# Factor and Component Analysis

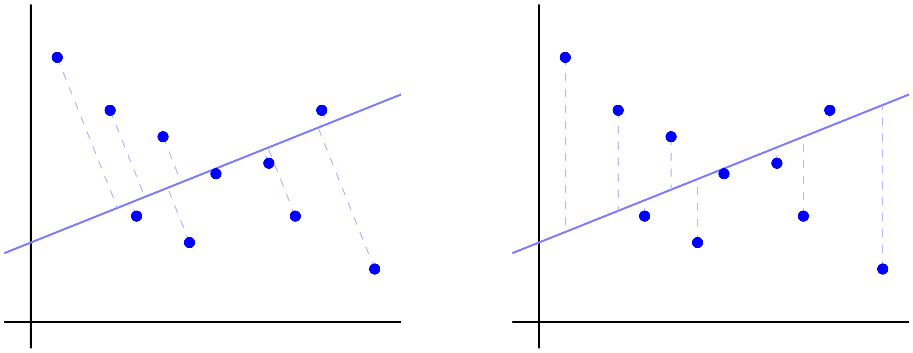
Muchang Bahng

Spring 2025

## Contents

Principal component analysis (PCA) and factor analysis (FA) originated independently by Pearson in 1901 and Spearman in 1904 [Pea01, Spe04]. Pearson gave the first formal treatment of it not to compute principal components, but to give a new measure of what a "best fit" line means. On the other hand, Spearman—frustrated by the lack of rigorous analyses on nontrivial in psychology—attempted to model the correlation between mental aptitude and sensory tasks. Though their discoveries were independent, the similarity of their models had inevitably caused their developments to coincide.

Note that PCA is similar to linear regression in that it fits some line (or hyperplane) of best fit to some data. However, linear regression—as a model that tries to use the covariates $x$ to predict the response $y$—attempts to minimize the *residual* $(y - \hat{y})^2$. If we were to flip the model and try to predict $x$ with $y$, then the best fit line would not be the same. As Pearson puts it, *the most probable stature of a man with a given length of leg $l$ being $s$, the most probable length of a leg for a man of stature $s$ will not be $l$* [Pea01]. This is further motivated by the fact that in many data collecting procedures, you do not collect a perfect measurement of $x$ first and then a noisy measurement of $y$. Rather, you are usually collecting both $x$ and $y$ together at the same time, in which they may both be perceptive to error.



(a) PCA minimizes the orthogonal distance to the subspace.

(b) Linear regression minimizes the residual distance to the subspace.

Figure 1: Note that this is in fact different from linear regression as it minimizes the expected *orthogonal distance* to the subspace, rather than the residual distance to the subspace as in linear regression.

# 1 Principal Component Analysis

Say that we have a random vector $x = (x_1, \ldots, x_d)$. These $d$ covariates will naturally be correlated, and we want to ask whether some more fundamental set of independent variables exist [Hot33] such that we can express

$$x = f(v_1, \ldots, v_k) \tag{1}$$

Naturally, we think of $f$ as a linear function.

We can think of PCA doing two things. First, it is a dimensionality-reduction algorithm where it takes samples $x \in \mathbb{R}^d$ and projects them into some smaller subspace $\mathcal{L}$ of dimension $k$. Second, it identifies an orthonormal basis of $\mathcal{L}$ that act as uncorrelated low-dimensional features. Because the projection map is linear and we are working in a lower-dimensional subspace, these new basis vectors are linear combinations of the original basis, which may reduce redundancy. Furthermore, by approximately modeling the original $x$ as a linear combination of these features, we are able to get a more parsimonious representation.

In PCA literature, it is more common to work with row vectors $x \in \mathbb{R}^{1 \times d}$, so linear mappings are realized through right matrix multiplication $xA$. Furthermore, we will assume that the data are 0-mean.

## 1.1 L2 Residual Minimization Approach

To give some motivation, we try to find a best fit line in $\mathbb{R}^d$. A line $\ell$ can be parameterized by a unit vector $v$ (note that it can be $\pm v$!), and so given some sample $x$, its projection onto $\ell$ is $\mathrm{proj}_\ell(x) = \langle x, v\rangle v$. Therefore, the residual is

$$\begin{aligned}
\|x - \langle x, v\rangle v\|^2 &= \|x\|^2 - 2\langle x, \langle x, v\rangle v\rangle + \|\langle x, v\rangle v\|^2 && (2)\\
&= \|x\|^2 - 2\langle x, v\rangle^2 + \langle x, v\rangle^2 \|v\|^2 && (3)\\
&= \|x\|^2 - \langle x, v\rangle^2 && (4)
\end{aligned}$$

since $\|v\|^2 = 1$ [Sha19]. Now given a 0-mean random variable $x$ (why we only consider 0-mean RVs will be clear later), our risk is

$$R(v) = \mathbb{E}_x\big[\|x - (x \cdot v)v\|\big] = \mathbb{E}_x\big[\|x\|^2\big] - \mathbb{E}_x\big[\langle x, v\rangle^2\big] \tag{5}$$

In practice, we want to minimize our empirical risk. Assume that we have sampled data $x^{(1)}, \ldots, x^{(n)} \sim x$. Then,

$$\begin{aligned}
\operatorname*{argmin}_{v \in \mathbb{R}^d, \|v\|=1} \hat{R}(v) &= \operatorname*{argmin}_{v \in \mathbb{R}^d, \|v\|=1} \frac{1}{n}\left(\sum_{i=1}^n \|x^{(i)}\|^2 - \sum_{i=1}^n \langle x^{(i)}, v\rangle^2\right) && (6)\\
&= \operatorname*{argmax}_{v \in \mathbb{R}^d, \|v\|=1} \frac{1}{n}\sum_{i=1}^n \langle x^{(i)}, v\rangle^2 && (7)
\end{aligned}$$

We have our loss function! Now what if we wanted to look for best fitting subspaces in general? Let's first rigorously define such a space.

**Definition 1.1 (Principal Subspace)**

Let $x$ be a 0-mean random variable in $\mathbb{R}^d$ and let $\mathcal{L}^k$ denote all $k$-dimensional linear subspaces of $\mathbb{R}^d$. The $k$**th principal subspace** is defined as

$$\ell_k = \operatorname*{argmin}_{\ell \in \mathcal{L}_k} \mathbb{E}_{\tilde{x}}\big[\|x - \mathrm{proj}_\ell\, x\|_2\big] \tag{8}$$

This isn't a big step from what we had before. We just want to construct a subspace $\ell$ that minimizes the expected $L^2$ distance between $x$ and $\ell$. Now how do we do such a thing? The most natural extension would be to identify an orthonormal basis $v_1, \ldots, v_k$, and since

$$\operatorname{proj}_\ell x = \sum_{i=1}^k \operatorname{proj}_{v_i} x \tag{9}$$

our loss can be simplified to

$$\begin{aligned}
R(\ell) = R(v_1, \ldots, v_k) &= \mathbb{E}\Big[\|x - \operatorname{proj}_\ell x\|^2\Big] && (10)\\
&= \mathbb{E}\Big[\|x\|^2 - 2\langle x, \sum_{i=1}^k \operatorname{proj}_{v_i} x\rangle + \|\sum_{i=1}^k \operatorname{proj}_{v_i} x\|^2\Big] && (11)\\
&= \mathbb{E}\Big[\|x\|^2 - 2\sum_{i=1}^k \langle x, \operatorname{proj}_{v_i} x\rangle + \sum_{i=1}^k \|\operatorname{proj}_{v_i} x\|^2\Big] && (12)\\
&= \mathbb{E}\Big[\|x\|^2 - 2\sum_{i=1}^k \langle x, v_i\rangle^2 + \sum_{i=1}^k \langle x, v_i\rangle^2 \|v_i\|^2\Big] && (13)\\
&= \mathbb{E}\Big[\|x\|^2 - \sum_{i=1}^k \langle x, v_i\rangle^2\Big] && (14)
\end{aligned}$$

Now if $x$ was not 0-mean, our intuition would tell us that the principal subspace should pass through its mean, or centroid. In fact [Pea01] showed this for a 1-dimensional subspace.

**Lemma 1.1 (Principal Subspace Must Intersect Mean)**

Assume that $\mathbb{E}[x] \neq 0$, and by abuse of notation let us denote $\ell + p \in \mathbb{R}^d$ as the affine subspace $\ell$ that goes through $p$ orthogonally from the origin.

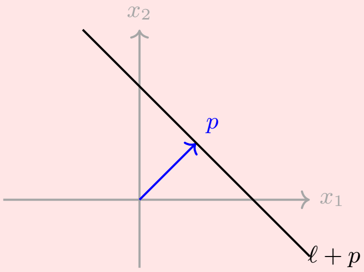


Figure 2: By orthogonal intersection, we mean that for any $w \in \ell$, $\langle w, p\rangle = 0$

Then, we claim that the value of $p$ that minimizes the risk is the average projection distance to the subspace.

$$p = \mathbb{E}[x - \operatorname{proj}_\ell(x)] \tag{15}$$

**Proof.**

The projection distance of $x$ onto $\ell + p$ is the same as the projection distance from $x - p$ onto $\ell$.

Therefore, noting that $v_i$'s are orthogonal, we can derive

$$\begin{aligned} R(\ell, p) &= \mathbb{E}_x \big[ \|(x-p) - \mathrm{proj}_\ell(x)\|^2 \big] && (16) \\ &= \mathbb{E}_x \bigg[ \bigg\| (x-p) - \sum_{i=1}^k \mathrm{proj}_{v_i}(x) \bigg\|^2 \bigg] && (17) \\ &= \mathbb{E}_x \bigg[ \|x-p\|^2 + \bigg\| \sum_{i=1}^k \mathrm{proj}_{v_i}(x) \bigg\|^2 - 2 \Big\langle x-p, \sum_{i=1}^k \mathrm{proj}_{v_i}(x) \Big\rangle \bigg] && (18) \\ &= \mathbb{E}_x \bigg[ \|x-p\|^2 - \sum_{i=1}^k \langle x, v_i\rangle^2 + 2 \sum_{i=1}^k \langle x, v_i\rangle \langle p, v_i\rangle \bigg] && (19) \end{aligned}$$

To find the minimum, we take the total derivative and set it equal to 0. We are taking the derivative w.r.t. $p$ of an integral w.r.t. $x$, so we can push the derivative into the integral and solve.

$$\begin{aligned} 0 = \frac{\partial R(\ell, p)}{\partial p} &= \mathbb{E}_x \left[ \frac{\partial}{\partial p} \left\{ \|x-p\|^2 - \sum_{i=1}^k \langle x, v_i\rangle^2 + 2 \sum_{i=1}^k \langle x, v_i\rangle \langle p, v_i\rangle \right\} \right] && (20) \\ &= \mathbb{E}_x \left[ 2(p-x) + 2 \sum_{i=1}^k \langle x, v_i\rangle \frac{\partial}{\partial p} \langle p, v_i\rangle \right] && (21) \\ &= \mathbb{E}_x \left[ 2(p-x) + 2 \sum_{i=1}^k \langle x, v_i\rangle v_i \right] && (22) \\ &= \mathbb{E}_x \left[ 2(p-x) + 2\,\mathrm{proj}_\ell(x) \right] && (23) \end{aligned}$$

and by solving for $p$, we get

$$p = \mathbb{E}_x[x - \mathrm{proj}_\ell(x)] \tag{24}$$

Therefore, we can just normalize the data to 0 and simply use 14 without having to account for the affine translation $p$. In the empirical case, we can get rid of the fixed $x$ and find

$$\operatorname*{argmax}_{v_i \in \mathbb{R}^d} \frac{1}{n} \sum_{j=1}^n \sum_{i=1}^k \langle x^{(j)}, v_i\rangle^2, \quad \text{subject to } \|v_i\|^2 = 1, \langle v_i, v_j\rangle = 0 \text{ for } i \neq j \tag{25}$$

By stacking the $v_i$'s left-to-right in matrix $V_k \in \mathbb{R}^{d \times k}$, we can get a cleaner form of the loss function.

$$\begin{aligned} \frac{1}{n} \sum_{j=1}^n \sum_{i=1}^k \langle x^{(j)}, v_i\rangle^2 &= \frac{1}{n} \sum_{j=1}^n \sum_{i=1}^k (x^{(j)})^T v_i v_i^T x^{(j)} && (26) \\ &= \frac{1}{n} \sum_{j=1}^n (x^{(j)})^T V_k V_k^T x^{(j)} && (27) \\ &= \frac{1}{n} \mathrm{Tr}(X V_k V_k^T X^T) = \frac{1}{n} \mathrm{Tr}(V_k^T X^T X V_k) && (28) \end{aligned}$$

This leads to an intuitive loss function for PCA.

**Theorem 1.1 (Constrained Empirical Risk of $k$th Principal Subspace)**

The empirical risk, or loss function, of PCA is

$$\operatorname*{argmax}_{V \in \mathbb{R}^{d \times k}, V_k^T V_k = I_k} \frac{1}{n} \|X - X V_k V_k^T\|^2 \tag{29}$$

**Proof.**

By the Frobenius norm expansion and since $V_k^T V_k = I_k$, we have

$$\begin{aligned}
\|X - XV_kV_k^T\|_F^2 &= \mathrm{Tr}((X - XV_kV_k^T)^T(X - XV_kV_k^T)) && (30)\\
&= \mathrm{Tr}(X^TX - X^TXV_kV_k^T - V_kV_k^TX^TX + V_kV_k^TX^TXV_kV_k^T) && (31)\\
&= \mathrm{Tr}(X^TX) - 2\,\mathrm{Tr}(X^TXV_kV_k^T) + \mathrm{Tr}(V_kV_k^TX^TXV_kV_k^T) && (32)\\
&= \mathrm{Tr}(X^TX) - 2\,\mathrm{Tr}(V_k^TX^TXV_k) + \mathrm{Tr}(V_k^TV_kV_k^TX^TXV_k) && (33)\\
&= \mathrm{Tr}(X^TX) - 2\,\mathrm{Tr}(V_k^TX^TXV_k) + \mathrm{Tr}(V_k^TX^TXV_k) && (34)\\
&= \mathrm{Tr}(X^TX) - \mathrm{Tr}(V_k^TX^TXV_k) && (35)
\end{aligned}$$

and since the empirical risk does not depend on $X$, minimizing the Frobenius norm is equivalent to maximizing the second trace term, i.e. 28.

**Definition 1.2 (Projection Operator)**

Note that there are two distinct projection operators, which are realized through right matrix multiplication.

1. The linear map $V_k : \mathbb{R}^d \to \mathbb{R}^k$ is a projection operator of the samples $x$ into the component space.
2. The linear map $V_kV_k^T : \mathbb{R}^d \to \mathbb{R}^d$ is called the **rank-k projection operator** onto the $k$th principal subspace.

Therefore, $XV_k \in \mathbb{R}^{n\times k}$ is the projection of the dataset into the component space. If we want to get the denoised samples in the sample space $\mathbb{R}^d$, we project it back out $XV_kV_k^T$.

## 1.2 Variance Maximization Approach

But we can turn this into a variance maximization problem. Note that $\mathrm{Var}_x[\langle x, v\rangle] = \mathbb{E}_x[\langle x, v\rangle^2] - \mathbb{E}_x[\langle x, v\rangle]^2$, and so we can rewrite our true risk as

$$\underset{v\in\mathbb{R}^d, \|v\|=1}{\mathrm{argmin}}\; R(v) = \underset{v\in\mathbb{R}^d, \|v\|=1}{\mathrm{argmin}}\; \mathbb{E}_x\left[\|x\|^2\right] - \mathrm{Var}_x[\langle x, v\rangle] - \mathbb{E}_x[\langle x, v\rangle]^2 \tag{36}$$

where the last term vanishes since $x$ is 0-mean, and hence by linearity of expectation $\mathbb{E}_x[\langle x, v\rangle] = \langle \mathbb{E}[x], v\rangle = \langle 0, v\rangle = 0$. In parallel the empirical risk reduces to simply the sample variance.

$$\underset{v\in\mathbb{R}^d, \|v\|=1}{\mathrm{argmax}}\; \hat{\mathrm{Var}}[\langle x, v\rangle] = \underset{v\in\mathbb{R}^d, \|v\|=1}{\mathrm{argmax}}\; \frac{1}{n}\left(\sum_{i=1}^n \langle x^{(i)}, v\rangle^2\right) \tag{37}$$

Therefore, we can think of the $L^2$ minimization problem as equivalent to a variance maximization approach.

**Lemma 1.2 (Variance Maximization Approach)**

Minimizing the $L^2$ distance of a random variable $x$ to a line $\ell$ in $\mathbb{R}^d$ is equivalent to maximizing the scalar variance in the projected space.

$$\underset{v\in\mathbb{R}^d, \|v\|=1}{\mathrm{argmin}}\; \mathbb{E}[\|x - \mathrm{proj}_v(x)\|_2] = \underset{v\in\mathbb{R}^d, \|v\|=1}{\mathrm{argmax}}\; \mathrm{Var}_x[\langle x, v\rangle] \tag{38}$$

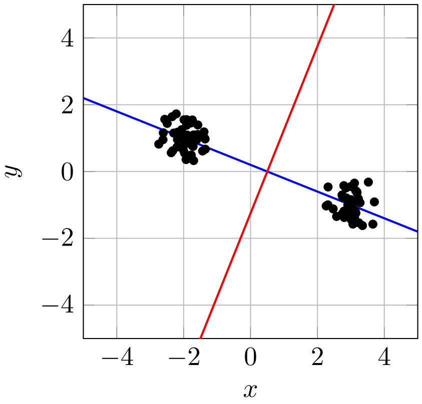


Figure 3: Projecting the dataset onto the blue line seems to retain more variance than projecting onto the red line.

Let's in fact try to directly maximize the variance. If we vertically stack our $n$ data points into a matrix $X \in \mathbb{R}^{n \times d}$, then the projections of this data onto $\mathbb{R}$ is simply $Xv \in \mathbb{R}^n$. Again, since this is 0-mean, the variance is

$$\begin{aligned} \hat{\mathrm{Var}}(Xv) &= \frac{1}{n}(Xv)^T(Xv) && (39)\\ &= \frac{1}{n} v^T X^T X v && (40)\\ &= v^T \frac{X^T X}{n} v && (41)\\ &= v^T \hat{\Sigma} v && (42) \end{aligned}$$

where $\hat{\Sigma}$ is the empirical covariance matrix of $X$. We want to find

$$\max_v v^T \hat{\Sigma} v \text{ subject to } \|v\|^2 = 1 \tag{43}$$

This is a classic Lagrange multiplier problem. We construct the Lagrangian and compute its partial derivatives to set equal to 0.

$$\begin{aligned} \mathcal{L}(v, \lambda) &= v^T \hat{\Sigma} v - \lambda(\|v\|^2 - 1) && (44)\\ \frac{\partial \mathcal{L}}{\partial v} &= 2\hat{\Sigma} v - 2\lambda v = 0 && (45)\\ \frac{\partial \mathcal{L}}{\partial \lambda} &= v^T v - 1 = 0 && (46) \end{aligned}$$

which gives us

$$\hat{\Sigma} v = \lambda v, \qquad v^T v = 1 \tag{47}$$

This tells us that $v$ is a unit eigenvector, and the maximizing vector will be the one corresponding to the largest eigenvalue. Essentially, we have reduced this to an eigenvalue problem.

**Theorem 1.2 (1st Principal Subspace as Eigenvector)**

The first principal subspace of data matrix $X \in \mathbb{R}^{n \times d}$ is spanned by the eigenvector corresponding to the largest eigenvalue of the sample covariance matrix $\hat{\Sigma} = \frac{1}{n} X^T X$.

Now for higher dimensional subspaces, we take the same approach. Going through the same derivation gives the expected risk in terms of the variance

$$R(v_1, \ldots, v_k) = \mathbb{E}[\|x\|^2] - \sum_{i=1}^k \mathbb{E}[\langle x, v_i \rangle^2] \tag{48}$$

$$= \mathbb{E}[\|x\|^2] - \sum_{i=1}^k \mathrm{Var}[\langle x, v_i \rangle] - \mathbb{E}[\langle x, v_i \rangle]^2 \tag{49}$$

By fixing the $x$'s and going through the same derivation as 39, we get our equivalent empirical risk.

$$\underset{v_i \in \mathbb{R}^d}{\mathrm{argmax}} \sum_{i=1}^k \hat{\mathrm{Var}}[\langle x, v_i \rangle] = \underset{v_i \in \mathbb{R}^d}{\mathrm{argmax}} \sum_{i=1}^k v_i^T \hat{\Sigma} v_i \tag{50}$$

**Theorem 1.3 (Constrained Empirical Risk of $k$th Principal Subspace)**

The empirical risk tells us to find an orthonormal basis that maximizes the sum of the variance of projections.

$$\underset{v_i \in \mathbb{R}^d}{\mathrm{argmax}} \sum_{i=1}^k v_i^T \hat{\Sigma} v_i \text{ subject to } \|v_i\|^2 = 1, \langle v_i, v_j \rangle = 0 \text{ for } i \neq j \tag{51}$$

The variance-maximization loss is very insightful, and we may naively think of just taking the unit eigenvectors corresponding to the top $k$ largest eigenvalues. Surprisingly, this greedy approach turns out to be correct.

Let's derive this further

$$\sum_{i=1}^k v_i^T \hat{\Sigma} v_i = \sum_{i=1}^k \mathrm{Tr}(v_i v_i^T \hat{\Sigma}) \tag{52}$$

$$= \mathrm{Tr}(V_k V_k^T \hat{\Sigma}) \tag{53}$$

$$= \mathrm{Tr}(\hat{\Sigma} V_k V_k^T) \tag{54}$$

which again is equal to 28. By the spectral theorem, we can take the eigendecomposition of self-adjoint $\hat{\Sigma} = Q \Lambda Q^T$ with orthogonal matrix $Q$. Setting $W_k = Q^T V_k$, we have

$$\mathrm{Tr}(\hat{\Sigma} V_k V_k^T) = \mathrm{Tr}(Q \Lambda Q^T V_k V_k^T) = \mathrm{Tr}(\Lambda Q^T V_k V_k^T Q) = \mathrm{Tr}(\Lambda W_k W_k^T) = \sum_{i=1}^d \lambda_i (W_k W_k^T)_{ii} \tag{55}$$

where without loss of generality we have $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$. However, we have two constraints. First, since $W_k W_k^T$ is a projection matrix, the eigenvalues must be between 0 and 1. Second, by the cyclic trace property, we have $\mathrm{Tr}(W_k W_k^T) = \mathrm{Tr}(I_k) = k$.[1] So denoting $w_i = (W_k W_k^T)_{ii}$, we have the optimal allocation problem

$$\max \sum_{i=1}^d \lambda_i w_i \text{ subject to } \begin{cases} w_i \in [0, 1] \; \forall i = 1, \ldots, d \\ \sum_i w_i = k \end{cases} \tag{56}$$

Since the eigenvalues are decreasing, it doesn't take too much to see that the optimal solution is to just put everything you have into the largest eigenvalues. So we fill the first $w_1 = \ldots = w_k = 1$ and the rest $w_{k+1}, \ldots w_d = 0$. Therefore, this solution corresponds to $W_k = (e_1, e_2, \ldots, e_k)$, and so

$$V_k = Q W_k = Q_k \tag{57}$$

which is the truncated matrix $Q$ containing the first $k$ eigenvectors of $\hat{\Sigma}$ corresponding to the largest eigenvalues. At this point, it does not suffice to talk about just a principal subspace anymore. We must identify its orthonormal basis, i.e. the eigenvectors.

[1] Though it is *not* the case that $W_k W_k^T = I_k$!

### Definition 1.3 (Principal Axis)

The eigenvectors $v_1, \ldots, v_k$ that span the $k$th principal subspace are called the top $k$ **principal axes**, or **principal directions**.[a]

[a] The terminology is misused and confusing sometimes. See https://stats.stackexchange.com/questions/88118/what-exactly-is-called-principal-component-in-pca.

### Definition 1.4 (Principal Scores)

Given a sample $x \in \mathbb{R}^d$, its top $k$ **principal scores** are defined in the equivalent ways.

1. It is the components of $V_k^T x$. That is, we take a sample and project it onto the component space.
$$x = \sum_i a_i e_i \mapsto V_k^T x = \sum_j b_j e_j' \tag{58}$$
where $e_i, e_j'$ are the basis vectors in $\mathbb{R}^d, \mathbb{R}^k$. Then $b_1, \ldots, b_k$ are the principal scores of $x$.
2. It is the coefficients of $x$ with respect to the basis spanned by the top $k$ eigenvalues.
$$x = c_1 v_1 + c_2 v_2 + \ldots + c_k v_k + c_{k+1} v_{k+1} + \ldots + c_d v_d \tag{59}$$
Then $c_1, \ldots, c_k$ are the principal scores.

**Proof.**

These two are clearly equivalent since

$$b_1 e_1' + \ldots + b_k e_k' = V_k^T x = V_k^T \left( \sum_{i=1}^d c_i v_i \right) = \sum_{i=1}^d c_i V_k^T v_i = \sum_{i=1}^k c_i e_i' \tag{60}$$

which is of the proper form when $c_i = b_i$. Since $V_k^T$ is a truncated orthogonal matrix, $V_k^T v_i = e_i'$ for $1 \leq i \leq k$ and 0 for all else.

This is similar to taking the first principal component $v_1$ on $X$, and then by computing the first principal component on the remaining residuals $X - \mathrm{proj}_{v_1} X$, we get the second principal component, which is guaranteed to be orthogonal. But usually, we end up just computing all eigenvectors at once.

Now how do we know that this sample decomposition is a good approximation to the true decomposition? It comes from the fact that the sample covariance $\hat{\Sigma}$ is a good approximation of the true covariance $\Sigma$, which we will later prove using concentration of measure.

### Theorem 1.4 (Risk)

The risk satisfies

$$R(k) = \mathbb{E}[\|x - V_k V_k^T x\|^2] = \sum_{j=k+1}^{D} \lambda_j \tag{61}$$

It is essential that you plot the spectrum in decreasing order. This allows you to analyze how well PCA is working. People often use the "elbow" technique to determine where to choose $K$, and we value

$$\frac{\sum_{j=1}^k \lambda_j}{\sum_{j=1}^d \lambda_j} \tag{62}$$

accounts for the **variance explained**, which should be high with $K$ low. If you have to go out to dimension $K = 50$ to explain 90% of the variance, then PCA is not working. It may not work because of many reasons, such as there being nonlinear structure within the data.

## 1.3 Decomposition Solvers

So far, the equivalence of the L2 minimization approach and variance maximization approach had been firmly established by 1933 in [Hot33]. However, the person who formalized the connection in using low-rank approximations in PCA was Eckart and Young in 1936 [EY36].[2]

**Theorem 1.5 (Eckart-Young Theorem)**

The solution to

$$\min_{A_k \in \mathbb{R}^{m \times n}} \|A - A_k\|_F, \qquad \text{subject to } \mathrm{rank}(A_k) \leq k \tag{63}$$

is given by the matrix $A_k = USV_k^T$, where $A = USV^T$ is the singular value decomposition of $A$, and $V_k$ is the truncated matrix formed from the first $k$ columns of $V$. Note that $\|\cdot\|_F$ is the Frobenius norm.

**Proof.**

Let $A \in \mathbb{R}^{m \times n}$ be a real (possibly rectangular) matrix with $m \leq n$. Suppose that

$$A = U\Sigma V^T \tag{64}$$

is the singular value decomposition of $A$.
We claim that the best rank $k$ approximation to $A$ in the Frobenius norm, denoted by $\|\cdot\|_F$, is given by

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T \tag{65}$$

where $u_i$ and $v_i$ denote the $i$th column of $U$ and $V$, respectively.
First, note that we have

$$\|A - A_k\|_F^2 = \left\| \sum_{i=k+1}^{n} \sigma_i u_i v_i^T \right\|_F^2 = \sum_{i=k+1}^{n} \sigma_i^2 \tag{66}$$

Therefore, we need to show that if $B_k = XY^T$ where $X$ and $Y$ have $k$ columns then

$$\|A - A_k\|_F^2 = \sum_{i=k+1}^{n} \sigma_i^2 \leq \|A - B_k\|_F^2. \tag{67}$$

By the triangle inequality with the spectral norm, if $A = A' + A''$ then $\sigma_1(A) \leq \sigma_1(A') + \sigma_1(A'')$. Suppose $A'_k$ and $A''_k$ respectively denote the rank $k$ approximation to $A'$ and $A''$ by SVD method described above. Then, for any $i, j \geq 1$

$$\begin{aligned}
\sigma_i(A') + \sigma_j(A'') &= \sigma_1(A' - A'_{i-1}) + \sigma_1(A'' - A''_{j-1}) && (68)\\
&\geq \sigma_1(A - A'_{i-1} - A''_{j-1}) && (69)\\
&\geq \sigma_1(A - A_{i+j-2}) \quad (\text{since } \mathrm{rank}(A'_{i-1} + A''_{j-1}) \leq i + j - 2) && (70)\\
&= \sigma_{i+j-1}(A). && (71)
\end{aligned}$$

Since $\sigma_{k+1}(B_k) = 0$, when $A' = A - B_k$ and $A'' = B_k$ we conclude that for $i \geq 1, j = k+1$

$$\sigma_i(A - B_k) \geq \sigma_{k+i}(A). \tag{72}$$

[2] In fact, a stronger version was proved before, but they rediscovered it.

Therefore,

$$\|A - B_k\|_F^2 = \sum_{i=1}^n \sigma_i(A - B_k)^2 \geq \sum_{i=k+1}^n \sigma_i(A)^2 = \|A - A_k\|_F^2, \tag{73}$$

as required.

Now this gives the theoretical justificiation to use SVD. Since $\frac{1}{n} X^T X = \frac{1}{n} V S U^T U \sigma V^T = \frac{1}{n} V S^2 V^T$, the columns of $V$ are the principal axes. Recall from linear algebra that $\Lambda = S^2$.

**Algorithm 1.1 (PCA with SVD)**

Given a dataset $X \in \mathbb{R}^{n \times d}$, let us denote the rows as $x_i$, and say that we are looking for a subspace of dimension $k$.

1. Compute the mean
$$\mu = \frac{1}{n} \sum_{i=1}^n x_i \in \mathbb{R}^d \tag{74}$$
2. Standardize the data $\tilde{X} = X - \mu$, i.e. $\tilde{x}_i = x_i - \mu$.
3. Compute the SVD $\tilde{X} = USV^T$.
4. Compute the submatrices $V_k \in \mathbb{R}^{k \times k}$ and $S_k \in \mathbb{R}^{D \times k}$.
5. Define the projection operator $P_k(x) = \mu + \sum_{j=1}^k \langle x - \mu, v_j \rangle \, v_j$, the change of basis operator $T$, and the embedding operator $T^{-1}(z) = \mu + V_k S_k z$.

A demonstration is done here.

**Example 1.1 (Walkthrough)**

Say that we have some dataset of 100 points in $\mathbb{R}^3$. The data matrix is shown on the right, but in reality I just generated a toy dataset.

```
def scatter(n=1000):
  X_2d = np.random.multivariate_normal(
    np.zeros(2), np.eye(2), n)
  A = np.array([[1, 1, 1], [-2, 2, 1]])
  X_3d = X_2d @ A +
    np.random.multivariate_normal(
    np.zeros(3), np.eye(3), n)
  return X_3d
```

```
[[ 8.864e-01  3.975e-01  7.009e-01]
 [-2.065e+00  3.258e+00  1.874e+00]
 [ 3.970e-01 -5.400e-01 -3.054e-01]
 [ 3.239e+00 -1.999e+00 -1.034e+00]
 ...
 [-1.295e-01  9.683e-01  2.861e-01]
 [-7.097e-01 -4.060e-01 -1.058e+00]
 [ 2.284e+00 -2.505e+00 -1.522e+00]]
```

We can take the SVD, which will give us

```
In [7]: U, S, Vt = np.linalg.svd(X)

In [8]: print(U)
[[ 4.804e-04  7.852e-02  4.071e-02 ...  5.320e-03  5.034e-02 -9.305e-02]
 [ 1.420e-01  3.992e-02 -1.229e-02 ... -5.512e-02  4.460e-02  7.192e-02]
 [-2.456e-02 -3.657e-03 -4.578e-04 ... -1.005e-01 -1.512e-01 -7.300e-02]
 ...
 [ 2.909e-02  2.754e-02 -1.075e-01 ...  9.871e-01 -1.325e-02 -3.559e-03]
 [-8.714e-03 -8.159e-02 -1.436e-01 ... -1.297e-02  9.733e-01 -1.055e-02]
 [-1.241e-01  2.077e-03 -6.028e-02 ... -3.435e-03 -8.286e-03  9.819e-01]]

In [9]: print(S)
[29.90178039 15.17454164  3.01267412]

In [10]: print(Vt.T)
```

```
[[-0.59855644  0.77282378 -0.21088764]
 [ 0.70716875  0.38606989 -0.59233639]
 [ 0.37635428  0.50367991  0.77760144]]
```

The `Vt.T` represents $V$, with its columns being our principal axes, and we wish to plot them along with our data points. We would like to scale them by their variance captured.

1. Take the singular values $(\sigma_1, \sigma_2, \sigma_3) = (29.9, 15.2, 3.0)$.
2. Square them to get the eigenvalues $(\lambda_1, \lambda_2, \lambda_3) = (894, 230, 9)$.
3. Normalize them to get the percent variance captured. $(0.789, 0.203, 0.008)$, and use this as a scale for each eigenvector.

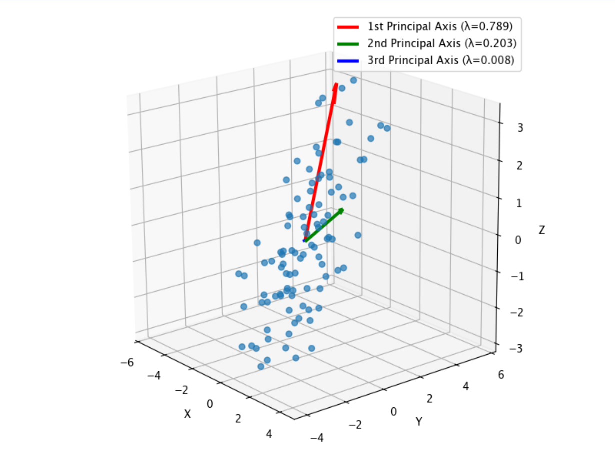


Figure 4: We can see that the data approximately lies on a 2-dimensional subspace of $\mathbb{R}^3$.

### Example 1.2 (Eigenfaces)

In 1991, Turk and Pentland presented an eigenface method of face recognition by taking the low-rank approximation of a dataset of face images [TP91].

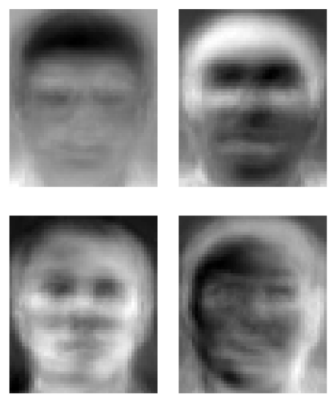


Figure 5: Some eigenfaces from AT&T Labs.

The first $k$ principal components provide not only the best rank $k$ approximation to $X$, but also the covariance matrix $\hat{\Sigma}$.

**Theorem 1.6 (PCA is an Unbiased Estimator of Sample Variance)**

The truncated covariance matrix $\Sigma^k = V_k S^2 V_k^T$ is an unbiased estimator of $X^T X$.

**Proof.**

Indeed, $\hat{\Sigma} = X^T X = V S^2 V^T$, and the last equation is the SVD decomposition of $\hat{\Sigma}$. So again using Eckart-Young theorem, the best rank $k$ approximation to $\hat{\Sigma}$ is given by $\Sigma^k = V_k S^2 V_k^T$ (now truncated both in rows and columns).

## 1.4 Iterative Solvers

A disadvantage of decomposition is that the time complexity of SVD on a $n \times m$ matrix is $O(nm \min\{n, m\})$. Therefore, we use iterative methods, which is really just applications of numerical linear algebra.

### 1.4.1 Power Method

The simplest iterative eigenvalue solver was due to von Mises in 1929 [MP29]. Intuitively, given a diagonalizable matrix that we want to solve for, we can "normalize" the spectrum by dividing all the eigenvectors by $\lambda_1$, the largest eigenvalue. Then by composing these linear maps, all the other eigenvectors should die out while keeping the largest eigenvector in place.

**Theorem 1.7 (Convergence of Power Iteration)**

Let $A \in \mathbb{R}^{n \times n}$ be a diagonalizable matrix and $x_0 \in \mathbb{R}^n$ any random vector. Given the two assumptions:

1. $A$ has a unique greatest eigenvalue $\lambda_1$.
2. $x_0$ has a nonzero component in the direction of the eigenvalue $v_1$ associated with $\lambda_1$.

Then, the sequence $(x_t)$ by defined recursively as

$$x_{t+1} = \frac{A x_t}{\|A x_t\|} \tag{75}$$

converges to $v_1$.

**Proof.**

We rewrite the recurrence relation as

$$x_{t+1} = \frac{A x_t}{\|A x_t\|} = \frac{A^{t+1} x_0}{\|A^{t+1} x_0\|} \tag{76}$$

Since $A$ is diagonalizable, we can write $A = V S V^{-1}$ where $V$ is the matrix of eigenvectors and $S$ is the diagonal matrix of eigenvalues. Thus:

$$x_t = \frac{A^t x_0}{\|A^t x_0\|} = \frac{(V S V^{-1})^t x_0}{\|(V S V^{-1})^t x_0\|} = \frac{V S^t V^{-1} x_0}{\|V S^t V^{-1} x_0\|} \tag{77}$$

Since $x_0$ can be expressed as a linear combination of eigenvectors, we write $V^{-1} x_0 = c_1 e_1 + c_2 e_2 + \cdots + c_n e_n$ where $e_i$ are the standard basis vectors and $c_i$ are the coefficients. Therefore:

$$\begin{aligned} x_t &= \frac{V S^t V^{-1} (c_1 v_1 + c_2 v_2 + \cdots + c_n v_n)}{\|V S^t V^{-1} (c_1 v_1 + c_2 v_2 + \cdots + c_n v_n)\|} && (78) \\ &= \frac{V S^t (c_1 e_1 + c_2 e_2 + \cdots + c_n e_n)}{\|V S^t (c_1 e_1 + c_2 e_2 + \cdots + c_n e_n)\|} && (79) \end{aligned}$$

Factoring out the dominant eigenvalue $\lambda_1$:

$$x_t = \left(\frac{\lambda_1}{|\lambda_1|}\right)^t \frac{c_1}{|c_1|} \frac{v_1 + \frac{1}{c_1} V \left(\frac{1}{\lambda_1} S\right)^t (c_2 e_2 + \cdots + c_n e_n)}{\left\| v_1 + \frac{1}{c_1} V \left(\frac{1}{\lambda_1} S\right)^t (c_2 e_2 + \cdots + c_n e_n) \right\|} \tag{80}$$

Since $|\lambda_i / \lambda_1| < 1$ for $i > 1$, we have $\left(\frac{\lambda_i}{\lambda_1}\right)^t \to 0$ as $t \to \infty$. Therefore:

$$\frac{1}{c_1} V \left(\frac{1}{\lambda_1} S\right)^t (c_2 e_2 + \cdots + c_n e_n) \to 0 \text{ as } t \to \infty \tag{81}$$

Thus, as $t \to \infty$:

$$x_t \to \left(\frac{\lambda_1}{|\lambda_1|}\right)^t \frac{c_1}{|c_1|} \frac{v_1}{\|v_1\|} = \pm v_1 \tag{82}$$

Since we assumed that $c_1 \neq 0$, the sequence $(x_t)$ converges to $\pm v_1$.

Therefore, a direct consequence of this theorem is.

**Algorithm 1.2 (Power Iteration to Solve Largest Eigenvector)**

Let $A \in \mathbb{R}^{n \times m}$. Then

**Algorithm 1** Power Iteration Method

```
procedure PowerIteration($A$)
Require: $A \in \mathbb{R}^{n \times n}$ is a matrix, $max\_iter > 0$ is maximum iterations
    $x \leftarrow$ random vector of unit norm
    $\lambda \leftarrow 0$
    for $t \leftarrow 1$ to $max\_iter$ do
        $y \leftarrow Ax$
        $\lambda \leftarrow x^T y$
        $x \leftarrow \frac{y}{\|y\|}$
    end for
    return $\lambda, x$
end procedure
```

**Corollary 1.1 (Rate of Convergence of Power Iteration)**

The rate of convergence is

$$\frac{\lambda_2}{\lambda_1} \tag{83}$$

### 1.4.2 Lanczos Algorithms

The time complexity and the slow convergence of the power iteration led to faster algorithms, with one notable class being *Lanczos algorithms*, first developed in 1950 [Lan50].

**Definition 1.5 (Krylov Subspace)**

Given linear map $A : V \to V$ and vector $v \in V$, the $m$**th Krylov subspace** is

$$\mathcal{K}_m(A, v) = \text{span}\{v, Av, A^2 v, \ldots, A^{m-1} v\} \tag{84}$$

**Algorithm 1.3 (Lanczos Algorithm)**

The basic idea is to construct an orthonormal basis for the Krylov subspace $\mathcal{K}_m(A, v)$ using a three term recurrence relation. This process transforms the original matrix $A$ into a tridiagonal matrix $T$ that can be used to compute the eigenvectors efficiently.

**Algorithm 2** Lanczos Algorithm

```
procedure LANCZOS($A$, $m$)
Require: Hermitian matrix $A$ of size $n \times n$, number of iterations $m$ (default: $m = n$)
Ensure: $n \times m$ matrix $V$ with orthonormal columns and tridiagonal real symmetric matrix $T = V^* A V$ of size $m \times m$
    Let $v_1 \in \mathbb{C}^n$ be an arbitrary vector with Euclidean norm 1
    $w_1' \leftarrow A v_1$
    $\alpha_1 \leftarrow w_1'^* v_1$
    $w_1 \leftarrow w_1' - \alpha_1 v_1$
    for $j \leftarrow 2$ to $m$ do
        $\beta_j \leftarrow \|w_{j-1}\|$ (Euclidean norm)
        if $\beta_j \neq 0$ then
            $v_j \leftarrow w_{j-1}/\beta_j$
        else
            Pick $v_j$ as arbitrary vector with Euclidean norm 1 orthogonal to all of $v_1, \ldots, v_{j-1}$
        end if
        $w_j' \leftarrow A v_j - \beta_j v_{j-1}$
        $\alpha_j \leftarrow w_j'^* v_j$
        $w_j \leftarrow w_j' - \alpha_j v_j$
    end for
    Let $V$ be the matrix with columns $v_1, \ldots, v_m$ and
```

$$T = \begin{pmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \alpha_3 & \ddots & & \\ & & \ddots & \ddots & \beta_{m-1} & \\ & & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & & \beta_m & \alpha_m \end{pmatrix} \tag{85}$$

```
    return $V, T$
end procedure
```

For the runtime,

1. Each iteration consists of an $O(nm)$ matrix multiplication with the rest of the operations $O(n)$.
2. If the matrix is sparse, then this is of order $O(nk)$, where $k$ is the average number of nonzero elements in each row. If $k$ is bounded, then we can compute each iteration in linear time $O(n)$.

Now that we have $V, T$, what do we do? The following theorem tells us how to compute the original eigenvectors.

**Theorem 1.8 ()**

If $\lambda$ is an eigenvalue of $T$ and $x$ is its eigenvector (i.e., $Tx = \lambda x$), then $y = Vx$ is a corresponding eigenvector of $A$ with the same eigenvalue.

**Proof.**

Let $\lambda$ be an eigenvalue of $T$ with eigenvector $x$, so that $Tx = \lambda x$. We need to show that $y = Vx$ is an eigenvector of $A$ with eigenvalue $\lambda$.
Since the Lanczos algorithm produces the relation $T = V^* AV$, we have $A = VTV^*$. Therefore:

$$\begin{aligned} Ay &= AVx && (86) \\ &= VTV^*Vx && (87) \\ &= VTIx && (88) \\ &= VTx && (89) \\ &= V(\lambda x) && (90) \\ &= \lambda Vx && (91) \\ &= \lambda y && (92) \end{aligned}$$

One note of warning. This algorithm tends to be numerically unstable and sometimes there is a loss of orthogonality among the Lanczos vectors due to finite precision arithmetic. We can however just reorthogonalize them in the end.

### 1.4.3 LOBPCG

### 1.4.4 NIPALS

### 1.4.5 Oja's Neuron

With neural networks on the rise, Oja developed a simple neuron function that can compute the greatest eigenvector [Oja82]. Let's start with an extremely simple model of a neuron.

**Definition 1.6 (Neuron)**

Consider a linear model of a neuron that returns a sum of its inputs according to weighted presynaptic weights $y(x) = w^T x$.

**Definition 1.7 (Hebb's Rule)**

In neuroscience, **Hebb's rule** (aka Hebbian learning) colloquially states that neurons are activated together have greater strengths, like how muscles grow when you work them more.[a]

$$w_{t+1} - w_t = \eta y(x_t) x_t \tag{93}$$

[a] Neurons that fire together, wire together.

Since the weights are not constrained at all, we would find that the weights approach infinity if $\eta > 0$. Therefore, Oja resticted the weights so that the $p$-norm is 1 , $0 \leq w_i \leq 1$, which models a form of competition for resources between the neurons [Oja82]. Oja's mathematically formalized this rule in the following.

**Theorem 1.9 (Oja's Rule)**

Hebbian learning of a neuron can be modeled with

$$w_{t+1} - w_t = \eta y_t (x_t - y_t w_t) \tag{94}$$

To parse this, let's focus on a component element $(x_t)_i$. If $x_t$ represents the neurons firing and $y_t w_t$ represents which neurons' wiring scaled by how much the overall neuron has fired, we want the $y_t w_t$ to move closer to $x_t$. Therefore, we update $w_t$ in the direction of this difference, and then scale it by $\eta y_t$.

**Proof.**

Starting with Hebb's rule and applying normalization constraints, we have[a]

$$\mathbf{w}_{t+1} = \frac{\mathbf{w}_t + \eta y_t \mathbf{x}_t}{\|\mathbf{w}_t + \eta y_t \mathbf{x}_t\|_p} \tag{95}$$

where $\|\mathbf{v}\|_p = \left(\sum_{j=1}^m |v_j|^p\right)^{1/p}$. For a small learning rate $|\eta| \ll 1$, we can expand this equation as a power series in $\eta$. Using the binomial approximation $(1+\epsilon)^{-1/p} \approx 1 - \frac{\epsilon}{p}$ for small $\epsilon$, we get:

$$\mathbf{w}_{t+1} = \frac{\mathbf{w}_t}{\|\mathbf{w}_t\|_p} + \eta \left( \frac{y_t \mathbf{x}_t}{\|\mathbf{w}_t\|_p} - \frac{\mathbf{w}_t \mathbf{w}_t^T \mathbf{x}_t y_t}{\|\mathbf{w}_t\|_p^{p+1}} \|\mathbf{w}_t\|_p^{p-1} \right) + O(\eta^2) \tag{96}$$

For small $\eta$, the higher-order terms $O(\eta^2)$ vanish. We specify a linear neuron where the output is:

$$y_t = \mathbf{x}_t^T \mathbf{w}_t^{(p-1)} \tag{97}$$

where $\mathbf{w}_t^{(p-1)}$ denotes the component-wise power $[\mathbf{w}_t^{(p-1)}]_j = (w_t)_j^{p-1}$. We also specify that our weights are normalized: $\|\mathbf{w}_t\|_p = 1$.
For $p = 2$, this simplifies to $y_t = \mathbf{x}_t^T \mathbf{w}_t$ and $\|\mathbf{w}_t\|_2 = 1$. Substituting these conditions into our expansion gives:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta y_t (\mathbf{x}_t - y_t \mathbf{w}_t) \tag{98}$$

[a] Oja had $p = 2$ in his original paper, but in fact, any type of normalization, even linear, will give the same result without loss of generality.

Finally, we claim that $w_t$ asympototically converges to the first principal axes.

**Theorem 1.10 (Oja's Neuron Convergence to First Principal Component)**

Consider a single neuron trained by Oja's rule with weight vector $\mathbf{w}_t$ and input data $\{\mathbf{x}_t\}$ having correlation matrix $R = E[\mathbf{x}\mathbf{x}^T]$. Let $\mathbf{q}_1$ be the eigenvector corresponding to the largest eigenvalue $\lambda_1$ of $R$. Under the following conditions:

1. The learning rate $\eta(t)$ satisfies $\sum_{t=1}^\infty \eta(t) = \infty$ and $\sum_{t=1}^\infty \eta(t)^p < \infty$ for some $p > 1$
2. The activation function $y(\mathbf{x}(t))$ is continuously differentiable in both $\mathbf{x}$ and $\mathbf{w}$ with bounded derivatives
3. The correlation matrix $R$ has a unique largest eigenvalue $\lambda_1$

Then the weight vector $\mathbf{w}_t$ converges to $\pm\mathbf{q}_1$ as $t \to \infty$, and the variance of the neuron's output converges to the principal eigenvalue:

$$\lim_{t\to\infty} \sigma^2(t) = \lim_{t\to\infty} \langle y^2(t) \rangle = \lambda_1 \tag{99}$$

**Proof.**

TBD: Go over this. We analyze the convergence using Lyapunov function analysis. Consider the continuous-time version of Oja's rule:

$$\frac{d\mathbf{w}}{dt} = \eta(t)y(t)(\mathbf{x}(t) - y(t)\mathbf{w}(t)) \tag{100}$$

where $y(t) = \mathbf{w}(t)^T\mathbf{x}(t)$.
Taking the expected value and assuming ergodicity of the input process:

$$\frac{d\mathbf{w}}{dt} = \eta(t)E[y(\mathbf{x} - y\mathbf{w})] = \eta(t)E[\mathbf{w}^T\mathbf{x}(\mathbf{x} - \mathbf{w}^T\mathbf{x}\mathbf{w})] \tag{101}$$

This simplifies to:

$$\frac{d\mathbf{w}}{dt} = \eta(t)(R\mathbf{w} - \mathbf{w}^T R\mathbf{w}\mathbf{w}) \tag{102}$$

Since $\mathbf{w}$ is normalized ($\|\mathbf{w}\| = 1$), we have $\mathbf{w}^T R\mathbf{w} = \mathbf{w}^T R\mathbf{w}$. Let $\mathbf{w} = \sum_{i=1}^n c_i\mathbf{q}_i$ where $\{\mathbf{q}_i\}$ are the orthonormal eigenvectors of $R$ with eigenvalues $\{\lambda_i\}$ ordered as $\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_n$.
Substituting the eigendecomposition:

$$\frac{d\mathbf{w}}{dt} = \eta(t)\left(\sum_{i=1}^n c_i\lambda_i\mathbf{q}_i - \left(\sum_{i=1}^n c_i^2\lambda_i\right)\sum_{j=1}^n c_j\mathbf{q}_j\right) \tag{103}$$

For the coefficient $c_1$ corresponding to the first principal component:

$$\frac{dc_1}{dt} = \eta(t)c_1\left(\lambda_1 - \sum_{i=1}^n c_i^2\lambda_i\right) \tag{104}$$

Since $\lambda_1$ is the largest eigenvalue and $\sum_{i=1}^n c_i^2 = 1$, we have $\lambda_1 - \sum_{i=1}^n c_i^2\lambda_i > 0$ whenever $c_i \neq 0$ for $i > 1$.
For $i > 1$:

$$\frac{dc_i}{dt} = \eta(t)c_i\left(\lambda_i - \sum_{j=1}^n c_j^2\lambda_j\right) \tag{105}$$

Since $\lambda_i < \lambda_1$, the term $(\lambda_i - \sum_{j=1}^n c_j^2\lambda_j)$ becomes negative as $c_1$ grows, causing $c_i \to 0$ for $i > 1$.
The conditions on $\eta(t)$ ensure convergence by the Robbins-Monro theorem: the divergent sum condition ensures the algorithm can reach any point in the space, while the convergent power sum condition ensures the noise diminishes sufficiently for convergence.
As $t \to \infty$, we have $c_1 \to \pm 1$ and $c_i \to 0$ for $i > 1$, implying $\mathbf{w}_t \to \pm\mathbf{q}_1$.
The variance of the output is:

$$\sigma^2(t) = E[y^2(t)] = E[(\mathbf{w}_t^T\mathbf{x}_t)^2] = \mathbf{w}_t^T R\mathbf{w}_t \tag{106}$$

As $\mathbf{w}_t \to \pm\mathbf{q}_1$, we get:

$$\lim_{t\to\infty}\sigma^2(t) = \mathbf{q}_1^T R\mathbf{q}_1 = \lambda_1 \tag{107}$$

**Algorithm 1.4 (Oja's Rule)**

**Algorithm 3** Oja's Neuron Algorithm

```
procedure OjaNeuron($\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T$, $\eta$)
Require: Input data vectors $\mathbf{x}_t \in \mathbb{R}^n$ for $t = 1, 2, \dots, T$, learning rate $\eta > 0$
Ensure: Weight vector $\mathbf{w}$ that converges to the first principal component
    Initialize $\mathbf{w}_0 \in \mathbb{R}^n$ randomly with small values
    for $t \leftarrow 1$ to $T$ do
        $y_t \leftarrow \mathbf{w}_{t-1}^T \mathbf{x}_t$                          ▷Neural output
        $\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} + \eta y_t (\mathbf{x}_t - y_t \mathbf{w}_{t-1})$        ▷Oja's learning rule
    end for
    return $\mathbf{w}_T$
end procedure
```

## 1.5 The Importance of Standardizing

Say that a covariate was a person's height in meters. If we had measured in centimeters, then the values would a hundred times higher. PCA does not account for this and so in most cases, you should standardize your data to have component-wise unit variance before.

## 1.6 Asymptotic Analysis

It turns out that the elements of $\hat{\Sigma}$ are close entry-wise to those of $\Sigma$. But if this is true, then does it mean that the eigenvalues of the sample covariance matrix are close to the true eigenvalues of the covariance matrix? It turns out that the answer is no, and we need a proper metric to satisfy this assumption. The metric, as we can guess from linear algebra, is the operator norm, and we will show some results from matrix perturbation theory.

**Lemma 1.3 ()**

It turns out that

$$||\hat{\Sigma} - \Sigma|| = O_p\left(\frac{1}{\sqrt{n}}\right) \tag{108}$$

where $||\cdot||$ is the operator norm.

**Theorem 1.11 (Weyl's Theorem)**

If $\hat{\Sigma}$ and $\Sigma$ are close in the operator norm, then their eigenvalues are close.

$$||\hat{\Sigma} - \Sigma|| = O_p\left(\frac{1}{\sqrt{n}}\right) \implies |\hat{\lambda}_j - \lambda_j| = O_p\left(\frac{1}{\sqrt{n}}\right) \tag{109}$$

This only talks about their eigenvalues, but this does not necessarily imply that the eigenvalues are close. We need an extra condition.

**Theorem 1.12 (David-Kahan Theorem)**

If $\hat{\Sigma}$ and $\Sigma$ are close in the operator norm, and if the eigenvectors of $\Sigma$ are well-conditioned, then the eigenvectors of $\hat{\Sigma}$ are close to the eigenvectors of $\Sigma$. More specifically,

$$||\hat{v}_j - v_j|| \leq \frac{2^{3/2}||\hat{\Sigma} - \Sigma||}{\lambda_j - \lambda_{j+1}} \tag{110}$$

# 2 Factor Analysis

As we have constantly seen, there are specific themes that run between models. In PCA, we have taken some data $x$ in high-dimension $d$ and reduced it to a lower-dimensional orthogonal representation in $\mathbb{R}^k$. In other words, for some sample $x \in \mathbb{R}^{1 \times d}$, the projection onto its component space $xV_k \in \mathbb{R}^{1 \times k}$ is a more parsimonious representation with respect to some other basis vectors. The $v_1, \ldots, v_k$ are new features that are linear combinations of the old vectors. Are they interpretable? In some cases yes and in most cases no, which is why we also call them *latent variables* that live in a *latent space.*

Another type of model that encodes covariates in a latent space are factor models, which was developed by Spearman in 1904 [Spe04]. The general idea was that we have some $d$-dimensional random vector $x$, and we would like to encode it in a $k$-dimensional random vector $f$, called the *factors.* Since we are trying to compress the data, generally $k < d$. The first thing that comes to mind is to try and compare how the variables $x_i$ and $f_j$ correlate to each other, and this is exactly what Spearman did.

Before we get into factor models, let's step back and talk more about latent variable models. Colloquially, we would like to find the distribution of some data, whether it'd be $(x, y)$ supervised tasks or $x$ for unsupervised. For the unsupervised case, say that we have some covariates $x$ and we want to find its true distribution $p^*$. In density estimation so far, what we have done is define a family of distributions $\{p_\theta\}$ and optimize the loss by maximizing the MLE or something else.

$$\min_\theta L(p_\theta, p^*) = \max_\theta \prod_i p_\theta(x^{(i)}) \tag{111}$$

In order to do this we work with explicitly parameterized distribution families (e.g. Gaussian, Gamma, multinomial, etc.), but this is too simple to model complex things in real like (e.g. the distribution of faces).

Therefore, we consider *implicitly parameterized* probability distributions by "adding" a latent distribution $z$, creating the joint distribution $(x, z)$. This may look more complicated, but it captures a much richer family of distributions. For example, we might try modeling $x$ as a function of $z$, and try to learn some function $x = f(z)$. If we have an accurate function $f$, we can do many things.

1. Given an $x$, we might find the closest point on the image of $f$, perhaps some manifold, as low-rank approximation of $x$. This dimensionality reduction is essentially what PCA does with projections.[3]
2. If we can sample from $z$, then we can forward it through $f$ and can sample from $x$, making this is a generative model.

Like we do with everything else in math, we take a look at the simplest case when the class of functions are linear. This is known as *linear latent variable modeling.*

$$x = \mu + \Lambda z + \epsilon \tag{112}$$

where the noise $\epsilon$ is typically Gaussian and diagonal (but not necessarily the same component-wise variances).

## 2.1 Probabilistic PCA

We're talking about probabilistic PCA (PPCA), but why is this under factor analysis? I think that the jump from PCA to PPCA is greater than from PPCA to factor analysis, so I will introduce it here as a stepping stone. The main goal of PCA was to do dimensionality reduction by creating a botttleneck in the number of dimensions $k$. Our goal was to approximate the original random variable $x$ by first projecting onto a lower-dimensional space $z = V_k^T x$, and then embedding it through a linear injection.

$$x \approx V_k V_k^T x = V_k z \tag{113}$$

What if we don't restrict the dimensions at all, and just let $k = d$? Then we have the exact equation.

$$x = V V^T x = V z \tag{114}$$

[3] We will in fact extend PCA to probabilistic PCA soon to make it generative.

This is trivial since $V$—as an orthogonal matrix—satisfies $VV^T = V^TV = I_d$. Essentially, this is just a rotation of the axes. We have the maximal restriction when $k = 1$, and as we increase $k$ to $d$, our reconstruction loss will decrease. But almost always, if our data does not sit exactly on a subspace, we cannot get an exact reconstruction of our data with $k < d$. This is the motivation behind probabilistic PCA (PPCA), and there are essentially two goals that we want to solve. [TB99]

1. *Generative.* We would like our model to be generative. In regular PCA, we saw that for some $z \in \mathbb{R}^k$ in the latent component space, our reconstructed sample is $\hat{x} = SV_kz$. Therefore, if we just change $z$ from a point to a distribution (e.g. Gaussian), we can sample $z \sim \mathcal{N}(0, I_k)$, and then transform it to get a random variable $x = \mu + SV_kz$, which will give a density.

$$x \sim \mathcal{N}\big(\mu, (V_kS)(V_kS)^T\big) = \mathcal{N}\big(\mu, V_kSU^TUSV^T\big) = \mathcal{N}\big(\mu, X_k^TX_k\big) \tag{115}$$

2. *Exact Reconstruction.* However, $X_k \in \mathbb{R}^{n \times d}$ with $d << n$, and so $X_k^TX_k \in \mathbb{R}^{d \times d}$ is not full rank, and so the distribution is restricted to strictly the $k$-dimensional subspace $L_k \subset \mathbb{R}^D$. We want a model that has the both of best worlds: it has a bottleneck so that $k < d$, but at the same time it can do an exact reconstruction of the data. This can be solved by introducing a probabilistic error term $\epsilon$ that accounts for the variability of the data around the principal subspace. So let's add an isotropic Gaussian $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$[4], which gives us

$$x = SV_kz + \epsilon \implies X \sim \mathcal{N}(\mu, X_k^TX_k + \sigma^2 I) \tag{116}$$

Note again that we are adding *two* probabilistic terms, each of which serves a specific purpose. Great, and finally, let's clean up some notation and polish it up.

1. First, let's just call $W = SV_k \in \mathbb{R}^{d \times k}$, keeping the $k$ implicit, and treat it as the parameter to estimate.
2. Second, let's remove the assumption that $x$ is 0-mean and add back the mean term $\mu$.

This gives us our model.

**Definition 2.1 (Probabilistic PCA)**

The **probabilistic PCA** model is a latent factor model that summarizes the data generating distribution $x$ as

$$x = \mu + Wz + \epsilon, \qquad z \sim N(0, I_k), \epsilon \sim N(0, \sigma^2 I_d) \tag{117}$$

with parameters $\theta = \{\mu, W, \sigma\}$. Let's go over the assumptions.

1. $z$ is an isotropic Gaussian, which mirrors the fact that the principal components must be orthogonal (uncorrelated RVs). We can normalize it to be unit variance since the scaling can be done with $W$. This allows for a better representation where the components are uncorrelated, and—better yet—we can now sample from $z$.
2. $\epsilon$ is an isotropic Gaussian. This is just to ensure that we capture variability beyond the subspace. However, the isotropic part doesn't really seem to be justified... This assumption will be relaxed in factor models.
3. The $z$'s are uncorrelated with each other.
4. The $\epsilon$'s are uncorrelated with each other.
5. $\epsilon$ and $z$ are independent.

An immediate consequence is that the closed form of the distrbution of $x$ under this model can be solved. Calculating the pdf of $x$ requires us to marginalize out the $z$, but since marginals of Gaussians are Gaussians, this is quite easy.

[4] Why isotropic? No real reason. In factor models, we generalize this to arbitrary covariance matrices, and so PPCA is a specific case of factor analysis.

**Lemma 2.1 (Marginal Distribution of PPCA)**

We claim that given $\theta = \{\mu, W, \sigma\}$, we have

$$x \sim \mathcal{N}(\mu, WW^T + \sigma^2 I) \tag{118}$$

**Proof.**

By assuming it is Gaussian, you can just directly compute the expectation and covariance, but I will do the full density derivation. We start with the conditional and prior distributions from the probabilistic PCA model:

$$p(x|z, W) \propto \exp\left(-\frac{(x - W^T z)^T (x - W^T z)}{2\sigma^2}\right), \tag{119}$$

$$p(z) \propto \exp\left(-\frac{z^T z}{2}\right). \tag{120}$$

The joint distribution is given by:

$$p(x, z|W) = p(x|z, W)p(z) \quad \text{(since } z \perp W) \tag{121}$$

$$\propto \exp\left(-\frac{(x - W^T z)^T (x - W^T z)}{2\sigma^2} - \frac{z^T z}{2}\right). \tag{122}$$

Expanding the quadratic term $(x - W^T z)^T (x - W^T z)$:

$$(x - W^T z)^T (x - W^T z) = x^T x - x^T W^T z - z^T W x + z^T W W^T z \tag{123}$$

$$= x^T x - 2x^T W^T z + z^T W W^T z, \tag{124}$$

where we used the fact that $x^T W^T z = z^T W x$ (scalar quantities).
Substituting back into the joint distribution:

$$p(x, z|W) \propto \exp\left(-\frac{x^T x - 2x^T W^T z + z^T W W^T z}{2\sigma^2} - \frac{z^T z}{2}\right) \tag{125}$$

$$= \exp\left(-\frac{x^T x - 2x^T W^T z + z^T W W^T z}{2\sigma^2} - \frac{z^T z}{2}\right). \tag{126}$$

Factoring out $-\frac{1}{2}$ and collecting terms:

$$p(x, z|W) \propto \exp\left(-\frac{1}{2}\left(x^T \left(\frac{1}{\sigma^2} I\right) x + 2x^T \left(-\frac{1}{\sigma^2} W^T\right) z + z^T \left(\frac{1}{\sigma^2} W W^T + I\right) z\right)\right). \tag{127}$$

We can rewrite this in quadratic form. Let $v = \begin{bmatrix} x \\ z \end{bmatrix}$. Then:

$$p(x, z|W) \propto \exp\left(-\frac{1}{2}\begin{bmatrix} x^T & z^T \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma^2} I & -\frac{1}{\sigma^2} W^T \\ -\frac{1}{\sigma^2} W & \frac{1}{\sigma^2} W W^T + I \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix}\right). \tag{128}$$

This has the form of a multivariate Gaussian distribution:

$$p(v|W) \propto \exp\left(-\frac{1}{2}(v - \mu)^T \Sigma^{-1} (v - \mu)\right), \tag{129}$$

with $v = \begin{bmatrix} x \\ z \end{bmatrix}$, $\mu = 0$, and precision matrix:

$$\Sigma^{-1} = \begin{bmatrix} \frac{1}{\sigma^2} I & -\frac{1}{\sigma^2} W^T \\ -\frac{1}{\sigma^2} W & \frac{1}{\sigma^2} W W^T + I \end{bmatrix}. \tag{130}$$

Remember that if we write a multivariate Gaussian in partitioned form,

$$\begin{bmatrix} x \\ z \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} \mu_x \\ \mu_z \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xz} \\ \Sigma_{zx} & \Sigma_{zz} \end{bmatrix} \right), \tag{131}$$

then the marginal distribution $p(x)$ (integrating over $z$) is given by

$$x \sim \mathcal{N}(\mu_x, \Sigma_{xx}). \tag{132}$$

For probabilistic PCA we assume $\mu_x = 0$, but we partitioned $\Sigma^{-1}$ instead of $\Sigma$. To get $\Sigma$ we can use a partitioned matrix inversion formula:

$$\Sigma = \begin{bmatrix} \frac{1}{\sigma^2} I & -\frac{1}{\sigma^2} W^T \\ -\frac{1}{\sigma^2} W & \frac{1}{\sigma^2} W W^T + I \end{bmatrix}^{-1} = \begin{bmatrix} W^T W + \sigma^2 I & W^T \\ W & I \end{bmatrix}, \tag{133}$$

which gives that the solution to integrating over $z$ is

$$x | W \sim \mathcal{N}(0, W^T W + \sigma^2 I). \tag{134}$$

**Theorem 2.1 (Log-Likelihood and Risk of PPCA)**

The negative log-likelihood—which will act as our loss—of a single sample is

$$L(x \mid W, \mu, \sigma) = \frac{1}{2} (x - \mu)^T (W^T W + \sigma^2 I)^{-1} (x - \mu) \tag{135}$$

This allows us to define our risk (expected loss) and therefore our empirical risk/loss for a dataset of $n$ elements $\{x^{(i)}\}_{i=1}^n$.

$$R(W, \mu, \sigma) = \mathbb{E}_x[L(x \mid W, \mu, \sigma)] = \int L(x \mid W, \mu, \sigma)\, p(x)\, dx \tag{136}$$

$$\hat{R}(W, \mu, \sigma) = \frac{1}{2} \sum_{i=1}^n (x^{(i)} - \mu)^T (W^T W + \sigma^2 I)^{-1} (x^{(i)} - \mu) \tag{137}$$

**Proof.**

Trivial with previous marginal distribution.

Now that we have the likelihood of our dataset, optimizing this model is actually quite easy and can be solved analytically.

**Corollary 2.1 (MLE of PPCA Model)**

Given $x^{(i)} \sim X$ iid, the MLEs for $W, \mu, \sigma$ are

$$\mu^* = \frac{1}{n} \sum_{i=1}^n x^{(i)} \tag{138}$$

$$\sigma^{2*} = \frac{1}{d-k} \sum_{j=k+1}^d \lambda_j \tag{139}$$

$$W^* = R(\hat{\Sigma} - \sigma^{2*} I_d)^{1/2} V_k \tag{140}$$

where $\lambda_j$ are the eigenvalues of $X^T X$ in decreasing order, $V_k$ is the truncated orthogonal matrix consisting of the first $k$ columns of $V$ for SVD $X = USV^T$, and $R$ is any unitary matrix.[a]

[a] Note that $W^*$ is not unique. Say that $W^*$ is an MLE, then, for any unitary $R \in \mathbb{R}^{d \times d}$, we have $W^{*T} W^* = (RW^*)^T (RW^*)$.

**Proof.**

We could also just differentiate the expected risk directly, but for no particular reason I will differentiate the empirical risk.

1. For $\mu^*$, we use the matrix derivative $\frac{\partial}{\partial x} x^T A x = 2Ax$ and get

$$0 = \frac{1}{2} \sum_{i=1}^n \frac{\partial}{\partial \mu} \Big\{ (x^{(i)} - \mu)^T (W^T W + \sigma^2 I)(x^{(i)} - \mu) \Big\} \tag{141}$$

$$= \frac{1}{2} \sum_{i=1}^n 2(W^T W + \sigma^2 I)(x^{(i)} - \mu) \tag{142}$$

$$= (W^T W + \sigma^2 I) \left( \sum_{i=1}^n x^{(i)} - \mu \right) \tag{143}$$

and since $W^T W + \sigma^2 I$ is positive definite, its inverse is positive definite and so it can only be 0 when it is mapping the 0 vector. So $\mu = \sum_{i=1}^n x^{(i)}$.

2. For $\sigma$, we first take a look at $C = \mathrm{Var}[X] = WW^T + \sigma^2 I$. It is the sum of positive semidefinite matrices that are also symmetric, so by the spectral theorem it is diagonalizable and has full rank $d$. But $WW^T$ is rank $k$, so $d - k$ of the eigenvalues of $WW^T$ is 0, indicating that the same $d - k$ smallest eigenvalues of $C$ is $\sigma^2$. Therefore, we can take the smallest $d - k$ eigenvalues of our MLE estimator of $C$ and average them to get our MLE for $\sigma$.

$$\hat{\sigma}^{2*} = \frac{1}{d - k} \sum_{j=k+1}^d \lambda_j \tag{144}$$

3. TBD: Justify this again. For $W$, we can set $\mu^*$ first and then compute

$$\widehat{\mathrm{Var}}(\mu^*) = \hat{\Sigma} = \frac{1}{n} \sum_{i=1}^n (x^{(i)} - \mu^*)(x^{(i)} - \mu^*)^T \tag{145}$$

We can approximate $WW^T = C - \sigma^2 I \approx \hat{\Sigma} - \hat{\sigma}^{2*} I$, and by further taking the eigendecomposition $C = U \Sigma U^T \implies WW^T = U(\Sigma - \sigma^2 I)U^T$ and cutting off the last $d - k$ smallest eigenvalues and their corresponding eigenvectors, we can get

$$W^* = R(\Sigma - \sigma^{2*} I_d)^{1/2} V_k \tag{146}$$

where the $R$ just accounts for any unitary matrix.

It turns out that the MLE of $W$ for PPCA, denoted $W^*_{\text{PPCA}} = (\Sigma - \sigma^{*2} I_d)^{1/2} V_k$ has columns in the same direction but with a smaller length than $W_{\text{PCA}} = SV_k = \Sigma^{1/2} V_k$ for standard PCA. For this reason, you should think of PPCA as “almost” PCA.

Our next result is intuitive. We have introduced the error term $\epsilon$ that allows us to extend beyond the principal subspace. If we let $\epsilon$ vanish, the density model defined by PPCA becomes very sharp around these $d$ dimensions spanned by the columns of $W$. At 0, we are reduced to regular PCA.

**Theorem 2.2 (PPCA as $\sigma \to 0$)**

As $\sigma \to 0$, the MLE estimates of $W$ is equivalent to that of PCA. That is, when $W \in \mathbb{R}^{d \times k}$,

$$W^* = \Sigma V_k \tag{147}$$

where $X = U \Sigma V^T$, and $V_k$ is the matrix formed by the first $k$ columns of $V$. That is, the conditional expected value of $z$ given $X$ becomes an orthogonal projection of $X - \mu$ onto the subspace spanned by the columns of $W$.

**Proof.**

At 0, our MLE of $W$ is simplified and we have

$$x = W^* z + \mu^* + \epsilon = \Sigma V_k z + \mu^* \tag{148}$$

which essentially reduces to regular PCA.

Intuitively, we can see that we are estimating the Gaussian, which corresponds to the mean squared distance from each $x^{(i)}$ to $\ell_k$.

## 2.2 Linear Factor Models

A linear factor model is pretty much the same thing as PPCA. In fact, I don't even know why these two models are distinguished. The only two differences is notation and that linear factor models loosen the restriction that the covariance matrix of $\epsilon$ must be isotopic.

**Definition 2.2 (Linear Gaussian Factor Model)**

A **linear Gaussian factor model** is a latent factor model that models the data generating distribution $x$ over $\mathbb{R}^d$ as

$$x = \mu + \Lambda \eta + \epsilon, \eta \sim N(0, I_k), \epsilon \sim N(0, \Psi) \tag{149}$$

for some diagonal matrix $\Psi$. The parameters are $\theta = \{\Lambda, \mu, \Psi\}$.[a] $\mu = \mathbb{E}[x]$ simply normalizes the distribution, $\eta \in \mathbb{R}^k$ are the **factors** that act as latent variables. $\Lambda \in \mathbb{R}^{d \times k}$ is the **loading matrix** that maps the factors to the samples and $\epsilon$ is an error term. Let's go over the assumptions.

1. $\eta$ is 0-mean (since the translation is captured in $\mu$) and isotropic.[b]
2. $\epsilon$ has mean 0 and has component variables that are uncorrelated, which is reflected in $\Psi$ being diagonal.
3. $\eta$'s are independent of each other.
4. $\epsilon$'s are independent of each other.
5. $\epsilon$'s are uncorrelated with $\eta$'s.

[a] It is also common notation to use $L$ and $f$ as the loading matrix and factors.

[b] This parallels the uncorrelated assumptions of PCA and PPCA.

The biggest glaring change is really just the terminology. Rather than components, we say *factors*, and rather than projections, we say *loadings*. Finally, we say observable component $x_i$ *loads on* factor $z_j$ if $\Lambda_{ji} \neq 0$, i.e. if $z_j$ affects the value of $x_i$. However, factor models just have more degrees of freedom. In practice, they also perform very similarly, but we will discuss the subtleties soon.

To get more comfortable with the notation, let's write out the formula given a data matrix. If we were working with the data matrix $X \in \mathbb{R}^{n \times d}$, then we would be using right matrix multiplication, and so our model will look like

$$X - \mu = \eta \Lambda + \epsilon \tag{150}$$

where $\eta \in \mathbb{R}^{n \times k}$, $\Lambda \in \mathbb{R}^{k \times d}, \epsilon \in \mathbb{R}^{n \times d}$.

**Lemma 2.2 (Marginal Distribution of Linear Gaussian Factor Model)**

We claim that given $\theta = \{\mu, \Lambda, \Psi\}$, we have

$$x \sim \mathcal{N}(\mu, \Lambda\Lambda^T + \Psi) \tag{151}$$

**Proof.**

We know that linear transformations and sums of Gaussians are Gaussian. Therefore, it suffices to compute the mean and variance. For the mean, we have

$$\begin{aligned} \mathbb{E}[x] &= \mathbb{E}[\mu + \Lambda\eta + \epsilon] \qquad (152) \\ &= \mu + \Lambda\mathbb{E}[\eta] + \mathbb{E}[\epsilon] \qquad (153) \\ &= \mu + \Lambda \cdot 0 + 0 \qquad (154) \\ &= \mu \qquad (155) \end{aligned}$$

For the variance, since $x - \mu = \Lambda\eta + \epsilon$,

$$\begin{aligned} \mathrm{Var}(x) &= \mathbb{E}[(x-\mu)(x-\mu)^T] \qquad (156) \\ &= \mathbb{E}[(\Lambda\eta + \epsilon)(\Lambda\eta + \epsilon)^T] \qquad (157) \\ &= \mathbb{E}[(\Lambda\eta + \epsilon)(\eta^T\Lambda^T + \epsilon^T)] \qquad (158) \\ &= \mathbb{E}[\Lambda\eta\eta^T\Lambda^T + \Lambda\eta\epsilon^T + \epsilon\eta^T\Lambda^T + \epsilon\epsilon^T] \qquad (159) \end{aligned}$$

Since $\eta$ and $\epsilon$ are uncorrelated, we have

$$\begin{aligned} \mathbb{E}[\eta\epsilon^T] &= \mathbb{E}[\eta]\mathbb{E}[\epsilon^T] = 0 \qquad (160) \\ \mathbb{E}[\epsilon\eta^T] &= \mathbb{E}[\epsilon]\mathbb{E}[\eta^T] = 0 \qquad (161) \end{aligned}$$

Therefore, the cross terms vanish.

$$\begin{aligned} \mathrm{Var}(x) &= \mathbb{E}[\Lambda\eta\eta^T\Lambda^T] + \mathbb{E}[\epsilon\epsilon^T] \qquad (162) \\ &= \Lambda\mathbb{E}[\eta\eta^T]\Lambda^T + \mathbb{E}[\epsilon\epsilon^T] \qquad (163) \\ &= \Lambda I_k \Lambda^T + \Psi \qquad (164) \\ &= \Lambda\Lambda^T + \Psi \qquad (165) \end{aligned}$$

Note that in factor models, we generally do not need the assumption that $\epsilon$ is Gaussian. This is just an extra choice we make in order to compute the marginal distribution.

**Theorem 2.3 (Log-Likelihood and Risk of Linear Gaussian Factor Model)**

The negative log-likelihood—which will act as our loss—of a single sample is

$$L(x \mid \Lambda, \mu, \Psi) = \frac{1}{2}\log|\Lambda\Lambda^T + \Psi| + \frac{1}{2}(x-\mu)^T(\Lambda\Lambda^T + \Psi)^{-1}(x-\mu) \tag{166}$$

This allows us to define our risk (expected loss) and therefore our empirical risk/loss for a dataset of

$n$ elements $\{x^{(i)}\}_{i=1}^n$.

$$R(\Lambda, \mu, \Psi) = \mathbb{E}_x[L(x \mid \Lambda, \mu, \Psi)] = \int L(x \mid \Lambda, \mu, \Psi)\, p(x)\, dx \tag{167}$$

$$\begin{aligned}\hat{R}(\Lambda, \mu, \Psi) &= \frac{1}{2n} \sum_{i=1}^n \left[\frac{p}{2} \log 2\pi + \log |\Lambda\Lambda^T + \Psi| + (x^{(i)} - \mu)^T (\Lambda\Lambda^T + \Psi)^{-1} (x^{(i)} - \mu)\right] &(168)\\ &= \frac{p}{2} \log 2\pi + \frac{1}{2} \log |\Lambda\Lambda^T + \Psi| + \frac{1}{2} \mathrm{Tr}\left[(\Lambda\Lambda^T + \Psi)^{-1} \hat{\Sigma}\right] &(169)\end{aligned}$$

where $\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^n (x^{(i)} - \mu)(x^{(i)} - \mu)^T$ is the sample covariance matrix.[a]

[a] Notice that the likelihood—and therefore the loss—only involves the data through the sample covariance matrix $\hat{\Sigma}$. The actual factor scores $\eta$ are not needed.

**Proof.**

From our previous result, we know that the marginal distribution of the observed data is:

$$x \sim \mathcal{N}(\mu, \Lambda\Lambda^T + \Psi) \tag{170}$$

The likelihood function is the probability density:

$$p(x|\Lambda, \mu, \Psi) = \frac{1}{(2\pi)^{p/2} |\Lambda\Lambda^T + \Psi|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T (\Lambda\Lambda^T + \Psi)^{-1} (x - \mu)\right) \tag{171}$$

Taking the negative log-likelihood to define our loss function:

$$\begin{aligned}L(x \mid \Lambda, \mu, \Psi) &= -\log p(x|\Lambda, \mu, \Psi) &(172)\\ &= \frac{p}{2} \log(2\pi) + \frac{1}{2} \log |\Lambda\Lambda^T + \Psi| + \frac{1}{2}(x - \mu)^T (\Lambda\Lambda^T + \Psi)^{-1} (x - \mu) &(173)\end{aligned}$$

Dropping the constant term $\frac{p}{2} \log(2\pi)$ (which doesn't affect optimization), we get:

$$L(x \mid \Lambda, \mu, \Psi) = \frac{1}{2} \log |\Lambda\Lambda^T + \Psi| + \frac{1}{2}(x - \mu)^T (\Lambda\Lambda^T + \Psi)^{-1} (x - \mu) \tag{174}$$

The risk is the expected loss over the true data distribution:

$$R(\Lambda, \mu, \Psi) = \mathbb{E}_x[L(x \mid \Lambda, \mu, \Psi)] = \int L(x \mid \Lambda, \mu, \Psi)\, p(x)\, dx \tag{175}$$

The empirical risk is the average loss over our dataset. For completeness, let's include back the constant terms.

$$\begin{aligned}\hat{R}(\Lambda, \mu, \Psi) &= \frac{1}{n} \sum_{i=1}^n L(x^{(i)} \mid \Lambda, \mu, \Psi) &(176)\\ &= \frac{1}{n} \sum_{i=1}^n \left[\frac{p}{2} \log 2\pi + \frac{1}{2} \log |\Lambda\Lambda^T + \Psi| + \frac{1}{2}(x^{(i)} - \mu)^T (\Lambda\Lambda^T + \Psi)^{-1} (x^{(i)} - \mu)\right] &(177)\\ &= \frac{p}{2} \log 2\pi + \frac{1}{2} \log |\Lambda\Lambda^T + \Psi| + \frac{1}{2n} \sum_{i=1}^n (x^{(i)} - \mu)^T (\Lambda\Lambda^T + \Psi)^{-1} (x^{(i)} - \mu) &(178)\end{aligned}$$

Now we use a neat trick. The sum on the right is just a scalar, so it is equal to its own trace. Then by using the linearity of trace and the cyclic trace property, we get

$$\begin{aligned}
\mathrm{Tr}\left(\sum_{i=1}^n (x^{(i)}-\mu)^T(\Lambda\Lambda^T+\Psi)^{-1}(x^{(i)}-\mu)\right) &= \mathrm{Tr}\left(\sum_{i=1}^n (x^{(i)}-\mu)^T(\Lambda\Lambda^T+\Psi)^{-1}(x^{(i)}-\mu)\right) && (179)\\
&= \sum_{i=1}^n \mathrm{Tr}\left((x^{(i)}-\mu)^T(\Lambda\Lambda^T+\Psi)^{-1}(x^{(i)}-\mu)\right) && (180)\\
&= \sum_{i=1}^n \mathrm{Tr}\left((\Lambda\Lambda^T+\Psi)^{-1}(x^{(i)}-\mu)(x^{(i)}-\mu)^T\right) && (181)\\
&= \sum_{i=1}^n \mathrm{Tr}\left((\Lambda\Lambda^T+\Psi)^{-1}\hat{\Sigma}\right) && (182)\\
&= \sum_{i=1}^n \mathrm{Tr}\left((\Lambda\Lambda^T+\Psi)^{-1}\hat{\Sigma}\right) && (183)\\
&= n\,\mathrm{Tr}\left((\Lambda\Lambda^T+\Psi)^{-1}\hat{\Sigma}\right) && (184)
\end{aligned}$$

and substituting this back in gives our desired result.

Unlike PPCA, this empirical loss is not convex, so it must be optimized with numerical methods. In most cases, the solutions tend to be similar, and the MLE estimates between PPCA and FA tend to share the same properties, e.g. the non-uniqueness of $W$ up to rotational factor.

## 2.3 Numerical Solvers for Linear Factor Models

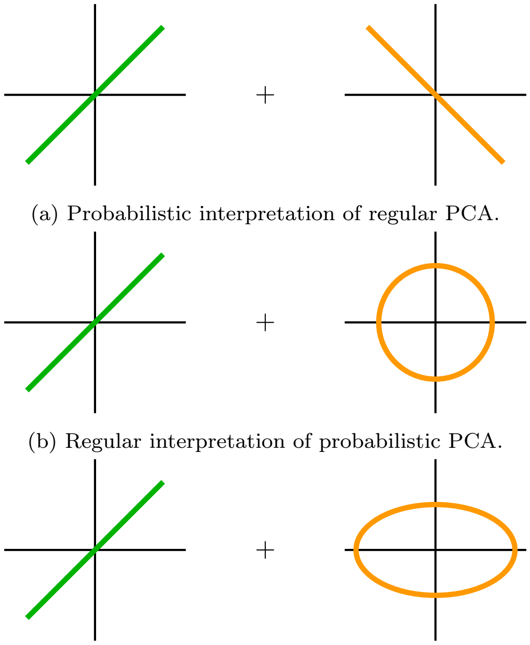
Since there is not an analytical solution of factor models, much less an analogue of Eckart-Young theorem, we introduce the development of mainstream numerical techniques.

### 2.3.1 Iterated Principal Factors

**Algorithm 2.1 (Iterated Principal Factors)**

## 2.4 PCA vs PPCA vs Factor Models

We have stressed that FA is the same as PPCA, but let's get serious and talk about the differences. In a sense, we could have reformulated PCA as a probabilistic model by assuming that the error terms were normally distributed with variance *orthogonal* to the principal subspace. This would have given the exact same expected risk and thus the models would be equivalent. On the other hand, probabilistic PCA assumes an isotropic Gaussian and factor models assume uncorrelated Gaussian.



(a) Probabilistic interpretation of regular PCA.

(b) Regular interpretation of probabilistic PCA.

(c) Factor analysis.

Figure 6: The green (left) represents the latent feature, and the orange on the right represents the errors. Figure credits to *ttnphns* from here.

It turns out that this simple change of not having constant variance in the Gaussian error term $\epsilon$ has some implications.

**Example 2.1 (Scaling Features does not Affect FA)**

Remember that in PCA, we must normalize each component of the data to unit variance since PCA tries to maximize variance of projections. However, FA does not chase large-noise features that are uncorrelated with other features.

**Example 2.2 (FA is Affected by Rotation of Data)**

FA is affected by rotation of the data. [Sch18]

**Theorem 2.4 (Factor Models Converge to PPCA in High Dimensions)**

As $d \to +\infty$, factor models are the same model as PPCA.

# 3 Linear Independent Component Analysis

PCA and factor analysis have found an enormous number of applications. In fact, it is one of the go-to methods for EDA, and exploratory factor analysis (EFA) is often synonymous with PCA.

However, a shortcoming of FA is that we can't even identify factor directions since the loading matrix $W$ can be rotated to produce just as good of a model. Therefore, Pierre Comon in 1992 had produced *independent component analysis* (ICA). It is essentially a method to separate a multivariate signal into additive, statistically independent components. It does come with a lot of assumptions, and is a specific instance of a linear factor model where $\mu = 0$ and $\epsilon = 0$.

**Definition 3.1 (Linear ICA)**

In **linear ICA**, we model the true distribution of $x$ as

$$x = Wz, \qquad z \sim \mathbb{P}_z \tag{185}$$

where $x$—the **mixture vector**—and $z$ are random variables of $\mathbb{R}^d$ and $\mathbb{R}^k$, and $W \in \mathbb{R}^{d \times k}$ is a **mixing matrix**. The parameters are both $W$ and $z$, and we need to recover them given $x$. We have 2 strong assumptions.

1. Each component of $z$ is independent (not just uncorrelated).
2. Independent components of $z$ must *not* be Gaussian. This is needed for us to be able to "unmix" the signals.[a]

[a] To see why, just suppose $z$ was Gaussian, and so the vector $Rz$ is also Gaussian for any invertible $R$. Therefore, we could find an infinite number of solutions of form $x = WR^{-1}Rz$ and have no way to separate them.

**Algorithm 3.1 (Fitting)**

Now let's see how linear ICA actually estimates $W$ and $z$. Once $W$ is estimated, the latent components of a given test mixture vector, $x^*$ is computed by $z^* = W^{-1}x^*$. So now all there's left to do is to estimate $W$, which we want to estimate so that $W^{-1}x$ is far from Gaussian. The reason for this is that given a bunch of independent non-Gaussian $h_i$'s, if we mix them with a matrix that is not $\pm I$ , then by CLT, a linear combination of random variables will tend to be Gaussian, and so for an arbitrary $W$ we would expect $x$ to be Gaussian. Therefore, what we want to do is guess some matrix $A$, and compute

$$Ax = AWh \tag{186}$$

and if we get things right, $A \approx W^{-1}$, and the result of $Ax$ would look pretty non-Gaussian. If it it not the case, then $AW$ will still be some mixing matrix, and so $Ax$ would look Gaussian. So now the question reduces to how do we choose this $A$? There are multiple ways to measure non-Gaussianity:

1. The absolute or squared kurtosis, which is 0 for Gaussians. This is a differentiable function w.r.t. $W$, so we can try maximizing it. This is done for the sample kurtosis, of course.
2. Another measure is by maximizing the neg-entropy.

There are further ambiguities with ICA regarding uniqueness of a best representation. For one, we can only estimate the latent components up to a scaling factor since we will still get

$$x = (\alpha W)(\frac{1}{\alpha} z) \text{ for some } \alpha > 0 \tag{187}$$

We can fix this by forcing $\mathbb{E}[z_i^2] = 1$. However, there is still an ambiguity for the sign of hidden components, but this is insignificant in most applications. Second, we can estimating the components up to permutation. We have

$$x = WP^{-1}Pz \tag{188}$$

for some permutation matrix $P$.

**Example 3.1 (Blind Source Separation)**

The canonical example of ICA is *blind source separation.*

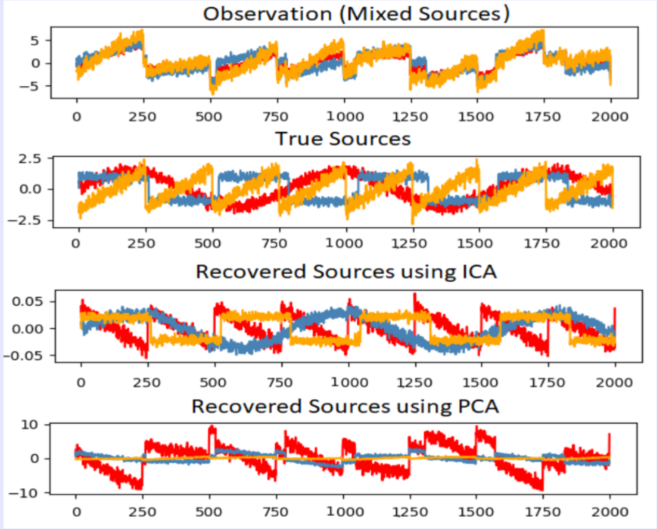


Figure 7: We can perform this on three mixed signals with additive noise, and ICA does very well, though again some recovered signals are scaled or permuted weirdly.

# 4 Robust PCA

# 5 Sparse PCA

# 6 Dynamic PCA

# 7 Functional PCA

# 8 Kernel PCA

**Definition 8.1 (Kernel PCA)**

Let $N_i$ be the neighborhood around $X_i$. Then, we want to find a mapping $W : \mathbb{R}^n \to \mathbb{R}^k$ that minimizes

$$\min_W \sum_{i=1}^n \bigg|\bigg| X_i - \sum_{j \in N_i} W_{ij} X_j \bigg|\bigg|^2 \text{ where } \sum_j W_{ij} = 1 \tag{189}$$

We can constrain the weights in $W$ so that anything that is not in the neighborhoods are 0.

# 9 Group PCA

# 10 Slow Feature Analysis

Slow feature analysis also another special case of a linear factor model that uses information from time signals to learn invariant features. It is motivated by a general principle called the **slowness principle**. The idea is that the important characteristics of scenes change very slowly compared to the individual measurements that make up a description of a scene. For example, in computer vision, individual pixels can change very rapidly. If a zebra moves from left to right across the image, an individual pixel wil rapidly change from black to white. By comparison, the feature indicating whether a zebra is in the image will not change at all, and the feature describing the zebra's position will change slowly. Therefore, we want to regularize our model to learn features that change slowly over time.

We can apply the slowness principle to any differentiable model trained with gradient descent. That is, we can add the following term to the loss function:

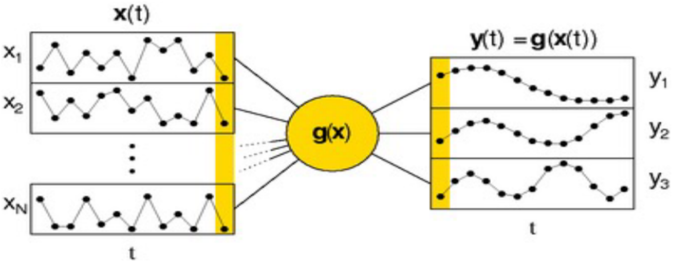
$$\lambda \sum_i d\big(f(x^{(t+1)}), f(x^{(t)})\big) \tag{190}$$

where $\lambda$ is a hyperparameter determining the strength of the slowness regularization term, $t$ is the time index, $f$ is the feature extractor to be regularized, and $d$ is the distance between $f(x^{(t)})$ and $f(x^{(t+1)})$. A common choice for $d$ is the mean squared difference.

Essentially, given a set of time-varying input signals $x^{(t)}$, SFA learns a nonlinear function $f$ that transforms $x$ into slowly-varying output signals $y$. Obviously, we can't just take some trivial function like $f = 0$, so we have the following constraints

$$\mathbb{E}_t[f(x^{(t)})_i] = 0 \tag{191}$$

$$\mathbb{E}_t[f(x^{(t)})_i^2] = 1 \tag{192}$$



We can restrict the nonlinear $f$ to some subspace of functions, and this becomes a standard optimization problem where we solve

$$\min_\theta \mathbb{E}_t\big[\big(f(x^{(t+1)})_i - f(x^{(t)})_i\big)^2\big] \tag{193}$$

# 11 Sparse Dictionary Learning

Latent variables can help us represent data in lower dimensions, but another advantage is that we can get *sparse* representations as well. What we want to do in sparse coding is that for each input $x^{(i)}$, we want to find a latent representation $z^{(i)}$ such that it is sparse (i.e. has many 0s) and also we can reconstruct the original input $x^{(i)}$ well. We have basically two things to optimize: the latent representations $z$ and the decoding mechanism, which we can do with a *dictionary matrix* $D$. Note that we are optimizing for *both* the latent encodings and the decoding mechanism, and so this isn't a generative model.

**Definition 11.1 (Sparse Dictionary Encoding Model)**

The **sparse dictionary encoding model** is a representation model defined

$$X = g_D(Z) = DZ \tag{194}$$

where $D \in \mathbb{R}^{d \times k}$ is a **dictionary matrix** that decodes the latent $Z \in \mathbb{R}^k$ to $X \in \mathbb{R}^d$. Note that both the $z^{(i)}$'s and $D$ are optimized, so we want to perform the *joint* optimization[a]

$$\min_D \frac{1}{N} \sum_{i=1}^N \min_{z^{(i)}} \underbrace{\frac{1}{2} ||x^{(i)} - Dz^{(i)}||_2^2}_{\text{reconstruction error}} + \underbrace{\lambda ||z^{(i)}||_1}_{\text{sparsity penalty}} \tag{195}$$

[a] To break this term down, let's just assume that we have a fixed dictionary $D$. Then, we just need to minimize with respect to each $h^{(t)}$. Now we can add the dictionary parameter back again.

Note that the reconstruction, or decoding, of $x = Dz$ is linear and explicit, but if we want to encode $x \mapsto z$, we need to substitute the $x$ into the term above and minimize it w.r.t. $D$ and $z$ to solve it. Therefore, this encoder is an implicit and *nonlinear* function of $x$.

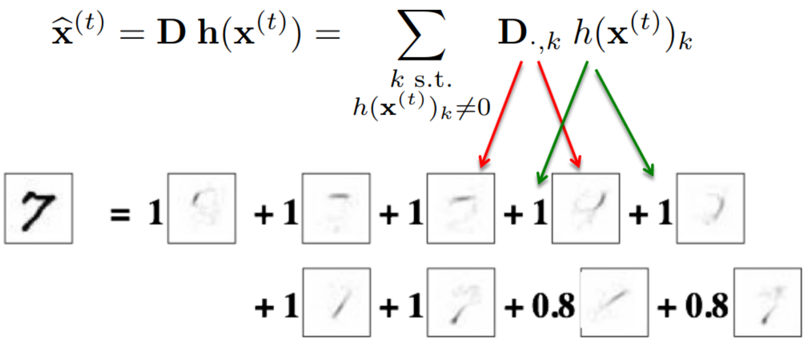

$\widehat{\mathbf{x}}^{(t)} = \mathbf{D}\,\mathbf{h}(\mathbf{x}^{(t)}) = \sum_{\substack{k \text{ s.t.} \\ h(\mathbf{x}^{(t)})_k \neq 0}} \mathbf{D}_{\cdot,k}\, h(\mathbf{x}^{(t)})_k$
= 1
+ 1
+ 1
+ 1
+ 1
+ 1
+ 1
+ 0.8
+ 0.8

Figure 8: We can reconstruct an image of a seven as a linear combination of a set of images. Note that each of the images of strokes are columns of $W$ and the coefficients make up the sparse vector $h$.

Let's think about how we can optimize the objective function w.r.t. $h$, keeping $D$ constant. We can do stochastic gradient descent, which gives us the steps

$$\nabla_{h^{(t)}} \mathcal{L}(x^{(t)}) = D^T (Dh^{(t)} - x^{(t)}) + \lambda \, \text{sign}(h^{(t)}) \tag{196}$$

but this wouldn't achieve sparsity since it overshoots the 0 all the time. Therefore, we can clip it, or we can use proximal gradient descent/ISTA to take a step, and shrink the parameters according to the L1 norm.

$$h^{(t)} = h^{(t)} - \alpha D^T (Dh^{(t)} - x^{(t)}) \tag{197}$$

$$h^{(t)} = \text{shrink}(h^{(t)}, \alpha\lambda) \tag{198}$$

where $\text{shrink}(a, b) = [\ldots, \text{sign}(a_i) \max(|a_i| - b_i, 0), \ldots]$. This is guaranteed to converge if $1/\alpha$ is bigger than the largest eigenvalue of $D^T D$.

# Bibliography

[EY36] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, September 1936.

[Hot33] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:498–520, 1933.

[Lan50] Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Natl. Bur. Stand. B*, 45:255–282, 1950.

[MP29] R. V. Mises and H. Pollaczek-Geiringer. Praktische Verfahren der Gleichungsauflösung . *Zeitschrift Angewandte Mathematik und Mechanik*, 9(1):58–77, January 1929.

[Oja82] Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, 1982.

[Pea01] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

[Sch18] Mark Schmidt. Probabilistic pca, factor analysis, independent component analysis, 2018. CPSC 540: Machine Learning, Lecture 17.5.

[Sha19] Cosma Rohilla Shalizi. *Advanced Data Analysis from an Elementary Point of View*. Cambridge University Press, 2019.

[Spe04] C. Spearman. "general intelligence," objectively determined and measured. *The American Journal of Psychology*, 15(2):201–292, 1904.

[TB99] Michael E. Tipping and Christopher M. Bishop. Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society Series B*, 61(3):611–622, 1999.

[TP91] M.A. Turk and A.P. Pentland. Face recognition using eigenfaces. *Journal of Cognitive Neuroscience*, pages 586–591, 1991.