CS281 Section 4: Factor Analysis and PCA

Scott Linderman

At this point we have seen a variety of machine learning models, with a particular emphasis on models for supervised learning. In particular, we have discussed linear regression for fitting continuous outputs given their corresponding features, and classification methods which learn a linear decision boundary, like logistic regression. Finally, we've extended these methods to include nonlinear functions of the input using generalized linear models.

In many settings, however, we do not have a set of outputs and features, we just have a set of data. This is the domain of **unsupervised machine learning**. Our goals in this setting are less clear. Rarely is our data entirely random. We often believe there is some latent, low-dimensional structure in the data that is corrupted by noise. Unsupervised machine learning is about finding this latent structure, and today we will discuss some of the most widely used methods for doing so.

1. Principal Component Analysis (PCA)

   Imagine we are presented with a bunch of data $\{\boldsymbol{x}_n\}$, where each $\boldsymbol{x}_n$ lives in $\mathbb{R}^D$. For example, in Figure 1 we have a cloud of points in $\mathbb{R}^2$. In many cases we believe the data is actually lower dimensional, 1-dimensional in this case. That is, the 2-D data points are well approximated by a line, and each data point is described by a 1-D value indicating the location on the line. How could we find the line that best explains these data? This is very similar to linear regression, except in this case we do not know the latent position along the line.

   (a) Heuristic motivation

   One way is to assume the data is Gaussian distributed and find the MLE covariance. This corresponds to the sample covariance. Then, intuitively, the best line will be parallel to the long axis of the ellipse corresponding to the covariance matrix. That is, the best line will be parallel to the eigenvector of the covariance matrix with the largest eigenvalue. Each point will be approximated by its projection onto this eigenvector.

   This is the intuition behind PCA: We find the eigenvectors of the sample covariance matrix, and then summarize the observed data as projections onto the top eigenvectors, or **principal components**. The size of this projection for a given datapoint $\boldsymbol{x}_n$ is called the **score**.

   (b) Minimizing reconstruction error

   Can we formalize the process by which we arrived at this algorithm? Under what objective function are the eigenvectors with largest eigenvalues the "best" principal components? It turns out that given a data matrix $\boldsymbol{X} \in \mathbb{R}^{D \times N}$ (columns correspond to $\boldsymbol{x}_n$), PCA finds principal components $\boldsymbol{W} \in \mathbb{R}^{D \times L}$ and scores $\boldsymbol{Z}^{L \times N}$ that minimize the

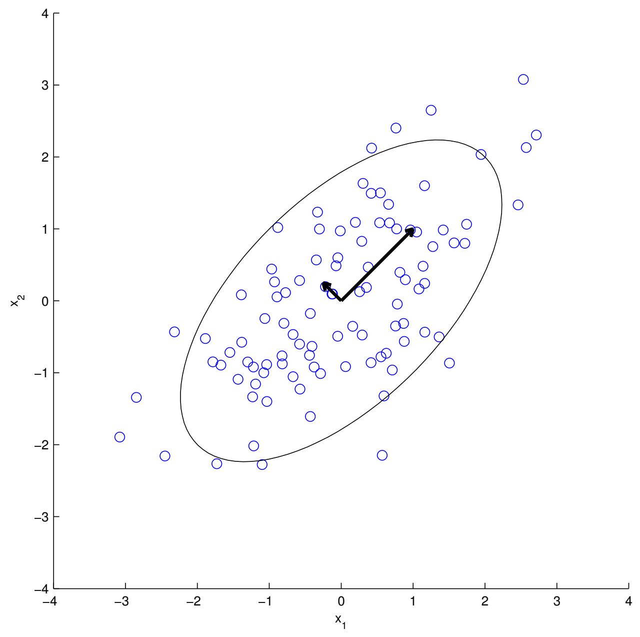

Figure 1: A cloud of data in $\mathbb{R}^2$ that we would like to approximate with a lower dimensional representation.

reconstruction error

$$J(\boldsymbol{W}, \boldsymbol{Z}) = ||\boldsymbol{X} - \boldsymbol{W}\boldsymbol{Z}||_F^2,$$

where $\boldsymbol{W}$ is constrained to be orthonormal and the error is measured with respect to the Frobenius norm.

Let us solve for the optimal first principal component $\boldsymbol{w}_1$ and "scores" $\boldsymbol{z}_1$ under the objective given in the first characterization

$$\begin{aligned} J(\boldsymbol{w}_1, \boldsymbol{z}_1) &= \frac{1}{N}\sum_n ||\boldsymbol{x}_n - z_{1,n}\boldsymbol{w}_1||^2 \\ &= \frac{1}{N}\sum_n \left[\boldsymbol{x}_n^T\boldsymbol{x}_n - 2z_{1,n}\boldsymbol{w}_1^T\boldsymbol{x}_n + z_{1,n}^2\boldsymbol{w}_1^T\boldsymbol{w}_1\right] \\ &= \frac{1}{N}\sum_n \left[\boldsymbol{x}_n^T\boldsymbol{x}_n - 2z_{1,n}\boldsymbol{w}_1^T\boldsymbol{x}_n + z_{1,n}^2\right]. \end{aligned}$$

Taking partial derivatives with respect to $z_{1,n}$ yields

$$\begin{aligned} \frac{\partial J(\boldsymbol{w}_1, \boldsymbol{z}_1)}{\partial z_{1,n}} &= \frac{1}{N}\left[-2\boldsymbol{w}_1^T\boldsymbol{x}_n + 2z_{1,n}\right] = 0 \\ \implies z_{1,n} &= \boldsymbol{w}_1^T\boldsymbol{x}_n. \end{aligned}$$

Hence the optimal score $z_{1,n}$ is the projection of the data onto the first principal component.

Plug this back in to get

$$\begin{aligned}
J(\boldsymbol{w}_1) &= \frac{1}{N}\sum_n \left[\boldsymbol{x}_n^T\boldsymbol{x}_n - 2z_{1,n}\boldsymbol{w}_1^T\boldsymbol{x}_n + z_{1,n}^2\right] \\
&= \frac{1}{N}\sum_n \left[\boldsymbol{x}_n^T\boldsymbol{x}_n - 2z_{1,n}^2 + z_{1,n}^2\right] \\
&= \text{const.} - \frac{1}{N}\sum_n z_{1,n}^2 \\
&= \text{const.} - \text{var}[\boldsymbol{z}_1],
\end{aligned}$$

where the last line follows because

$$\begin{aligned}
\mathbb{E}[\boldsymbol{z}_1] &= \mathbb{E}[\boldsymbol{X}^T\boldsymbol{w}] \\
&= \mathbb{E}[\boldsymbol{X}^T]\boldsymbol{w} \\
&= 0.
\end{aligned}$$

Note: It is tempting to make the following substitution

$$\begin{aligned}
J(\boldsymbol{w}_1) &= \frac{1}{N}\sum_n \left[\boldsymbol{x}_n^T\boldsymbol{x}_n - 2z_{1,n}\boldsymbol{w}_1^T\boldsymbol{x}_n + z_{1,n}^2\right] \\
&= \frac{1}{N}\sum_n \left[\boldsymbol{x}_n^T\boldsymbol{x}_n - 2(\boldsymbol{w}_1^T\boldsymbol{x}_n)(\boldsymbol{w}_1^T\boldsymbol{x}_n) + z_{1,n}^2\right] \\
&= \frac{1}{N}\sum_n \left[\boldsymbol{x}_n^T\boldsymbol{x}_n - 2\boldsymbol{x}_n^T(\boldsymbol{w}_1\boldsymbol{w}_1^T)\boldsymbol{x}_n) + z_{1,n}^2\right],
\end{aligned}$$

and then claim that $\boldsymbol{w}_1\boldsymbol{w}_1^T = 1$ since $\boldsymbol{w}$ has unit norm, *but this is false!*. Instead, $\boldsymbol{w}_1\boldsymbol{w}_1^T = \tilde{\boldsymbol{W}}$ is a rank-one matrix with entries $\tilde{W}_{i,j} = w_i w_j$.

Returning to our objective function, we see that minimizing $J(\boldsymbol{w}_1)$ yields a first principal component $\boldsymbol{w}_1$in the direction that maximizes the variance of the projected data. What is this variance? Plugging back in for $\boldsymbol{z}_n$ yields

$$\text{var}[\boldsymbol{z}_1] = \frac{1}{N}\sum_n \boldsymbol{w}_1^T\boldsymbol{x}_n\boldsymbol{x}_n^T\boldsymbol{w}_1 = \boldsymbol{w}_1^T\boldsymbol{\Sigma}\boldsymbol{w}_1,$$

where $\boldsymbol{\Sigma}$ is the empirical covariance matrix. If the magnitude of $\boldsymbol{w}_1$ is unconstrained then the variance is trivially maximized when $||\boldsymbol{w}_1|| \to \infty$, but by the orthonormality of $\boldsymbol{W}$ we have the constraint $||\boldsymbol{w}_1|| = 1$. This can be added to our objective function with a Lagrange multiplier

$$J(\boldsymbol{w}_1) = \boldsymbol{w}_1^T\boldsymbol{\Sigma}\boldsymbol{w}_1 + \lambda_1(\boldsymbol{w}_1^T\boldsymbol{w}_1 - 1).$$

Taking partial derivatives and setting to zero yields

$$\Sigma\boldsymbol{w}_1 = \lambda_1\boldsymbol{w}_1.$$

We see that the optimal $\boldsymbol{w}_1$ is an eigenvector of the covariance matrix. Left multiplying by $\boldsymbol{w}_1^T$ we see that the variance of the projected data is the eigenvalue corresponding to $\boldsymbol{w}_1$, so to maximize the variance we choose the eigenvector with the largest λ_1.

(c) Generative model approach

It would be nice if we could formulate PCA as a generative model and thereby glean some intuition for why the eigenvectors of the empirical covariance matrix are good principal components. If we were to invert the process described above, we would first sample projections $\boldsymbol{z}_n$ and then transform them with $\boldsymbol{W}$ in order to get the corresponding observed data $\boldsymbol{x}_n$. There was no noise in the reconstruction error formulation; the error only stemmed from the fact that we had fewer principal components than dimensions ($L < D$). Hence, in the generative model, $\boldsymbol{x}_n$ would be deterministic given $\boldsymbol{z}_n$ and $\boldsymbol{W}$.

We can relax this assumption a bit and add isotropic Gaussian noise such that

$$\boldsymbol{x}_n \sim \mathcal{N}(\boldsymbol{W}\boldsymbol{z}_n, \sigma^2 \boldsymbol{I}),$$

where $\boldsymbol{W}$ is still constrained to be orthogonal and where we are ignoring mean shifts for simplicity. If we assume

$$\boldsymbol{z}_n \sim \mathcal{N}(0, \boldsymbol{I}),$$

then we have a model known as **probabilistic PCA**. It turns out that in the limit of $\sigma^2 \to 0$ the MLE estimate of $\boldsymbol{W}$ and $\boldsymbol{z}_n$ recovers the classical PCA solution.

2. Factor Analysis

Factor analysis (FA) is another dimensionality reduction technique with a long history in statistics, psychology, and other fields. It turns out that both PCA and FA can be viewed as special cases of the generative model described above. In factor analysis, however, we have the following model:

$$\begin{aligned} \boldsymbol{z}_n &\sim \mathcal{N}(0, \boldsymbol{I}), \\ \boldsymbol{x}_n \mid \boldsymbol{z}_n &\sim \mathcal{N}(\boldsymbol{W}\boldsymbol{z}_n, \boldsymbol{\Psi}), \end{aligned}$$

where $\boldsymbol{\Psi}$ is restricted to be diagonal. This is just like probabilistic PCA except that here we allow different dimensions of our observations to have different variances. Instead of principal components, we will call the $\boldsymbol{z}_n$'s latent **factors**. The model is best motivated with an example.

Suppose we have a set of of ratings vectors $\{\boldsymbol{x}_n\}_{n=1}^N$ that N users gave to each of D jokes. Our goal is to predict a user's rating on a joke she hasn't heard, based on her ratings for other jokes and other users' ratings for the new joke. This is the **collaborative filtering** problem. We will assume that each user has a latent vector of preferences $\boldsymbol{z}_n$ along L dimensions, e.g. "length of the joke," "knock-knock-iness," or "years of math required to understand the joke." Each joke will have a weighting of these dimensions, $\boldsymbol{w}_d \in \mathbb{R}^L$. Users' ratings are weighted sums, $\boldsymbol{w}_d^T \boldsymbol{z}_n$, of how well the joke aligns to their latent preferences, plus noise. If we knew the preference dimensions, weights, and user preferences, we could easily describe the distribution over joke ratings in terms of additive noise, but otherwise we have to model the potentially complex distribution over ratings directly.

Take the simplest case where

$$\boldsymbol{x}_n = \boldsymbol{W}\boldsymbol{z}_n + \boldsymbol{\mu} + \text{noise},$$

i.e. $\boldsymbol{x}$ is a linear function of $\boldsymbol{z}$, with $\boldsymbol{W} \in \mathbb{R}^{D \times L}$. Furthermore, assume that both the $\boldsymbol{z}$'s and the noise follow multivariate Gaussian distributions.

$$\begin{aligned} \boldsymbol{z}_n &\sim \mathcal{N}(\boldsymbol{z}_n \,|\, \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \\ \boldsymbol{x}_n | \boldsymbol{z}_n &\sim \mathcal{N}(\boldsymbol{W}\boldsymbol{z}_n + \boldsymbol{\mu}, \Psi). \end{aligned}$$

Since both are Gaussian, their joint distribution will be Gaussian and the marginal distribution over $\boldsymbol{x}$ will be Gaussian.

Our goal is to explain the data in terms of latent variables $\boldsymbol{z}$. In the Gaussian case this means explaining covariance in $\boldsymbol{x}$ in terms of covariance in $\boldsymbol{z}$, rather than in terms of noise. Hence we restrict Ψ to be diagonal.

$$\boldsymbol{\Psi} = \operatorname{diag}\left(\left[\sigma_{11}^2, \quad \ldots, \quad \sigma_{DD}^2\right]\right).$$

Under this restriction, the model is specified by

$$\begin{aligned} \boldsymbol{\theta} &= \{\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0, \boldsymbol{W}, \boldsymbol{\mu}, \boldsymbol{\Psi}\} \\ &= L + \frac{L(L+1)}{2} + LD + D + D \text{ parameters.} \end{aligned}$$

This is actually still overparameterized because we can take $\boldsymbol{\mu}_0 = 0$ and $\boldsymbol{\Sigma}_0 = \boldsymbol{I}$ without loss of generality by pushing them into $\boldsymbol{\mu}$ and $\boldsymbol{W}$, respectively. This leaves us with:

$$\begin{aligned} \boldsymbol{\theta} &= \{\boldsymbol{W}, \boldsymbol{\mu}, \boldsymbol{\Psi}\} \\ &= LD + D + D = O(LD) \text{ parameters.} \end{aligned}$$

With a standard trick of augmenting our factors with a constant we can roll the $\boldsymbol{\mu}$ into $\boldsymbol{W}$. By these arguments we have arrived at the generative model for factor analysis posited above.

(a) Learning
In order to learn the parameters $\boldsymbol{W}$ and $\boldsymbol{\Psi}$ and infer the latent factors $\boldsymbol{z}_n$ of the model, we make use of the EM algorithm. In the E step we update the latent factors given the current weights and noise matrices, and in the M step we set the weights and noise matrices to their MAP estimates under the current factors.
One benefit of the EM algorithm is that it is easy to handle missing data. We simply estimate it during the E step.

(b) Unidentifiability
Just as in mixture models, the learned weight matrix is not uniquely defined. We can rotate by any orthogonormal matrix $\boldsymbol{R}$ and the likelihood will not change. Hence one must take care when interpreting the factors.