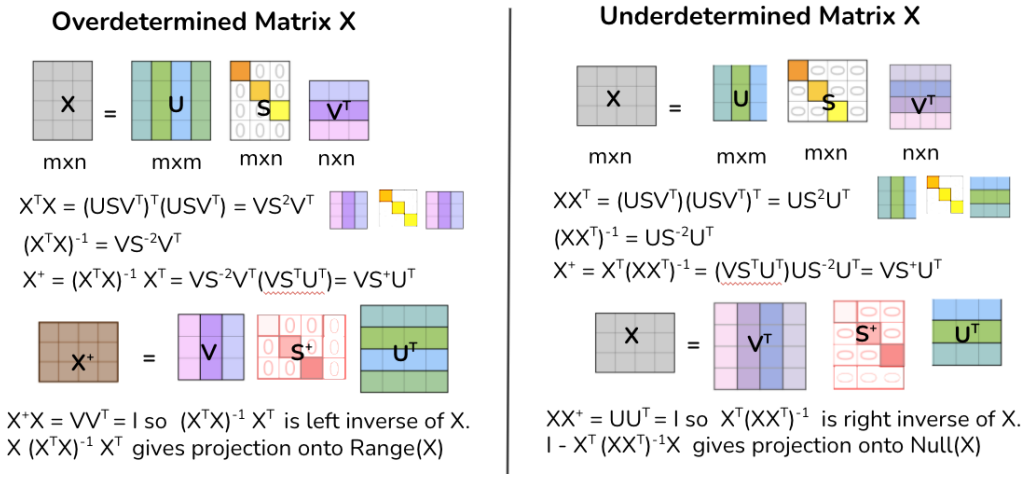


Figure 1: SVD form of the pseudoinverse for tall and wide matrices.

The following table summarizes the block form of the pseudo-inverse for all 4 scenarios listed above.

	row m, col n, rank r	$\boldsymbol{X}^+$	=	$\boldsymbol{V}$	$\boldsymbol{S}^+$	$\boldsymbol{U}$
1.	$m = n = r$	$\boldsymbol{X}^{-1}$	=	$\begin{bmatrix} \boldsymbol{V}_R \end{bmatrix}$	$\begin{bmatrix} \boldsymbol{S}^{-1} \end{bmatrix}$	$\begin{bmatrix} \boldsymbol{U}_R^T \end{bmatrix}$
2.	$m > n, n = r$	$\boldsymbol{X}^+$ $\boldsymbol{X}^{-L}$	=	$\begin{bmatrix} \boldsymbol{V}_R \end{bmatrix}$	$\begin{bmatrix} \boldsymbol{S}^{-1} & 0 \end{bmatrix}$	$\begin{bmatrix} \boldsymbol{U}_R^T \\ \boldsymbol{U}_N^T \end{bmatrix}$
3.	$m = r, m < n$	$\boldsymbol{X}^+$ $\boldsymbol{X}^{-R}$	=	$\begin{bmatrix} \boldsymbol{V}_R & \boldsymbol{V}_N \end{bmatrix}$	$\begin{bmatrix} \boldsymbol{S}^{-1} \\ 0 \end{bmatrix}$	$\begin{bmatrix} \boldsymbol{U}_R^T \end{bmatrix}$
4.	$m \neq n \neq r$	$\boldsymbol{X}^+$	=	$\begin{bmatrix} \boldsymbol{V}_R & \boldsymbol{V}_N \end{bmatrix}$	$\begin{bmatrix} \boldsymbol{S}^{-1} & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} \boldsymbol{U}_R^T \\ \boldsymbol{U}_N^T \end{bmatrix}$

3.2.3 Deriving Minimum Norm Solution Using Lagrangian Multipliers

Let us now derive the minimum norm solution for underdetermined equations using Lagrange Multipliers.

Our problem is formulated as:

$$\min_w \|w\|_2$$
$$\text{s.t.} \quad Xw = y$$

Let us introduce Lagrange Multipliers:

$$L(w, \lambda) = w^T w + \lambda^T (Xw - y)$$

Taking derivatives:

$$\nabla_w L = 2w + X^T \lambda = 0 \Rightarrow w = -\frac{X^T \lambda}{2}$$
$$\nabla_\lambda L = Xw - y = 0$$

Substituting w from the first equation into the 2nd equation, we get:

$$Xw - y = -X\frac{X^T \lambda}{2} - y = 0 \Rightarrow \lambda = -2(XX^T)^{-1} y$$

And therefore we plug back into the first equation to get:

$$w = \frac{X^T \lambda}{2} = (X^T)(XX^T)^{-1} y$$

which is our minimum norm solution.

3.2.4 Extension: General Norm Optimization

More generally, we may have a problem of the form

$$\min_w \|Xw - y\|$$
$$\text{subject to} \quad Cw = d$$

Here, least squares and least norm problems are a special case of the formulation above. The problem formulation is equivalent to saying:

$$\begin{aligned} &\min_{w} \frac{1}{2}\|Xw - y\|^2 \\ &\text{subject to} \quad Cw = d \end{aligned}$$

The Lagrangian is:

$$\begin{aligned} L(x, \lambda) &= \frac{1}{2}\|Xw - y\|^2 + \lambda^T(Cw - d) \\ &= \frac{1}{2}w^T X^T X w - y^T X w + \frac{1}{2}y^T y + \lambda^T C w - \lambda^T d \end{aligned}$$

Taking partial derivative to x and λ and setting to 0:

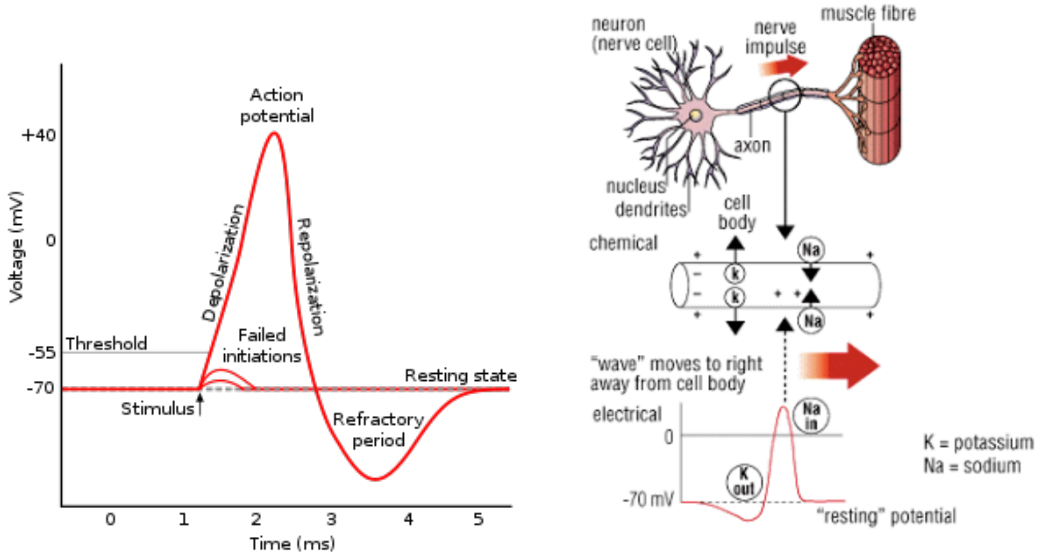
$$\begin{aligned} \nabla_w L &= X^T X w - X^T y + C^T \lambda = 0 \\ \nabla_\lambda L &= Cw - d = 0 \end{aligned}$$

or

$$\begin{bmatrix} X^T X & X^T \\ X & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} X^T y \\ d \end{bmatrix}$$

If the left-most block matrix is invertible, then we will have:

$$\begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} X^T X & X^T \\ X & 0 \end{bmatrix}^{-1} \begin{bmatrix} X^T y \\ d \end{bmatrix}$$



(a) The form of the change in membrane voltage during an action potential.

(b) Propogation of an action potential down an axon.

Figure 2: Neuron Action Potentials.

4 Introduction to Brain Computer Interface

Brain Computer Interface (BCI) refers to the idea of connecting the human brain with an external device (e.g. computer, robot). Successful integration can be applied for various purposes such as research, augmenting human abilities, controlling devices through thought, or treating neurological disorders.

Generally in the process of implementing a BCI, a device will be used to record electrical activity from the brain, which can be placed superficially on the head or implanted in the brain through surgery. These signals will need to be processed and understood in order to extract meaningful information. We will give a basic overview of basic neural signal processing workflow, and how dimensionality reduction can be useful.

4.1 Electrophysiology Basics

Neurons conduct electrical signals in the brain via action potentials (AP). These occur when the electrochemical gradient across the cell membranes reaches a level that causes ion movement, generating a 'current' as the neuron propogates this signal across axons. Neurotransmitters, a type of chemical signal, will then be released to neighboring neurons. With enough neurotransmitter release, ion flow across neighboring neurons will activate another action potential and pass on the signal around the brain.

These electrical impulses can be picked up with sensors. Figure 2a shows us the form of an action potential, and Figure 2b shows how these action potentials propogate through neurons.

4.2 Neuronal Recordings and Applications

Electrical impulses from neurons can be picked up by electrodes, although the type and quality of the signal will depend upon many factors. At a small scale, it is possible to record from a single neuron. Usually in humans, recordings are done via extracellular recordings, where the electrodes are placed directly on the brain (called ECoG, or electrocorticography). These recordings usually pick up an ensemble of neuron signals, but some single neuron recordings can possibly be extracted. At an even higher and non-invasive level, electrical signals can be picked up by electrocorticography (EEG), which involved recording from the skin. These signals are usually noisy and give average electrical

activity in a large region of the brain. Usually, there can be high levels features that can be extracted, such as time, frequency, space, and phase.

For recordings at a level where single neuron spikes can be picked up, one technique widely used is spike detection and sorting. From ensemble waveforms, spikes can be extracted, and then classified based upon the waveform characteristics of the spike to be associated with individual neurons or independent spiking events. Knowing what class a spike comes from is useful for various applications by then looking at spike frequency. By classifying and bringing meaning to the neuronal signals picked up by the sensors, we can then use that to extract meaningful commands to help us control a computer or device! This is the appeal of brain computer interface!

4.3 Spike Sorting

With extracellular recordings, usually the resolution is at a level where we can extract signals from nearby individual neurons. Let us walk through the step of processing this signal:

1. Filtering: The first step after obtaining a recording is to perform filtering in order to extract the frequencies of interest, and remove noise. Removing noise is an important step to extract meaningful information from our signal. There are various techniques for filtering, but frequencies of interest should also be done into consideration of likely frequencies associated with the task of interest.

2. Spike Detection: With a cleaned signal, we want to extract parts of the signal may be associated with a single neuron. Usually this signals are louder than the surrounding noise. With a strong clean signal, simple peak detection methods work fine. For more complicated signals, algorithms such as nonlinear energy operator (NEO) attempts to identify peaks using the energy of the signal.

3. Waveform Extraction: With peaks identified, the waveforms contributing to those peaks can then be identified to represent the shape of the neuronal spike at that time. Usually a timeframe before and after the highest peak point are used as features to represent the waveform.

4. Waveform Classification Using Dimensionality Reduction: With the waveform extracted, it is not an unsupervised clustering problem to identify what waveforms belong to which neurons. Waveforms can be projected into a lower dimensional space using PCA as a dimensionality reduction algorithm. Then various methods can be used to identify clusters. With new signals and new waveforms, the waveform can be projected into the PCA space and classified based upon the most likely cluster that the waveform belongs to. This is the basis of spike sorting.

Figure 3 gives a pictoral overview of the steps outlines above.

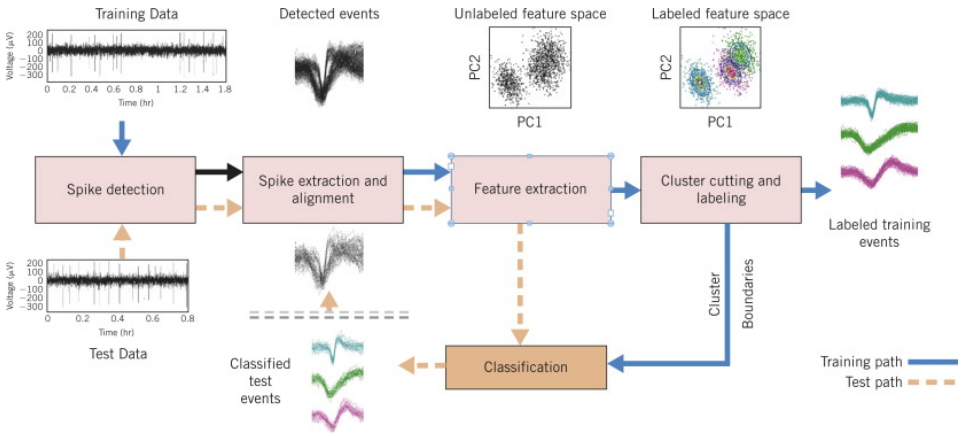


Figure 3: General Spike Sorting Pipeline.

4.4 Machine Learning Applications

Now that we understand basic neural signal processing, we can apply this to machine learning. We can record electrical activity from a resting brain and use spike sorting to identify individual neurons.

Now, we can extend that framework to record electrical activity with an event. Say we have a participant with an electrode on their brain. We can record neuron spiking with motor imagery, such as asking the participant to imagine moving their left or right hand. With these recordings, we can then implement machine learning models to identify how spiking can lead to classification of a left or right hand signal. That way, in the future, we can apply this model to new and unseen data to predict whether the person is intending to move their left or right hand!

The applications for BCI and machine learning are endless!