Linear Factor Models - Probabilistic PCA

Angus Scroggie

Abstract

Probabilistic Principal Component Analysis (PPCA) plays a pivotal role in probabilistic modelling, a topic at the forefront of modern ML/DL problems. PPCA provides a more powerful framework for analysis than conventional PCA, with key applications like dimensionality reduction, handling missing samples in datasets, mixture modelling, and computationally efficient parameter estimation. This tutorial provides a succinct overview of PPCA's applications to learning problems and its advantages over standard PCA, such as the formulation of an EM algorithm. We briefly recap PCA, then introduce latent variable models as a solution to the evolving demands of ML before delving into the details of PPCA.

Contents

1 Introduction

Probabilistic Principal Component Analysis (PPCA) serves as a crucial model at the crossroads between probabilistic modelling and dimensionality reduction. The model was independently proposed by Roweis (1997) [1] and Tipping & Bishop (1999) [3] with the key result of an Expectation-Maximization (EM) algorithm for estimating the principal subspace spanned by the principal components of PCA. It has since been expanded upon by Bishop in his landmark text *Pattern Recognition and Machine Learning (PRML)* (2006) [5], as well as by other authors [7] [11] [18]. This tutorial aims to succinctly explain PPCA's motivation, principles, applications, and practical implementations. A review of conventional Principal Component Analysis (PCA) is given in Appendix A, discussing its formulation, applications across various domains, and inherent constraints. Transitioning from PCA, this paper delves into latent variable modeling, establishing the foundation for PPCA within a linear-Gaussian framework. Through this probabilistic view, we explore essential concepts such as parameter estimation, discerning PPCA's relationship with traditional PCA, and acknowledging its strengths and drawbacks.

To support understanding, a coded example is provided, illustrating the application of the EM algorithm for PPCA. Additionally, we compare the projections of both PCA and PPCA from a 2D data space. We finish with a brief look into more advanced latent variable models. Exercises with solutions are included to test comprehension.

2 The Probabilistic PCA Model

2.1 Latent variable models

We first introduce latent variable models, a powerful class of probabilistic models that capture underlying, unobservable factors. In learning problems, where high-dimensional data can be noisy or redundant, latent variable models offer an approach to dimensionality reduction and feature extraction. The latent variable approach is motivated by the assumption that datasets lie close to a low-dimensional manifold embedded in the original data space [10] [14]. The manifold captures the underlying latent space distribution, where the deviations of data points are attributed to noise.

One prominent class of latent variable models is linear factor models, where observed variables are generated by linear combinations of unobserved latent factors plus noise. These models are of the form

$$\boldsymbol{x} = W\boldsymbol{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon} \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ is an observed data point in the d-dimensional original space; $\boldsymbol{z} \in \mathbb{R}^m$ is a latent variable in an m-dimensional latent space; $W \in \mathbb{R}^{d\times m}$ and $\boldsymbol{\mu} \in \mathbb{R}^d$ define the linear transformation of latent variables to original data; and $\boldsymbol{\epsilon}$ represents the noise. Note importantly that the columns of W span a linear subspace which corresponds to the principal subspace from PCA, which we will revisit.

A linear factor model describes the data generation process as follows. The latent variables are first sampled from some simple distribution, $\boldsymbol{z} \sim p(\boldsymbol{z})$. Then the observable values are sampled according to the linear transformation (1), where the noise term is typically Gaussian and diagonal.

2.2 Formulating PPCA

Following the recap of conventional PCA in Appendix A and an overview of linear factor models, we now delve into the probabilistic interpretation of PCA. We consider a linear-Gaussian framework for the linear factor model described in (1), such that the latent variable prior is the standard unit-covariance Gaussian distribution

$$p(\boldsymbol{z}) = N(\boldsymbol{z}; \boldsymbol{0}, I) \tag{2}$$

where the observed variables $\boldsymbol{x}$ are conditionally independent given $\boldsymbol{z}$. Furthermore, the noise is assumed to be drawn from an isotropic Gaussian $\boldsymbol{\epsilon} \sim N(\boldsymbol{0}, \sigma^2 I)$ for some scalar $\sigma^2 \geq 0$. Using properties of Gaussian distributions, we determine the conditional distribution

$$p(\boldsymbol{x}|\boldsymbol{z}) = N(\boldsymbol{x}; W\boldsymbol{z} + \boldsymbol{\mu}, \sigma^2 I) \tag{3}$$

which can be understood from the generative viewpoint in Figure 1.

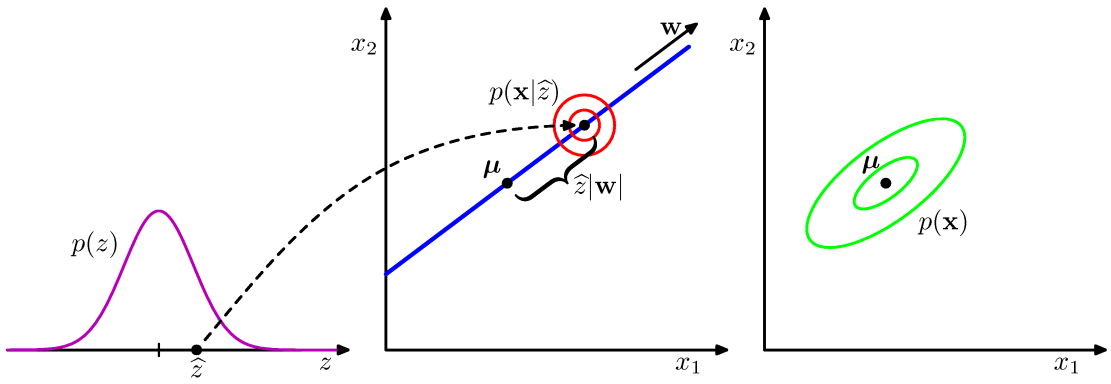

Figure 1: The generative view of PPCA for a 2D data space and 1D latent space. The latent variable $\hat{z}$ is drawn from the prior, then $\boldsymbol{x}$ is drawn according to (3). The rightmost plot show density contours for the marginal distribution $p(\boldsymbol{x})$. Figure is from Bishop (2006) [5].

To obtain the conventional understanding of PCA, we can derive the reverse mapping from data space to latent space using Bayes' theorem. The marginal distribution of $\boldsymbol{x}$ is given by

$$p(\boldsymbol{x}) = N(\boldsymbol{x}; \boldsymbol{\mu}, C) \tag{4}$$

where $C = WW^T + \sigma^2 I$, and the posterior distribution is given by

$$p(\boldsymbol{z} \mid \boldsymbol{x}) = N(\boldsymbol{z} \mid M^{-1}W^T(\boldsymbol{x} - \boldsymbol{\mu}), \sigma^{-2}M^{-1}) \tag{5}$$

for the $m \times m$ matrix $M = W^TW + \sigma^2 I$. See Exercises 1 and 2 for derivations of these results.

2.3 Maximum-likelihood for PPCA

With the PPCA model defined, we are now interested in estimating the model parameters W, $\boldsymbol{\mu}$, and σ^2. Maximum likelihood (ML) does provide a laborious exact solution to this problem, which we state below, however also has some weaknesses regarding computation. Derivation details can be found in [3]. Jumping ahead quickly, we will find the obvious estimate $\boldsymbol{\mu}_{\text{ML}} = \bar{\boldsymbol{x}}$. Given a dataset $X = \{\boldsymbol{x}_i\}$ of observed data, the log-likelihood function is given by

$$\ln p(X \mid \boldsymbol{\mu}, \boldsymbol{W}, \sigma^2) = -\frac{nd}{2}\ln(2\pi) - \frac{n}{2}\ln|C| - \frac{1}{2}\sum_{i=1}^{n}\boldsymbol{y}_i^T C^{-1}\boldsymbol{y}_i \tag{6}$$

$$= -\frac{n}{2}\left(d\ln(2\pi) + \ln|C| + \text{Tr}(C^{-1}S)\right) \tag{7}$$

where $Y = \{\boldsymbol{y}_i\}$ is the mean-centered data matrix, with $\boldsymbol{y}_i = \boldsymbol{x}_i - \boldsymbol{\mu}$. This yields the ML estimators,

$$\boldsymbol{\mu}_{\text{ML}} = \bar{\boldsymbol{x}} \tag{8}$$

$$W_{\text{ML}} = U_m(L_m - \sigma^2 I)^{\frac{1}{2}}R \tag{9}$$

$$\sigma^2_{\text{ML}} = \frac{1}{d-m}\sum_{i=m+1}^{d}\lambda_i \tag{10}$$

where the column vectors of U_m are the (ordered) m largest eigenvectors of S, L_m is a diagonal matrix of the corresponding eigenvalues λ_i, and R is an arbitrary $m \times m$ orthogonal matrix. Some interesting results follow from these solutions. The columns of W define the subspace spanned by the PCs (22) determined by normal PCA. The estimator σ^2_{ML} can be interpretted as the variance associated with the $d-m$ discarded dimensions. To comment on the invariance associated with R, this matrix corresponds to arbitrary rotations in latent space; substituting W_{ML} into C shows that the marginal density $p(\boldsymbol{x})$ is unaffected by such rotations [5].

Using the posterior mean with the ML estimates, we have the reverse mapping to latent space

$$\mathbb{E}\left[\boldsymbol{z} \mid \boldsymbol{x}\right] = M^{-1}W_{\text{ML}}^T(\boldsymbol{x} - \bar{\boldsymbol{x}}) \tag{11}$$

and hence the projection in the original space is $W\mathbb{E}[\boldsymbol{z} \mid \boldsymbol{x}] + \boldsymbol{\mu}$. Notably, in the limit as $\sigma^2 \to 0$, we have

$$\lim_{\sigma^2 \to 0} \mathbb{E}[\boldsymbol{z} \mid \boldsymbol{x}] = \left(W_{\mathrm{ML}}^T W_{\mathrm{ML}}\right)^{-1} W_{\mathrm{ML}}^T (\boldsymbol{x} - \bar{\boldsymbol{x}}) \tag{12}$$

which corresponds to an orthogonal projection of $\boldsymbol{x}$ onto the latent space, or in other words, the standard PCA model is recovered. Furthermore, the posterior covariance goes to zero and the density function becomes singular. In the general case $\sigma^2 > 0$, the projections are shifted towards the mean (or origin when the data matrix is mean-centered) [17], as we will see in the example section.

2.4 EM algorithm for PPCA

Despite the benefits of an exact solution, there are some drawbacks to the ML estimators for PPCA. It can be shown that the degrees of freedom, or number of independent parameters, for the ML solution is

$$dm + 1 - m(m-1)/2 \tag{13}$$

which is better than the general Gaussian distribution (i.e., $m = d - 1$) as it only scales linearly in d. However, for high-dimensional datasets, it may be computationally beneficial to consider an EM algorithm for iterative parameter estimates.

The EM algorithm for PPCA requires the complete-data log-likelihood,

$$\ln p(X, Z \mid \boldsymbol{\mu}, W, \sigma^2) = \sum_{i=1}^{n} \{\ln p(\boldsymbol{x}_i \mid \boldsymbol{z}_i) + \ln p(\boldsymbol{z}_i)\} \tag{14}$$

We use the previous result $\boldsymbol{\mu} = \bar{\boldsymbol{x}}$ from the ML solution because it is simple to determine and convenient when estimating the other parameters. Using (2) and (3) and taking the expectation, the E-step requires evaluating the following expectation expressions using the current parameter estimates (say on the t-th iterate),

$$\mathbb{E}[\boldsymbol{z}_i] = M^{-1} W^T (\boldsymbol{x}_i - \bar{\boldsymbol{x}}) \tag{15}$$

$$\mathbb{E}\left[\boldsymbol{z}_i \boldsymbol{z}_i^T\right] = \sigma^2 M^{-1} + \mathbb{E}[\boldsymbol{z}_i]\,\mathbb{E}[\boldsymbol{z}_i]^T \tag{16}$$

The M-step requires maximizing w.r.t. W and σ^2, yielding the following updated iterates (say for the $(t+1)$-th iterate),

$$W' = \left[\sum_{i=1}^{n} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}) \mathbb{E}[\boldsymbol{z}_i]^T\right] \left[\sum_{i=1}^{n} \mathbb{E}\left[\boldsymbol{z}_i \boldsymbol{z}_i^T\right]\right]^{-1} \tag{17}$$

$$(\sigma^2)' = \frac{1}{nd} \sum_{i=1}^{n} \left\{ \|\boldsymbol{x}_i - \bar{\boldsymbol{x}}\|^2 - 2\mathbb{E}[\boldsymbol{z}_i]^T (W')^T (\boldsymbol{x}_i - \bar{\boldsymbol{x}}) + \mathrm{Tr}\left(\mathbb{E}\left[\boldsymbol{z}_i \boldsymbol{z}_i^T\right] (W')^T W'\right) \right\} \tag{18}$$

See Exercise 4 for a derivation. Computational benefits of this EM algorithm are discussed in the next section. A useful application is that an EM framework allows the PPCA model to deal with omitted data when they are missing at random (MAR) by marginalizing over the missing variables, using the complete- and incomplete-data log-likelihoods [1] [3] [6] [8]. Additionally, we can still take the limit $\sigma^2 \to 0$ to obtain standard PCA and retain a valid EM algorithm, and a relatively simple one at that [1] [5]. For a Variational Bayes approach to iterative parameter estimation, see [11] [12].

2.5 Advantages and disadvantages

Probabilistic PCA has several advantages over standard PCA:

- EM algorithm is computationally efficient over an exact solution when $m \ll d$. Specifically, standard PCA requires computing an $O(d^3)$ or $O(md^2)$ eigendecomposition and a $O(nd^2)$ or $O(n^3)$ sample covariance matrix. PPCA-EM requires sums over the data matrix which are $O(ndm)$, which can support the algorithm's iterative nature [5].

- EM algorithm can handle missing values and mixture models [1] [3] [8].
- EM produces a valid limiting case for standard PCA [1] [5].
- Conventional PCA orthogonally projects data points close to the subspace even if they are outliers; PPCA projections are robust to noise and shifted towards the mean [5] [17].
- Probabilistic formulation gives rise to a Bayesian treatment (discussed later) [2] [5].
- The model can be used generatively to produce samples from the distribution [13] [22] [23].

There are some weaknesses of the model, however. Most notable is that the linear-Gaussian assumption is restrictive for more complex data; the isotropic noise assumption may also be limiting [17]. Of course, dimension reduction risks information loss due to an overly-simple noise model. This motivates approaches like non-isotropic noise (factor analysis), nonlinear PCA (kernel PCA), non-Gaussian latent models (ICA), and nonlinear relationships between latent and observed variables. These are briefly discussed later.

3 Practical Example

3.1 A sample dataset using Python

For this example, we will implement the EM algorithm for PPCA, then compare the results to conventional PCA. First, we need a dataset. We will simulate $n = 40$ samples from a multivariate normal distribution with $d = 2$ features, with the aim of projecting to a latent space with dimension $m = 1$.

```
import numpy as np
from sklearn.decomposition import PCA

## Construct dataset
rng = np.random.default_rng(seed=88)
X = rng.multivariate_normal((2, 1), [[5, 3], [3, 4]], size=40)
```

3.2 Implementation

The first demonstration is an implementation of the EM algorithm for PPCA specified by (14)-(18). Figure 2 shows that the log-likelihood converges over the algorithm iterations. More advanced implementations can take advantage of matrix operations like Cholesky decompositions and triangular matrix operations [9] [13].

```
def ppca_em(X, m=2, s=1.0, maxiter=20, tol=1e-6):
    """Perform EM algorithm to iteratively maximize likelihood of PPCA model."""
    d, n = X.shape
    mu = np.mean(X, axis=1, keepdims=True)
    X = X - mu # Mean centering

    # EM estimate initialization
    llh = np.zeros(maxiter)
    llh[0] = -np.inf
    W = np.random.randn(d, m)
    R = X.dot(X.T) / n

    for iter in range(1, maxiter):
        M = W.T.dot(W) + s * np.eye(m)
        U = np.linalg.inv(M.dot(W.T).dot(R).dot(W) + s * np.eye(m))

        # Likelihood computations
        C = W.dot(W.T) + s * np.eye(m)
```

```
        llh[iter] = -n * (d * np.log(2 * np.pi) + np.log(np.linalg.det(C)) + np.trace(
    C)) / 2

        if abs(llh[iter] - llh[iter - 1]) < tol * abs(llh[iter - 1]):
            break

        # Update parameters
        W = R.dot(W).dot(U)
        s = np.trace(R - R.dot(W).dot(M).dot(W.T)) / d

    return W, mu, s, llh[1:iter]
```

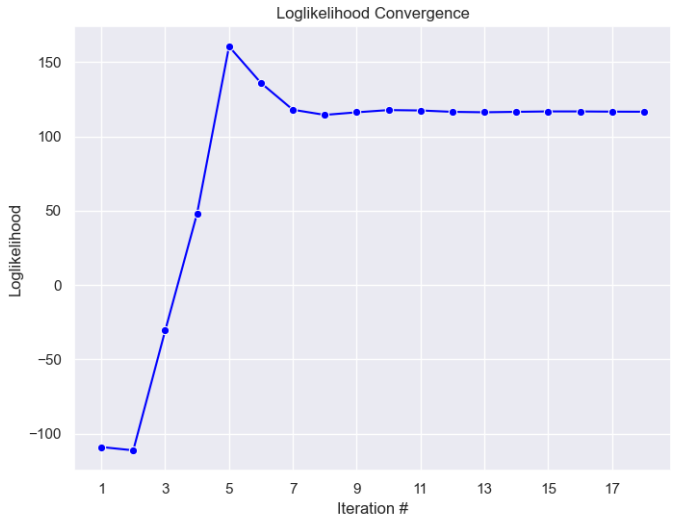

Figure 2: The loglikelihood over the EM algorithm iterations shows convergence as expected.

We now compare the projection in the data space determined by both conventional PCA and PPCA-EM, shown in Figure 3. The plotting code is omitted for brevity.

```
# Using sklearn's PCA object
pca = PCA(n_components=1, random_state=42)
X_r = pca.fit_transform(X)
X_pca_projects = pca.inverse_transform(X_r)

# Plot projections via probabilistic PCA
W_hat, mu, sigma2_hat, llhs = ppca_em(X.T, m=1, s=0.2)
M_hat = W_hat.T @ W_hat + sigma2_hat * np.eye(m)
Z = np.linalg.solve(M_hat, W_hat.T.dot((X.T - mu)))
X_ppca_projects = W_hat @ Z + mu
```

4 Further Applications

4.1 A Bayesian perspective

Often the choice of latent dimensionality m is not obvious. For instance, visualization applications would typically choose a projection onto $m = 2$, but choices are less clear in other cases. Solutions to this problem for standard PCA are usually rudimentary exploratory data analysis (EDA) based on arbitrary cutoffs, for example, using the Kaiser criterion on a scree plot [19]. Given the probabilistic formulation of PCA, a Bayesian approach appears reasonable for model selection by marginalizing out parameters w.r.t. the priors. Bishop describes this approach for both a prior only over W [5] and a full Bayesian treatment using variational methods [2].

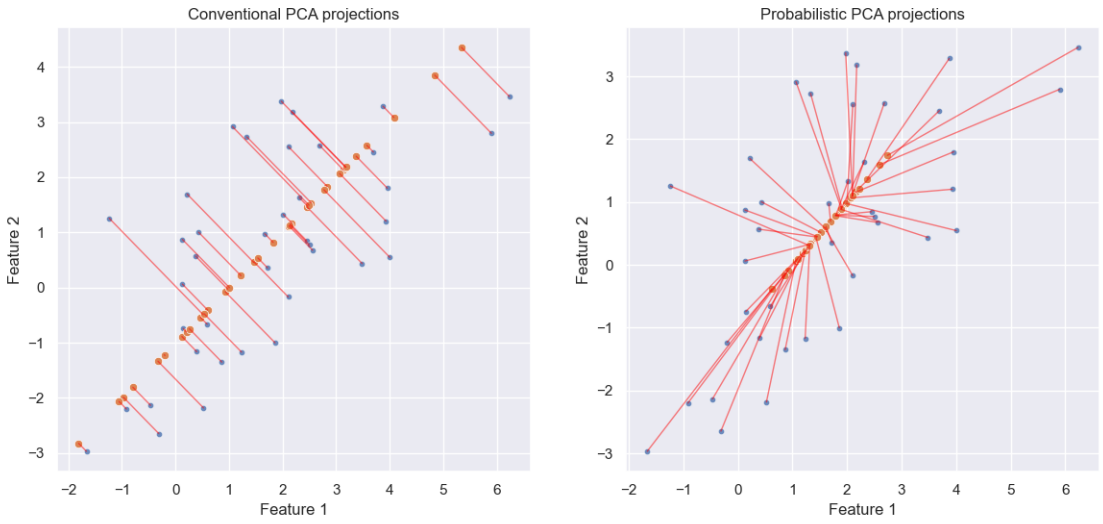

Figure 3: A comparison of the projections from the two PCA methods. We can see that conventional PCA orthogonally projects onto the embedded subspace, whereas PPCA has a scaling effect that brings the projections towards the mean of the distribution.

4.2 Non-Gaussian and nonlinear models

There are *several* generalizations of latent variable models and in particular linear factor models. One such example is Factor Analysis (FA). FA differs from PPCA in the treatment of the noise term $\boldsymbol{\epsilon}$, where it assumes a diagonal covariance rather than isotropic covariance. This changes (3) to

$$p(\boldsymbol{x} \mid \boldsymbol{z}) = N(\boldsymbol{x}; W\boldsymbol{z} + \boldsymbol{\mu}, \boldsymbol{\Psi}) \tag{19}$$

where $\boldsymbol{\Psi} \in \mathbb{R}^{d \times d}$ is a diagonal matrix with elements $\sigma_1^2, ..., \sigma_d^2$ for per-variable variances. The latent variables hence capture dependencies between the observed $\boldsymbol{x}_i$. An EM algorithm can still be derived for FA; a very useful result in the absence of an exact ML solution.

Some examples of models that go beyond the linear-Gaussian framework are independent component analysis (ICA) and autoassociative neural networks, such as the variational autoencoder (VAE) and Hopfield network. These models can retrieve data from memory given only a partial sample of that data, making them effective at de-noising input. The topic of manifold learning also arises in the general case. Further reading can be found in [5] [10] [18].

5 Exercises

Solutions can be found in Appendix B.

1. Derive the result (4) for the marginal distribution $p(\boldsymbol{x})$ in the PPCA model.
2. Derive the result (5) for the posterior distribution $p(\boldsymbol{z} \mid \boldsymbol{x})$ in the PPCA model.
3. Show that replacing the latent variable prior (2) for the PPCA model with a generic Gaussian distribution, $N(\boldsymbol{z}; \boldsymbol{\lambda}, \Sigma)$, produces an identical result to (4) for the marginal distribution $p(\boldsymbol{x})$, for any choice of $\boldsymbol{\lambda} \in \mathbb{R}^m$ and $\Sigma \in \mathbb{R}^{m \times m}$.
4. Derive the M-step equations (17) and (18) by maximizing the expectation of the complete-data log-likelihood (14) for the PPCA model.

References

[1] Sam Roweis. "EM algorithms for PCA and SPCA". In: *Proceedings of the 10th International Conference on Neural Information Processing Systems*. NIPS'97. Denver, CO: MIT Press, 1997, pp. 626–632.

[2] C. M. Bishop. "Variational principal components". In: vol. 1. IEE, 1999, pp. 509–514. URL: https://api.semanticscholar.org/CorpusID:13110506.

[3] Michael E. Tipping and Christopher M. Bishop. "Probabilistic Principal Component Analysis". In: *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 61.3 (1999), pp. 611–622. ISSN: 13697412, 14679868. URL: https://academic.oup.com/jrsssb/article/61/3/611/7083217?login=false.

[4] George Casella and Roger L. Berger. *Statistical Inference*. Springer, 2002.

[5] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. URL: https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/.

[6] Geoffery J. McLachlan and Thriyambakam Krishnan. *The EM Algorithm and Extensions*. 2nd. Wiley, 2008.

[7] Yue Guan and Jennifer Dy. "Sparse Probabilistic Principal Component Analysis". In: *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*. Ed. by David van Dyk and Max Welling. Vol. 5. Proceedings of Machine Learning Research. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 2009, pp. 185–192. URL: https://proceedings.mlr.press/v5/guan09a.html.

[8] Lingbo Yu et al. "Probabilistic Principal Component Analysis with Expectation Maximization (PPCA-EM) Facilitates Volume Classification and Estimates the Missing Data". In: *Journal of Structural Biology* 171.1 (2010), pp. 18–30. DOI: 10.1016/j.jsb.2010.04.002.

[9] C. Angermueller. *ppca*. GitHub repository. 2014. URL: https://github.com/cangermueller/ppca.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[11] Simon Rogers and Mark Girolami. *A First Course in Machine Learning*. 2nd. Chapman and Hall, 2016.

[12] Yunfeng Zhu. *ppca*. GitHub repository. 2017. URL: https://github.com/ymcdull/ppca.

[13] Mo Chen. *PRMLT*. GitHub repository. 2020. URL: https://github.com/PRML/PRMLT.

[14] DLVU (Deep Learning at VU Amsterdam). *Lecture 6.2: Probabilistic PCA*. YouTube video. 2020. URL: https://www.youtube.com/watch?v=BTUehwU_5Uo.

[15] Benyamin Ghojogh et al. "Factor Analysis, Probabilistic Principal Component Analysis, Variational Inference, and Variational Autoencoder: Tutorial and Survey". In: *ArXiv* abs/2101.00734 (2021). URL: https://api.semanticscholar.org/CorpusID:230435787.

[16] Trevor Hastie et al. *An Introduction to Statistical Learning*. Springer, 2021.

[17] Mutual Information. *Factor Analysis and Probabilistic PCA*. YouTube video. 2021. URL: https://www.youtube.com/watch?v=lJ0cXPoEozg.

[18] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. URL: probml.ai.

[19] Wikipedia contributors. *Scree plot*. 2023. URL: https://en.wikipedia.org/wiki/Scree_plot.

[20] Edward contributors. *Probabilistic PCA*. Online tutorial. 2024. URL: https://edwardlib.org/tutorials/probabilistic-pca.

[21] sci-kit learn contributors. *2.5. Decomposing signals in components (matrix factorization problems)*. Online tutorial. 2024. URL: https://scikit-learn.org/stable/modules/decomposition.html.

[22] Vivek Sivaramakrishnan. *Probabilistic PCA*. Online tutorial. 2024. URL: https://bsc-iitm.github.io/ML_Handbook/pages/Probabilistic_PCA.html.

[23] TensorFlow contributors. *Probabilistic PCA*. Online tutorial. 2024. URL: https://www.tensorflow.org/probability/examples/Probabilistic_PCA.

Appendix A - A Brief Recap of Principal Component Analysis

Principal component analysis (PCA) is a well-known technique with applications to learning problems, such as dimensionality reduction, feature extraction, and data visualization. There are two common formulations for the algorithm: variance maximization and error minimization. Consider the $n \times d$ data matrix $\boldsymbol{X}_{\text{all}}$ where each observation $\boldsymbol{x}_i$ has dimension d for $i = 1, ..., n$. The goal of PCA is to project the data onto a subspace with dimension $m < d$.

The first formulation achieves this by variance maximization of the projected data. We define the first principal component (PC) of the features $X_1, ..., X_d$ as the linear combination,

$$Z_1 = \phi_{11} X_1 + ... + \phi_{d1} X_d = \boldsymbol{\phi}_1^T \boldsymbol{X} \tag{20}$$

which has the largest variance, subject to the normalizing constraint $\boldsymbol{\phi}_1^T \boldsymbol{\phi}_1 = 1$ [16]. The vector $\boldsymbol{\phi}_1$ is called the principal component loading vector. The normalization condition is necessary to prevent arbitrarily large loadings. In practice, calculating the first PC requires maximizing the sample variance,

$$S = \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T \tag{21}$$

This gives the following optimization problem,

$$\underset{\boldsymbol{\phi}_1 \in \mathbb{R}^d}{\arg\max} \left\{ \boldsymbol{\phi}_1^T S \boldsymbol{\phi}_1 \right\} \quad \text{s.t.} \quad \boldsymbol{\phi}_1^T \boldsymbol{\phi}_1 = 1 \tag{22}$$

By introducing a Lagrange multiplier, it can be shown that (22) is solved by eigendecomposition. In particular, the first PC is given by the largest eigenvector of S. Additional PCs can be found by maximizing the uncorrelated projected variance, equivalent to the projection orthogonal to the existing PCs. We find that the optimal linear projection to an m-dimensional subspace is therefore given by the m largest eigenvectors $\lambda_1, ..., \lambda_m$. An alternative formulation is based on minimizing the projection error, quantified by the distortion measure

$$J = \sum_{i=m+1}^{d} \boldsymbol{u}_i^T S \boldsymbol{u}_i \tag{23}$$

and constrained by orthonormality conditions on the basis vectors $\boldsymbol{u}_i$, $i = 1, ..., d$. This problem ultimately yields the same result as (22) [5].

An obvious drawback of PCA is that a naive eigendecomposition has cost $O(d^3)$[1], which quickly gets expensive for large d. Calculating the sample covariance matrix is similarly expensive. One of the appeals of PCA is in learning problems, where feature extraction allows drastically improved computation time at the cost of small accuracy loss. However, ideally the computation time of dataset preprocessing does not increase much. In the tutorial paper we see that PCA can be expressed as the maximum-likelihood solution of a specific latent variable model, which yields a computationally efficient EM algorithm for parameter estimation. There is a special case of the PPCA-EM algorithm which gives an iterative way to compute the conventional principal components.

[1] Clever approaches, such as the power method, can achieve $O(md^2)$.

Appendix B - Solutions to Exercises

1. Let's quickly recap the information we have from the linear factor model set-up. There is a linear relationship (1) between $\boldsymbol{x}$ and $\boldsymbol{z}$ given by $\boldsymbol{x} = W\boldsymbol{z} + \boldsymbol{\mu} + \boldsymbol{\sigma^2 I}$. We also have the latent variable prior (2) and the conditional distribution (3). We therefore want to solve for the marginal distribution by

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x} \mid \boldsymbol{z}) p(\boldsymbol{z}) \, d\boldsymbol{z} \tag{24}$$

There is a nicer way to find the marginal distribution given we are working in a linear-Gaussian framework and hence the marginal will also be Gaussian. Namely, we can use the law of total expectation and covariance, which state, respectively,

$$\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X \mid Y]] \tag{25}$$

$$\operatorname{Cov}(X, Y) = \mathbb{E}[\operatorname{Cov}(X, Y \mid Z)] + \operatorname{Cov}(\mathbb{E}[X, Y \mid Z]) \tag{26}$$

As such, the expectation will be given by

$$\begin{aligned} \mathbb{E}[\boldsymbol{x}] &= \mathbb{E}[\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{z}]] \\ &= \mathbb{E}[W\boldsymbol{z} + \boldsymbol{\mu}] \\ &= W\mathbb{E}[\boldsymbol{z}] + \boldsymbol{\mu} \\ &= \boldsymbol{\mu} \end{aligned}$$

and the covariance by

$$\begin{aligned} \operatorname{Cov}(\boldsymbol{x}) &= \mathbb{E}[\operatorname{Cov}(\boldsymbol{x} \mid \boldsymbol{z})] + \operatorname{Cov}(\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{z}]) \\ &= \mathbb{E}\left[\sigma^2 I\right] + \operatorname{Cov}(W\boldsymbol{z} + \boldsymbol{\mu}) \\ &= \sigma^2 I + \mathbb{E}\left[(W\boldsymbol{z} + \boldsymbol{\mu} - \mathbb{E}[W\boldsymbol{z} + \boldsymbol{\mu}])(W\boldsymbol{z} + \boldsymbol{\mu} - \mathbb{E}[W\boldsymbol{z} + \boldsymbol{\mu}])^T\right] \\ &= \sigma^2 I + \mathbb{E}\left[(W\boldsymbol{z})(W\boldsymbol{z})^T\right] \\ &= \sigma^2 I + \mathbb{E}\left[W\boldsymbol{z}\boldsymbol{z}^T W^T\right] \\ &= \sigma^2 I + W \operatorname{Cov}(\boldsymbol{z}\boldsymbol{z}^T) W^T \\ &= WW^T + \sigma^2 I := C \end{aligned}$$

These results combine to give (4).

2. We require a result from Bishop [5] referred to as Bayes' theorem for Gaussian distributions. If we have

$$p(\boldsymbol{u}) = N(\boldsymbol{u}; \boldsymbol{\mu}, \Lambda^{-1}) \tag{27}$$

$$p(\boldsymbol{v} \mid \boldsymbol{u}) = N(\boldsymbol{v}; A\boldsymbol{x} + \boldsymbol{b}, L^{-1}) \tag{28}$$

then

$$p(\boldsymbol{v}) = N(\boldsymbol{v}; A\boldsymbol{\mu} + \boldsymbol{b}, L^{-1} + A\Lambda^{-1}A^T) \tag{29}$$

$$p(\boldsymbol{u} \mid \boldsymbol{v}) = N(\boldsymbol{u}; \Sigma(A^T L(\boldsymbol{v} - \boldsymbol{b}) + \Lambda\boldsymbol{\mu}), \Sigma) \tag{30}$$

where

$$\Sigma = (\Lambda + A^T L A)^{-1} \tag{31}$$

Using (30), we can match variables and hence determine

$$\mathbb{E}[\boldsymbol{z} \mid \boldsymbol{x}] = (I + \sigma^{-2} W^T W)^{-1} W^T \sigma^{-2} I(\boldsymbol{x} - \boldsymbol{\mu}) \tag{32}$$

$$\operatorname{Cov}(\boldsymbol{z} \mid \boldsymbol{x}) = (I + \sigma^{-2} W^T W)^{-1} \tag{33}$$

where $M = W^T W + \sigma^2 I$. These results combine to give (5). Furthermore, we can confirm the result of the previous problem using (29).

3. We now consider a general Gaussian distribution for the latent variables

$$p(\boldsymbol{z}) = N(\boldsymbol{z}; \boldsymbol{\lambda}, \Sigma) \tag{34}$$

Following the solution to Exercise 1 using (25) and (26), we obtain

$$\mathbb{E}[\boldsymbol{x}] = W\boldsymbol{\lambda} + \boldsymbol{\mu} \tag{35}$$

$$\text{Cov}(\boldsymbol{x}) = W\Sigma W^T + \sigma^2 I \tag{36}$$

and hence the marginal distribution is

$$p(\boldsymbol{x}) = N(\boldsymbol{x}; W\boldsymbol{\lambda} + \boldsymbol{\mu}, W\Sigma W^T + \sigma^2 I) \tag{37}$$

Now defining

$$\tilde{\mu} = W\boldsymbol{\lambda} + \boldsymbol{\mu} \tag{38}$$

$$\tilde{W} = W\Sigma^{\frac{1}{2}} \tag{39}$$

and therefore we obtain a marginal distribution of the expected form

$$p(\boldsymbol{x}) = N(\boldsymbol{x}; \tilde{\mu}, \tilde{W}\tilde{W}^T + \sigma^2 I) \tag{40}$$

Thus, the marginal takes the same form given any generic Gaussian latent prior. We then choose a zero-mean unit-covariance latent prior for convenience.

4. See Appendix B of [3].