Machine Learning Notes

Matt Whiteway
Columbia University
m.whiteway@columbia.edu

Contents

Lecture 1

Latent Variable Models

In a previous lecture we saw models of the form $\mathbf{Y} = f_\theta(\mathbf{X})$, where both $\mathbf{Y}$ and $\mathbf{X}$ are observed. Maximum likelihood estimation (as well as maximum a posteriori estimation) is an important method in modern statistics for estimating the parameters θ of such models. But there are many instances where a set of predictors $\mathbf{X}$ is not available; perhaps these predictors are not observed, or not even clearly defined. However, in many cases we would still like to model the distribution of $\mathbf{Y}$.

Modeling the distribution of $\mathbf{Y}$ can be quite complicated for real data, especially if it is high-dimensional. One approach that has found wide success and applicability across many disciplines is to assume that the data $\mathbf{Y}$ can be described by a lower-dimensional set of variables $\mathbf{Z}$, where for each observation $\mathbf{y}_n \in \mathbb{R}^D$ there is a corresponding $\mathbf{z}_n \in \mathbb{R}^K$ such that $K << D$. We call the $\mathbf{Z}$ "latent variables" (LVs) because they are assumed to be unobserved (the notation $\mathbf{Z}$ denotes unobserved LVs while $\mathbf{X}$ denotes observed predictors).

Let us first consider a simple linear Gaussian example, where the data points are i.i.d (later we'll tackle the case where correlations exist between data points). The generative model of the data is defined as

$$\mathbf{y}_n = C\mathbf{z}_n + \epsilon_n, \tag{1.1}$$

where $C \in \mathbb{R}^{D \times K}$ is a matrix mapping from the latent space to the observation space; $\mathbf{z}_n \sim \mathcal{N}(\mathbf{z}|\mathbf{0}, Q)$; and $\epsilon_n \sim \mathcal{N}(\epsilon|\mathbf{0}, R)$ is the noise term in the observation space (we'll assume for now that the data is mean-centered and omit a bias term to keep the exposition cleaner). What type of statistical structure does this model enforce on the data? If we calculate the mean and variance of the observations we find

$$\begin{aligned} \mathbb{E}[\mathbf{y}_n] &= 0 && (1.2) \\ \mathrm{Cov}[\mathbf{y}_n] &= CQC^\intercal + R. && (1.3) \end{aligned}$$

Because both the LVs and the observation noise are Gaussian, the resulting $\mathbf{y}_n$ will also be Gaussian, with a mean and covariance given by the above equations. We will now make two more assumptions: first, without loss of generality, we will set Q to be equal to the identity matrix[1]; second, we will constrain R to be diagonal. We can see from equation (1.3) that the resulting model, which is referred to as Factor Analysis (FA), models the covariance matrix of a high-dimensional Gaussian as a low-rank matrix plus a diagonal matrix (where the rank, equal to the number of latent variables, is a hyperparameter of the model). Later on in this chapter we'll see how FA relates to the well-known Principal Component Analysis (PCA).

LV models are popular statistical models of high-dimensional data precisely because they are able to simplify high-dimensional joint probability distributions by combining much simpler distributions; in FA for example, the potentially complex pattern of correlations between different observation dimensions is modeled by the LVs, and each dimension is independent of the others when conditioned on the LVs.

The natural question that now arises is: how do we actually fit LV models to data? If the $\mathbf{Z}$ were observed, then this problem would reduce to the same ML estimation problem we saw with GLMs, since we could treat the $\mathbf{Z}$ as observed predictors. However, the $\mathbf{Z}$ are unobserved, and as a result model fitting is not as straightforward. The expectation-maximization (EM) algorithm [1] was developed as an iterative algorithm to perform ML estimation with such models. The EM algorithm is an intuitive one: we first make some guess about the LVs, then solve an ML estimation problem to update the model parameters given those LVs; then, given the new model parameters, update our guess about the LVs, and continue iterating until some convergence criterion is met. In the next section we'll delve deeper into the mechanics of the EM algorithm. The presentation will be somewhat general, but we will return to FA afterwards as an explicit example of the EM algorithm in action.

[1]Since Q is a covariance matrix, it is postive definite and can be diagonalized as $Q = VDV^\intercal$. Then, an equivalent model can be defined as $\mathbf{y}_i = WVD^{1/2}\mathbf{z}'_i + \epsilon_i$, where $\mathbf{z}'_i = \varepsilon_i$ and $\varepsilon_i \sim \mathcal{N}(0, I)$.

1.1 EM algorithm

To state the problem that the EM algorithm is going to solve more rigorously, let us assume that we have N samples of observed data $\{\mathbf{y}_n\}_{n=1}^N$ (denoted collectively as $\mathbf{Y}$). For each observation $\mathbf{y}_n$ we have a corresponding unobserved LV $\mathbf{z}_n$ (the collection of which is analagously referred to as $\mathbf{Z}$). Additionally, there is a set of model parameters θ that pertain to all observations. In the EM algorithm we make a distinction between the LVs - which are random variables, and hence we want to infer their distribution - and the parameters, for which we only make point estimates. In a fully Bayesian setting we would define distributions for all unobserved variables - both $\mathbf{Z}$ and θ - and hence there would be no distinction between the two. However, finding point estimates for θ makes the algorithm much simpler. Furthermore, while each LV is only based on a single data point, the parameters θ are based on all datapoints, and are therefore usually less noisy.

To use the EM algorithm we must define the likelihood function $p_\theta(\mathbf{Y}|\mathbf{Z})$ (where we make the distinction between parameters and random variables clear by using the subscript notation to refer to parameters), as well as a prior distribution on the LVs $p_\theta(\mathbf{Z})$ (which may or may not depend on parameters θ). As with the GLMs we will take logarithms of these various probability distributions to simplify calculations. With these LV models we can now make a distinction between the *complete* data log-likelihood, given by $\log p_\theta(\mathbf{Y}, \mathbf{Z})$, and the *observed* data log-likelihood, given by marginalizing the complete data log-likelihood over the unobserved LVs: $\log p_\theta(\mathbf{Y}) = \log \int p_\theta(\mathbf{Y}|\mathbf{Z}) p_\theta(\mathbf{Z}) d\mathbf{Z} \equiv \mathcal{L}(\theta)$. In order to find the ML estimate for the parameters, θ_{ML}, we maximize the observed data log-likelihood:

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}} \log p_\theta(\mathbf{Y}) = \log \int p_\theta(\mathbf{Y}|\mathbf{Z}) p_\theta(\mathbf{z}) d\mathbf{Z}. \tag{1.4}$$

This is now a difficult problem to solve for two reasons. First of all, the integral is high-dimensional, because we are integrating over *all* LVs $\mathbf{Z}$. Second, because the logarithm is *outside* of the integral, and we can't, for example, immediately take advantage of the distributions belonging to the exponential family. For a relatively simple model like FA there is in fact a closed-form solution for this optimization problem[2], but this is not the case for most LV models of interest. In fact, these types of integrals arise so frequently in Bayesian statistics that a vast literature has developed that focuses on techniques for approximating this type of integral. These techniques generally fall into two categories; the first uses sampling methods, such as Markov Chain Monte Carlo (MCMC), while the second finds deterministic approximations to the integral, a prototypical example of which is the EM algorithm.

Let us rewrite equation (1.4) in a slightly more suggestive way as

$$\theta_{ML} = \operatorname*{argmax}_{\theta} \log p_\theta(\mathbf{Y}) = \log \mathbb{E}_{p_\theta(\mathbf{z})}[p_\theta(\mathbf{Y}|\mathbf{Z})]. \tag{1.5}$$

This equation says that to find the ML estimate for θ we first take the expected value of the model likelihood with respect to the prior $p_\theta(\mathbf{Z})$, and then maximize the logarithm of the result. What if we instead approximated this process by choosing a different distribution $q(\mathbf{Z})$ and moved the expectation inside the logarithm to solve

$$\hat{\theta} = \operatorname*{argmax}_{\theta} \log p_\theta(\mathbf{Y}) = \log \mathbb{E}_{q(\mathbf{Z})}[p_\theta(\mathbf{Y}|\mathbf{Z})]? \tag{1.6}$$

Does this simplify our problem at all? We'll see shortly that we can find a distribution $q(\mathbf{Z})$ such that this expectation is easy to compute, and that there is a simple relationship between equations (1.5) and (1.6) that we can exploit to find an approximate ML solution.

This is exactly the approach used by the EM algorithm. For the first step (E-step) we are going to take the expectation of the complete data log-likelihood with respect to our current approximation for $q(\mathbf{Z})$; this gives us an expression that depends only on the observed data $\mathbf{Y}$ and the paramters θ, which we can then maximize with respect to θ (M-step). However, as we'll see, this updating of the parameters then necessitates an update of our approximation $q(\mathbf{Z})$, and so we repeat the E- and M-steps. One of the important properties of the EM algorithm is that each two-step iteration is actually guaranteed to monotonically increase the observed data log-likelihood, and so we iterate these steps until some convergence criterion has been met (e.g. the relative change in the observed data log-likelihood falls below some predefined threshold).

So how do we find the approximation $q(\mathbf{Z})$? We want to take the expectation of the complete data log-likelihood with respect to $q(\mathbf{Z})$, and then try to connect this to quantities that we know from specifying the model: the model log-likelihood $\log p_\theta(\mathbf{Y}|\mathbf{Z})$, the log-prior $\log p_\theta(\mathbf{Z})$, and the posterior $\log p_\theta(\mathbf{Z}|\mathbf{Y})$, as well as the quantity we don't know: the observed data log-likelihood $\log p_\theta(\mathbf{Y})$:

$$\begin{aligned}
\mathbb{E}_{q(\mathbf{Z})}[\log p_\theta(\mathbf{Y}|\mathbf{Z})] &= \int q(\mathbf{Z}) \log p_\theta(\mathbf{Y}|\mathbf{Z}) d\mathbf{Z} \\
&= \int q(\mathbf{Z}) \log \frac{p_\theta(\mathbf{Y}|\mathbf{Z}) p_\theta(\mathbf{Z})}{p_\theta(\mathbf{Z})} d\mathbf{Z} \\
&= \int q(\mathbf{Z}) \log p_\theta(\mathbf{Y}, \mathbf{Z}) d\mathbf{Z} - \int q(\mathbf{Z}) \log p_\theta(\mathbf{Z}) d\mathbf{Z} \\
&= \int q(\mathbf{Z}) \log p_\theta(\mathbf{Z}|\mathbf{Y}) p_\theta(\mathbf{Y}) d\mathbf{Z} - \int q(\mathbf{Z}) \log p_\theta(\mathbf{Z}) d\mathbf{Z} \\
&= \int q(\mathbf{Z}) \log p_\theta(\mathbf{Z}|\mathbf{Y}) d\mathbf{Z} + \int q(\mathbf{Z}) \log p_\theta(\mathbf{Y}) d\mathbf{Z} \\
&\quad - \int q(\mathbf{Z}) \log p_\theta(\mathbf{Z}) d\mathbf{Z} \\
&= \int q(\mathbf{Z}) \log \frac{p_\theta(\mathbf{Z}|\mathbf{Y}) q(\mathbf{Z})}{q(\mathbf{Z})} d\mathbf{Z} + \log p_\theta(\mathbf{Y}) \\
&\quad - \int q(\mathbf{Z}) \log p_\theta(\mathbf{Z}) d\mathbf{Z} \\
&= \int q(\mathbf{Z}) \log \frac{p_\theta(\mathbf{Z}|\mathbf{Y})}{q(\mathbf{Z})} d\mathbf{Z} + \int q(\mathbf{Z}) \log q(\mathbf{Z}) d\mathbf{Z} + \log p_\theta(\mathbf{Y}) \\
&\quad - \int q(\mathbf{Z}) \log p_\theta(\mathbf{Z}) d\mathbf{Z} \\
&= -\mathrm{KL}[q(\mathbf{Z})||p_\theta(\mathbf{Z}|\mathbf{Y})] + \mathrm{KL}[q(\mathbf{Z})||p_\theta(\mathbf{Z})] + \log p_\theta(\mathbf{Y}).
\end{aligned}$$

Rearranging this expression, we get

$$\log p_\theta(\mathbf{Y}) - \mathrm{KL}[q(\mathbf{Z})||p_\theta(\mathbf{Z}|\mathbf{Y})] = \mathbb{E}_{q(\mathbf{Z})}[\log p_\theta(\mathbf{Y}|\mathbf{Z})] - \mathrm{KL}[q(\mathbf{Z})||p_\theta(\mathbf{Z})]. \quad (1.7)$$

Equation (1.7) is the celebrated *Evidence Lower Bound*, or ELBO for short. Let us take a moment to study this expression. On the left hand side we have the observed data log-likelihood $\log p_\theta(\mathbf{Y})$ (which we would like to maximize), minus the KL divergence between the unknown distribution $q(\mathbf{Z})$ and the posterior $p_\theta(\mathbf{Z}|\mathbf{Y})$, which we could in theory evaluate if we knew $q(\mathbf{Z})$. On the right hand side we have the expected value of the model log-likelihood under the q distribution, and the KL divergence between q and the model prior $p_\theta(\mathbf{Z})$. Thus, the right hand side is a lower bound for the observed data log-likelihood, since the KL divergence is always non-negative.

This equation suggests a choice for the unknown distribution q: for a given estimate of the parameters θ, which we'll denote $\theta^{(k)}$, we can set $q(\mathbf{Z}) = p_{\theta^{(k)}}(\mathbf{Z}|\mathbf{Y})$. We'll denote the right hand side, for this choice of q, as

$$Q(\theta, \theta^{(k)}) = \mathbb{E}_{p_{\theta^{(k)}}(\mathbf{Z})}[\log p_\theta(\mathbf{Y}|\mathbf{Z})] - \mathrm{KL}[p_{\theta^{(k)}}(\mathbf{Z})||p_\theta(\mathbf{Z})] \tag{1.8}$$

where the first argument θ is a variable referencing the model parameters, and the second argument $\theta^{(k)}$ is a variable referencing the specific value of θ we use for the approximation q. Then, *at the point* $\theta = \theta^{(k)}$, the KL divergence between $q(\mathbf{Z})$ and $p_\theta(\mathbf{Z}|\mathbf{Y})$ goes to zero, and the right hand side is equal to the observed data log-likelihood! To further simplify notation we'll refer to the observed data log-likelihood as $\mathcal{L}(\theta) = \log p_\theta(\mathbf{Y})$, ignoring the dependence on $\mathbf{Y}$ to make clear that $\mathcal{L}$ is a function of the parmaters θ. Therefore, we have shown that in general

$$\mathcal{L}(\theta) >= Q(\theta, \theta^{(k)}) \tag{1.9}$$

since Q is a lower bound on the observed data log-likelihood. And specifically, we have shown that

$$\mathcal{L}(\theta^{(k)}) = Q(\theta^{(k)}, \theta^{(k)}) \tag{1.10}$$

So we know that at least one point the lower bound is tight. If we then maximize this lower bound $Q(\theta, \theta^{(k)})$ with respect to θ we get a new parameter estimate $\theta^{(k+1)}$, so that

$$Q(\theta^{(k+1)}, \theta^{(k)}) >= Q(\theta^{(k)}, \theta^{(k)}). \tag{1.11}$$

If we substitute $\theta = \theta^{(k+1)}$ into equation (1.9), we get

$$\mathcal{L}(\theta^{(k+1)}) >= Q(\theta^{(k+1)}, \theta^{(k)}) \tag{1.12}$$

In other words, with this new choice of θ the lower bound is no longer tight. However, by combining equations (1.9), (1.11), and (1.12), we find that

$$\mathcal{L}(\theta^{(k+1)}) >= Q(\theta^{(k+1)}, \theta^{(k)}) >= Q(\theta^{(k)}, \theta^{(k)}) = \mathcal{L}(\theta^{(k)}), \tag{1.13}$$

which demonstates that we have increased the observed data log-likelihood!

This is a remarkable property that has made the EM algorithm such a widely used method for ML estimation in LV models. To recap, the algorithm is as follows:

$$\begin{aligned} \text{E-step} &: \text{ set } q(\mathbf{Z}) = p_{\theta^{(k)}}(\mathbf{Z}|\mathbf{Y}) \\ \text{M-step} &: \theta^{(k+1)} = \underset{\theta}{\operatorname{argmax}}\, Q(\theta, \theta^{(k)}) \end{aligned}$$

These steps are then repeated until the increase in $\mathcal{L}(\theta)$ falls below a predefined threshold.

1.2 EM algorithm for Factor Analysis

To apply the EM algorithm to FA we first make the explicit assumption that the data $\{\mathbf{y}_n\}_{n=1}^N$ are i.i.d., which will allow us to perform the E-step in parallel for each data point. We now make use of the FA model definition in equation (1.1) to write out the following distributions for a single data point on iteration k:

$$\begin{aligned} p(\mathbf{y}_n|\mathbf{z}_n) &= \mathcal{N}(C^{(k)}\mathbf{z}_n, R^{(k)}) && (1.14)\\ p(\mathbf{z}_n) &= \mathcal{N}(0, I) && (1.15)\\ p(\mathbf{y}_n) &= \mathcal{N}(0, C^{(k)}(C^{(k)})^\intercal + R^{(k)}). && (1.16) \end{aligned}$$

With these definitions in hand we can now see the full EM algorithm for FA.

E-step. Given $R^{(k)}$ and $C^{(k)}$, the parameter estimates from iteration k, the LVs on iteration $k+1$ can be inferred by using Bayes Rule to calculate the posterior distribution:

$$\begin{aligned} p(\mathbf{z}_n|\mathbf{y}_n) &= \frac{p(\mathbf{y}_n|\mathbf{z}_n)p(\mathbf{z}_n)}{p(\mathbf{y}_n)} && (1.17)\\ &= \frac{\mathcal{N}(C^{(k)}\mathbf{z}_n, R^{(k)})\mathcal{N}(0, I)}{\mathcal{N}(0, C^{(k)}(C^{(k)})^\intercal + R^{(k)})} && (1.18)\\ &= \mathcal{N}(\beta^{(k)}\mathbf{y}_n, V^{(k)}) && (1.19) \end{aligned}$$

where $\beta^{(k)} = (C^{(k)})^\intercal[C^{(k)}(C^{(k)})^\intercal + R^{(k)}]^{-1}$, $V^{(k)} = I - \beta^{(k)}C^{(k)}$. The estimate of the LVs on iteration $k+1$ is then given by their expected value, $\hat{\mathbf{z}}_n^{(k+1)} = \beta^{(k)}\mathbf{y}_n$, which is computed for each data point independently thanks to the i.i.d. assumption.

M-step. Estimation of C can be performed using ML estimation given the full set of inferred latent variables $\hat{\mathbf{Z}}^{(k+1)} \in \mathbb{R}^{K\times N}$, their estimated covariance $V^{(k)}$, and the data $\mathbf{Y} \in \mathbb{R}^{D\times N}$:

$$C^{(k+1)} = \mathbf{Y}\left[\hat{\mathbf{Z}}^{(k+1)}\right]^\intercal \left(\hat{\mathbf{Z}}^{(k+1)}\left[\hat{\mathbf{Z}}^{(k+1)}\right]^\intercal + NV^{(k)}\right)^{-1} \quad (1.20)$$

The estimation of R starts with the unconstrained maximum likelihood estimate

$$\hat{R}^{(k+1)} = \hat{S} - \frac{C^{(k+1)}\hat{\mathbf{Z}}^{(k+1)}\mathbf{Y}^\intercal}{N} \quad (1.21)$$

where $\hat{S}$ is defined to be the sample covariance matrix of the data $\mathbf{Y}$. The constraint that R is diagonal can be easily implemented by first computing $\hat{R}$ and then simply setting the off-diagonal elements to zero [3].

1.3 Neuroscience applications of FA

The observation matrix C (known as the *factor loading matrix* in FA terminology) captures the correlational structure among the dimensions of $\mathbf{y}_n$, while the diagonal of R captures the variance that is unique to each dimension. For this reason FA has been popular in the neuroscience literature, because the variance of each neuron is in general different and depends on its firing rate.

The low-dimensional manifold extracted by FA can be used to discover structure in the high-dimensional data that is difficult to understand at the level of individual neurons. For example, in [4], FA was used to compare the differences in neural variability between spontaneous activity and stimulus-evoked activity. Single-neuron analyses demonstrated that variability decreased with stimulus onset. FA was able to further decompose this variability into a shared variability term (captured by the $CC^\intercal$ term in equation (1.16)) and a private variability term (captured by the diagonal R term in equation (1.16)), and the authors were able to conclude that stimulus onset caused a much larger decrease in the shared variability term.

FA has also been used to investigate constraints on learning through its use with brain-machine interfaces (BMI). In [5], monkeys learned to control a computer cursor that read neural activity patterns from their primary motor cortex. FA was used to find a low-dimensional manifold that contained most of the neural variability (the *Intrinsic Manifold*, or IM, defined by C in equation (1.1)). The mapping from neural activity to cursor movement was then altered by requiring new patterns of activity that fell within the IM or outside of it. Monkeys were able to learn new within-IM mappings quickly, but struggled to learn out-of-IM mappings. This study suggests that the IM is constrained by the structure of the network in which these neurons are embedded, and activity patterns that fall outside of the IM are much more difficult to learn.

One drawback to the use of FA in neuroscience is that it assumes a Gaussian distribution on the noise, whereas the Poisson distribution is often used to describe the discrete nature of spiking activity. A heuristic fix that is often used is to take the square root of the spike counts, which stabilizes the variance and provides a better match between data and model [6], though a Poisson FA has been developed as well [7].

1.4 PCA and probabilistic PCA

The model defined in equation (1.1) becomes probabilistic Principal Component Analysis (PPCA) [8] (and the independently defined Sensible Principal Component Analysis [9]) when R is constrained to be a scalar multiple of the identity matrix, $R = \sigma^2 I$. Similar to FA, this constraint can be easily implemented by calculating the full, unconstrained maximum likelihood estimate $\hat{R}$ (equation (1.21)) and then setting $\sigma^2 = \text{trace}(\hat{R})/N$ [9]. This choice of constraint does not allow each neuron to have an independent variance, and FA has been shown to significantly outperform PPCA when fitting neural data [6].

The ubiquitous PCA model, though not a proper probability model, can be recovered from the PPCA model by taking $R = \lim_{\sigma^2 \to 0} \sigma^2 I$. The posterior distribution of the latent states collapses to a single point, and the model fitting problem, though still solvable by the EM algorithm, is often performed by diagonalizing the sample covariance matrix $\hat{S} = VDV^\intercal$ and defining the columns of C to be the leading K eigenvectors:

$$\hat{C} = V_K \tag{1.22}$$
$$\hat{\mathbf{z}}_n = \hat{C}^\intercal \mathbf{y}_n \tag{1.23}$$

where V_K denotes the first K columns of the matrix V.

1.5 Neuroscience applications of PCA

PCA has enjoyed wide use in the neuroscience literature due to its computational simplicity and ease of interpretation. Perhaps its most common use (as in other fields) is as an exploratory data analysis tool, whereby neural activity can be projected into the first two or three PCA dimensions for visualization. For example, in [10], neural activity was recorded throughout larval zebrafish brains during a motor adaptation task. This activity was projected into the first three principal components, and four distinct phases of activity were discovered. These phases were then linked to distinct neural structures whose involvement in the task were previously unknown, and prompted additional experimental work to elucidate the role of these structures in motor learning.

PCA has also been used to study the effect of variability on behavior. In [11], PCA was used to define an "attention" axis in neural activity space by considering average responses across a range of attention conditions in a change-detection task. The location of single-trial activity along this dimension was predictive of the animal's performance in the task, establishing a link between neural variability and behavior. This study was an important first step in understanding the functional role of neural variability, and has spawned numerous studies that continue to utilize similar dimensionality reduction methods [12, 13].

PCA is typically applied to neural activity that has first been averaged over many identical trials and then subsequently smoothed, which destroys the discrete nature of the spike counts. Various extensions to PCA have been developed in the neuroscience literature to address this limitation and others, for instance to more accurately model discrete spike counts (Poisson PCA; [14]), to emphasize the structure associated with particular types of experimental trials (dPCA; [15]), and to emphasize dimensions relevant to dynamical structure (jPCA; [16]).

1.6 Resources

- Bishop 2006, Ch. 9: Mixture Models and EM [17]
- Roweis and Ghahramani (1999). A unifying review of Gaussian linear models. [3]
- Cunningham and Ghahramani (2015). Linear dimensionality reduction: survey, insights, and generalizations. [2]

Bibliography

[1] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[2] John P Cunningham and Zoubin Ghahramani. Linear dimensionality reduction: survey, insights, and generalizations. *Journal of Machine Learning Research*, 16(1):2859–2900, 2015.

[3] Sam Roweis and Zoubin Ghahramani. A unifying review of linear gaussian models. *Neural computation*, 11(2):305–345, 1999.

[4] Mark M Churchland, M Yu Byron, John P Cunningham, Leo P Sugrue, Marlene R Cohen, Greg S Corrado, William T Newsome, Andrew M Clark, Paymon Hosseini, Benjamin B Scott, et al. Stimulus onset quenches neural variability: a widespread cortical phenomenon. *Nature neuroscience*, 13(3):369, 2010.

[5] Patrick T Sadtler, Kristin M Quick, Matthew D Golub, Steven M Chase, Stephen I Ryu, Elizabeth C Tyler-Kabara, M Yu Byron, and Aaron P Batista. Neural constraints on learning. *Nature*, 512(7515):423, 2014.

[6] M Yu Byron, John P Cunningham, Gopal Santhanam, Stephen I Ryu, Krishna V Shenoy, and Maneesh Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. In *Advances in neural information processing systems*, pages 1881–1888, 2009.

[7] Gopal Santhanam, Stephen I Ryu, M Yu Byron, Afsheen Afshar, and Krishna V Shenoy. A high-performance brain–computer interface. *nature*, 442(7099):195, 2006.

[8] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[9] Sam T Roweis. Em algorithms for pca and spca. In *Advances in neural information processing systems*, pages 626–632, 1998.

[10] Misha B Ahrens, Jennifer M Li, Michael B Orger, Drew N Robson, Alexander F Schier, Florian Engert, and Ruben Portugues. Brain-wide neuronal dynamics during motor adaptation in zebrafish. *Nature*, 485(7399):471, 2012.

[11] Marlene R Cohen and John HR Maunsell. A neuronal population measure of attention predicts behavioral performance on individual trials. *Journal of Neuroscience*, 30(45):15241–15253, 2010.

[12] Neil C Rabinowitz, Robbe L Goris, Marlene Cohen, and Eero P Simoncelli. Attention stabilizes the shared gain of v4 populations. *Elife*, 4:e08998, 2015.

[13] AM Ni, DA Ruff, JJ Alberts, J Symmonds, and MR Cohen. Learning and attention reveal a general relationship between population activity and behavior. *Science*, 359(6374):463–465, 2018.

[14] David Pfau, Eftychios A Pnevmatikakis, and Liam Paninski. Robust learning of low-dimensional dynamics from large neural ensembles. In *Advances in neural information processing systems*, pages 2391–2399, 2013.

[15] Dmitry Kobak, Wieland Brendel, Christos Constantinidis, Claudia E Feierstein, Adam Kepecs, Zachary F Mainen, Xue-Lian Qi, Ranulfo Romo, Naoshige Uchida, and Christian K Machens. Demixed principal component analysis of neural population data. *Elife*, 5, 2016.

[16] Mark M Churchland, John P Cunningham, Matthew T Kaufman, Justin D Foster, Paul Nuyujukian, Stephen I Ryu, and Krishna V Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):51–56, 2012.

[17] CM Bishop. Pattern recognition and machine learning: springer new york. 2006.