# Unsupervised Learning

Carlos Valle

Departamento de Informática
Universidad Técnica Federico Santa María

*cvalle@inf.utfsm.cl*

December 8, 2015

# Overview

# Unsupervised learning

- In the previous chapters we have seen supervised learning methods which learn a map between $X$ and $Y$
- In this chapter we address unsupervised learning or "learning without a teacher."
- Now we have a set of $M$ observations $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M)$ of a random $I$-vector $X$ having joint density $P(X)$.
- The goal is to directly infer the properties of this probability density without the help of a supervisor or teacher providing correct answers or degree-of-error for each observation.

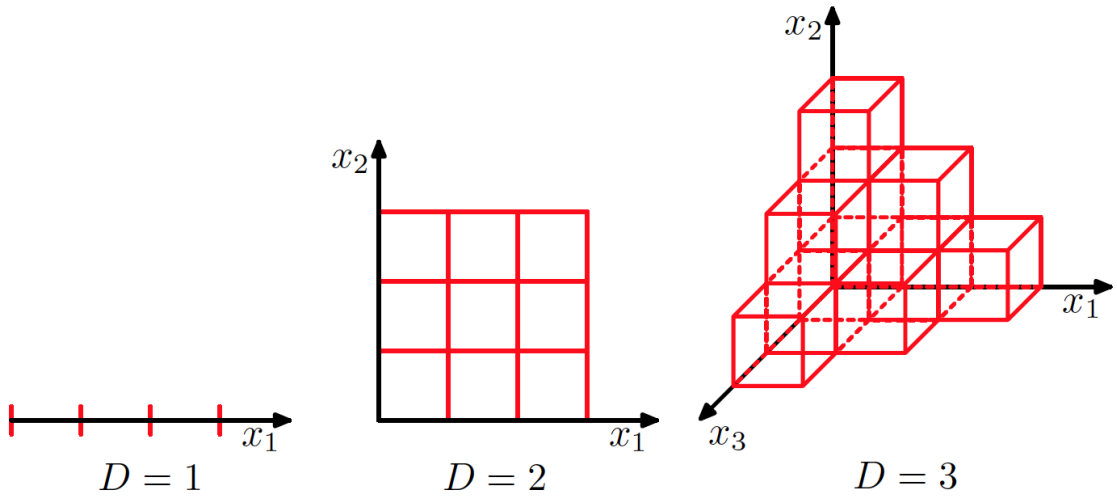
# Unsupervised learning (2)

- The dimension of $X$ is sometimes much higher than in supervised learning, and the properties of interest are often more complicated than simple location estimates.
- These factors are somewhat mitigated by the fact that $X$ represents all of the variables under consideration; one is not required to infer how the properties of $P(X)$ change, conditioned on the changing values of another set of variables.

# Curse of dimensionality

- In real applications of machine learning, we have to deal with input spaces of high dimensionality comprising many inputs variables.
- Consider the problem of labeling a point according to the labels of its neighbors.
- To tackle this problem, we will divide the input space in cells, and we will label the point as the class having the largest number of training points in the same cell.
- This approach has a serious problem when $x$ is a high-dimensional vector.
- If we divide a region of a space into regular cells, then the number of such cells grows exponentially with the dimensionality of the space.
- Therefore, with an exponentially large number of cells we would need an exponentially large amount of training examples in order to ensure that the cells are not empty.

# Curse of dimensionality (2)




- In regression problems we would like to model the problem by considering the dependencies among the input features:

$$f(\mathbf{x}, \beta) = \beta_0 + \sum_{i=1}^{I} \beta_i \mathbf{x}^{(i)} + \sum_{i=1}^{I} \sum_{j=1}^{I} \beta_{ij} \mathbf{x}^{(i)} \mathbf{x}^{(j)} + \sum_{i=1}^{I} \sum_{j=1}^{I} \sum_{k=1}^{I} \beta_{ijk} \mathbf{x}^{(i)} \mathbf{x}^{(j)} \mathbf{x}^{(k)}$$

# Curse of dimensionality (3)

- When $I$ increases, the number of coefficients of $\beta$ grows proportionally to $I^3$.
- Note that we only considered up to 3 attributes. Thus, this method is not scalable.
- Now, consider a sphere of radius $r = 1$ in a space of $I$ dimensions.
- The fraction of the volume of the sphere that lies between $r = 1 - \varepsilon$ and $r = 1$ is

$$\frac{V_I(1) - V_I(1-\varepsilon)}{V_I(1)} = 1 - (1-\varepsilon)^I,$$

- Here, $V_I(r) = K_I r^I$, where $K_I$ is a constant that only depends on $I$.
- Thus, for high-dimensional spaces, the volume of the sphere is concentrated near to the surface.

# Curse of dimensionality (4)

- Also, if we consider a probability distribution in a high-dimensional space (for example a multivariate gaussian), the probability mass of the Gaussian is concentrated just in a thin shell.
- In particular in clustering problems, the distance between two points converges as $I$ grows.

$$\lim_{I \to \infty} \frac{dist_{\max} - dist_{\min}}{dist_{\min}} = 0$$

- This problem is know as the curse of dimensionality.
- Fortunately we can develop effective algorithm for high-dimensional data.
- Real Data is usually confined in a subset of the whole input space. So, we have lower effective dimensionality.

# Curse of dimensionality (5)

- Real data exhibits local smoothness properties. Hence, small changes in the input variables produces small changes in the target variables.
- We can exploit local interpolation-like techniques to predict the target of the new instances.

# $K$-means

**Algorithm 1** $K$-means Algorithm

1: Initialize cluster centroids $\mu_1, \ldots, \mu_K \in \mathbb{R}^I$
2: **repeat**
3: **for** $m = 1, \ldots, M$ **do**
4: $c_m \leftarrow \text{argmin}_j ||\mathbf{x} - \mu_j||^2$
5: **end for**
6: **for** $j = 1, \ldots, K$ **do**
7: $\mu_j \leftarrow \frac{\sum_{m=1}^{M} 1\{c_m = j\}\mathbf{x}_m}{\sum_{m=1}^{M} 1\{c_m = j\}}$
8: **end for**
9: **until** convergence

# $K$-means (2)

- $K$ is the number of cluster we want to find.
- The cluster centroids $\mu_j$ represent the current centers of the clusters.
- To initialize the cluster centroids we can randomly select $K$ example from the training set.
- At each iteration each training example is assigned to the closest cluster by comparing it distance to each cluster centroid.
- And, the cluster centroids are updated as the mean of the points assigned to it.

# $K$-means convergence

- The distorsion function is defined

$$J(c, \mu) = \sum_{m=1}^{M} ||\mathbf{x}_{(m)} - \mu_{c_m}||^2$$

- It can be shown that $K$-means is exactly coordinate descent on $J$.
- The first loop of $K$ means repeatedly minimizes $J$ with respect to $c$ while holding $\mu$ fixed.
- Thus, $J$ must monotonically decrease, and the value of $J$ must converge.
- Usually this implies that $c$ and $\mu$ will converge too.
- However, $J$ is a non-convex function. Then, coordinate descent may fall into local optima.
- To reduce this effect, we can run $K$-means with different random initial values. Then, we use the model with the lowest $J$.

# Mixture of Gaussians

- We wish model the data by specifying a joint distribution $p(\mathbf{x}_m, \mathbf{z}_m) = p(\mathbf{x}_m|\mathbf{z}_m)p(\mathbf{z}_m)$,
- where $\mathbf{z}_m \sim$ Multinomial$(\phi)$ and $\phi_j = p(\mathbf{z}_m = j)$
- This implies $\phi_j \geq 0$, $\sum_{k=1}^{K} \phi_j = 1$.
- Now, we assume that $x_m|z_m = j \sim N(\mu_j, \Sigma_j)$.
- Here $K$ denote the number of values that the $\mathbf{z}_m$ can take on.
- The model suggest that each training example was drawn from one of the $K$ Gaussians depending on $\mathbf{z}_m$.
- This is called mixture of gaussians and $\mathbf{z}_m$ are the latent random variables.
- This means they are hidden or unobserved.

# Mixture of Gaussians (2)

- We can express the log-likelihood function, depending on $\phi, \mu, \Sigma$

$$\begin{aligned}\ell(\phi, \mu, \Sigma) &= \sum_{m=1}^{M} \log p(\mathbf{x}_m; \phi, \mu, \Sigma) \\ &= \sum_{m=1}^{M} \log \sum_{\mathbf{z}_m=1}^{K} p(\mathbf{x}_m|\mathbf{z}_m; \mu, \Sigma) p(\mathbf{z}_m; \phi)\end{aligned}$$

- However, we can't find a closed form for estimating the parameters using maximum likelihood.
- If we knew what the $\mathbf{z}_m$'s were, the maximum likelihood problem could have been arranged as

$$\ell(\phi, \mu, \Sigma) = \sum_{m=1}^{M} \log p(\mathbf{x}_m|\mathbf{z}_m; \mu, \Sigma) + \log p(\mathbf{z}_m; \phi)$$

# Mixture of Gaussians (3)

- Hence, if the $\mathbf{z}_m$'s were known we would compute $\phi, \mu$ and $\Sigma$

$$\begin{aligned}
\hat{\phi}_k &= \frac{1}{M}\sum_{m=1}^{M} I(\mathbf{z}_m = k),\ k = 1, \ldots, K \\
\hat{\mu}_k &= \frac{\sum_{m=1}^{M} I(\mathbf{z}_m = k)\mathbf{x}_m}{\sum_{m=1}^{M} I(\mathbf{z}_m = k)},\ k = 1, \ldots, K \\
\hat{\Sigma} &= \frac{\sum_{m=1}^{M} I(\mathbf{z}_m = k)(\mathbf{x}_m - \mu_k)(\mathbf{x}_m - \mu_k)^T}{\sum_{m=1}^{M} I(\mathbf{z}_m = k)},\ k = 1, \ldots, K \quad (1)
\end{aligned}$$

- Note that this solution is similar to what we obtained in LDA.

# EM algorithm for Mixture of Gaussians

- The EM algorithm is an iterative algorithm that has two main steps.
- For Mixture of Gaussians, the $E-$step tries to estimate the values of the $\mathbf{z}_m$'s as the posterior probability of our parameters given $\mathbf{x}_m$. I.e, using Bayes rule we obtain

$$p(\mathbf{z}_m = j|\mathbf{x}_m; \phi, \mu, \Sigma) = \frac{p(\mathbf{x}_m|\mathbf{z}_m; \mu, \Sigma)p(\mathbf{z}_m; \phi)}{\sum_{k=1}^{K} p(\mathbf{x}_m|\mathbf{z}_m; \mu, \Sigma)p(\mathbf{z}_m; \phi)}$$

- Here, $p(\mathbf{z}_m = j|\mathbf{x}_m; \phi, \mu, \Sigma)$ is given by evaluating the density of a Gaussian with mean $\mu_k$ and covariance $\Sigma_k$ at $\mathbf{x}_m$.
- $p(\mathbf{z}_m = k; \phi)$ is given by $\phi_k$
- The values $w_k^m$ calculated in the $E-$step represent our "soft" estimations for the values of $\mathbf{z}_m$.
- The $M-$step assumes that the estimations were correct, then we can compute $\phi, \mu$ and $\Sigma$ as in (1), except that we use $w_k^m$'s instead of $I(\mathbf{z} = k)$.

# EM algorithm for Mixture of Gaussians (2)

**Algorithm 2** EM for Gaussian Mixtures

1: **repeat**
2: **for** each $m, k$ **do** ▷ E-Step

$$w_k^m \leftarrow p(\mathbf{z}_m = k | \mathbf{x}_m; \phi, \mu, \Sigma)$$

3: **end for**
4: Update the parameters: ▷ M-Step

$$\begin{aligned}
\hat{\phi}_k &= \frac{1}{M} \sum_{m=1}^{M} w_k^{(m)}, \; k = 1, \ldots, K \\
\hat{\mu}_k &= \frac{\sum_{m=1}^{M} w_k^{(m)} \mathbf{x}_m}{\sum_{m=1}^{M} w_k^{(m)}}, \; k = 1, \ldots, K \\
\hat{\Sigma} &= \frac{\sum_{m=1}^{M} w_k^{(m)} (\mathbf{x}_m - \mu_k)(\mathbf{x}_m - \mu_k)^T}{\sum_{m=1}^{M} w_k^{(m)}}, \; k = 1, \ldots, K
\end{aligned}$$

5: **until** convergence

# EM algorithm for Mixture of Gaussians (3)

- Note that this algorithm is similar to $K$-means, except that we have "soft" cluster assignments instead of the 'hard" assignments of $K$-means.
- This algorithm is also susceptible to local optima.

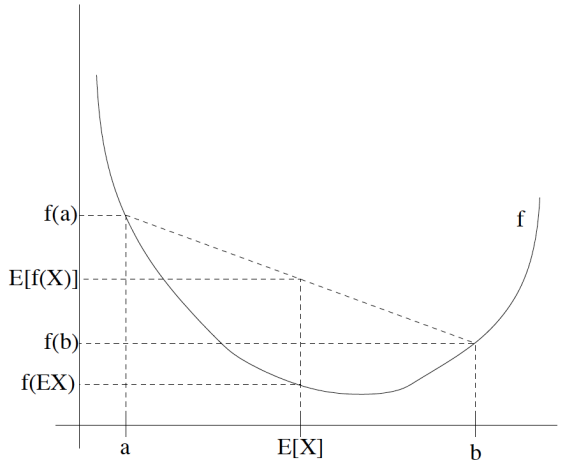
# Jensen's inequality

- Let $f$ be a function whose domain is the set of real numbers.
- Recall $f$ is a convex function if $f''(\mathbf{x}) \geq 0$, $\forall \mathbf{x} \in \mathbb{R}$.
- If $\mathbf{x} \in \mathbb{R}^I$, its hessian $H$ is positive semi-definite $(H \geq 0)$.
- If $f''(\mathbf{x}) > 0$, $\forall \mathbf{x}$ is a strictly convex function (in the vector-valued case, $H$ must be positive definite $H > 0$.

**Theorem (Jensen's inequality)**

*Let $f$ be a convex function, and let $X$ be a random variable. Then $E[f(X)] \geq f(E[X])$. Moreover, if $f$ is strictly convex, then $E[f(X)] = f(E[X])$ holds true if and only if $X = E[X]$ with probability 1 (i.e. if $X$ is a constant).*

# Jensen's inequality (2)




- In the figure above, $f$ is drawn by the solid line.
- $X$ is a random variable with two equally probable values $a$ and $b$.
- Thus, $E[X]$ is the midpoint between $a$ and $b$.
- Moreover, $E[f(X)]$ is the midpoint on the $y-$axis between $f(a)$ and $f(b)$.
- We can see that because $f$ is convex, it must be the case that $E[f(X)] \geq f(E[X])$.
- Jensen's inequality for concave functions is $E[f(X)] \leq f(E[X])$

# The EM algorithm

- We wish to fit the parameters of a model $p(\mathbf{x}, \mathbf{z})$ to the data, where the likelihood is given by

$$\begin{aligned} \ell(\theta) &= \sum_{m=1}^{M} \log p(\mathbf{x}_m; \theta) \\ &= \sum_{m=1}^{M} \log \sum_{\mathbf{z}_m} p(\mathbf{x}_m, \mathbf{z}_m; \theta) \end{aligned}$$

- Since the $\mathbf{z}_m$'s are the latent random variables, explicitly finding the maximum likelihood estimates of the parameters $\theta$ may be hard.
- It would be easy if we knew the $\mathbf{z}_m$'s.
- The EM algorithm construct a lower bound on $\ell$ ($E-$step), and then optimize that lower-bound ($M-$step).

# The EM algorithm (2)

- For each $m$ let $Q_m$ be some distribution over the $\mathbf{z}$'s ($\sum_{\mathbf{z}} Q_m(\mathbf{z}) = 1$, $Q_m(\mathbf{z}) \geq 0$)
- Consider the following

$$\sum_m \log p(\mathbf{x}_m; \theta) = \sum_m \log \sum_{\mathbf{z}_m} p(\mathbf{x}_m, \mathbf{z}_m; \theta) \quad (2)$$

$$= \sum_m \log \sum_{\mathbf{z}_m} Q_m(\mathbf{z}_m) \frac{p(\mathbf{x}_m, \mathbf{z}_m; \theta)}{Q_m(\mathbf{z}_m)} \quad (3)$$

$$\geq \sum_m \sum_{\mathbf{z}_m} Q_m(\mathbf{z}_m) \log \frac{p(\mathbf{x}_m, \mathbf{z}_m; \theta)}{Q_m(\mathbf{z}_m)} \quad (4)$$

- Since $f(\mathbf{x}) = \log \mathbf{x}$ is a concave function ($f''(\mathbf{x}) = -1/\mathbf{x}^2$, $\forall \mathbf{x} \in \mathbb{R}^+$).

# The EM algorithm (3)

- Note that the term

$$\sum_{\mathbf{z}_m} Q_m(\mathbf{z}_m) \left[ \frac{p(\mathbf{x}_m, \mathbf{z}_m; \theta)}{Q_m(\mathbf{z}_m)} \right],$$

- is an expectation of $\frac{p(\mathbf{x}_m, \mathbf{z}_m; \theta)}{Q_m(\mathbf{z}_m)}$ with respect to $\mathbf{z}_m$ drawn from $Q_m$.
- Thus, we can go from (3) to (4) by using Jensen's inequality

$$f\left( E_{\mathbf{z}_m \sim Q_m} \left[ \frac{p(\mathbf{x}_m, \mathbf{z}_m; \theta)}{Q_m(\mathbf{z}_m)} \right] \right) \geq E_{\mathbf{z}_m \sim Q_m} \left[ f\left( \frac{p(\mathbf{x}_m, \mathbf{z}_m; \theta)}{Q_m(\mathbf{z}_m)} \right) \right]$$

- We can see that $E_{\mathbf{z}_m \sim Q_m} \left[ f\left( \frac{p(\mathbf{x}_m, \mathbf{z}_m; \theta)}{Q_m(\mathbf{z}_m)} \right) \right]$ is a lower bound on $\ell(\theta) = \sum_m \log p(\mathbf{x}_m; \theta)$

# How to select $Q_m$?

- We could pick $Q_i$ to try to make the lower-bound tight at that value of $\theta$
- Hence, we need that equation (4) holds with equality at our value of $\theta$.
- This is only true when

$$\frac{p(\mathbf{x}_m, \mathbf{z}_m; \theta)}{Q_m(\mathbf{z}_m)} = c$$

  for some $c > 0$ does not depend on $\mathbf{z}_m$.
- So, we can choose

$$Q_m(\mathbf{z}_m) \propto p(\mathbf{x}_m, \mathbf{z}_m; \theta)$$

# How to select $Q_m$? (2)

- Also, we should note that $z_m$ is a distribution, hence, $\sum_{\mathbf{z}} Q_m(\mathbf{z}) = 1$. Then,

$$\begin{aligned} Q_m(\mathbf{z}_m) &= \frac{p(\mathbf{x}_m, \mathbf{z}_m; \theta)}{\sum_{\mathbf{z}} p(\mathbf{x}_m, \mathbf{z}_m; \theta)} \\ &= \frac{p(\mathbf{x}_m, \mathbf{z}_m; \theta)}{p(\mathbf{x}_m; \theta)} \\ &= p(\mathbf{z}_m | \mathbf{x}_m; \theta) \end{aligned}$$

- Now in the $E-$step we choose the $Q_m$'s according to the posterior distribution $\mathbf{z}_m|\mathbf{x}_m$.
- Then in the $M-$step we maximize (4) with respect to the set of parameters $\theta$ to obtain the new $\theta$'s.

# General EM algorithm

**Algorithm 3** General EM Algorithm

1: **repeat**
2: **for** $m = 1, \ldots, M$ **do** ▷ E-Step
3: $Q_m(\mathbf{z}_m) \leftarrow p(\mathbf{z}_m|\mathbf{x}_m; \theta)$
4: **end for**
5: Set ▷ M-step

$$\theta \leftarrow \text{argmax}_\theta \sum_m \sum_{\mathbf{z}_m} Q_m(\mathbf{z}_m) \log \frac{p(\mathbf{x}_m, \mathbf{z}_m; \theta)}{Q_m(\mathbf{z}_m)}$$

6: **until** convergence

# EM convergence

- We need $\ell(\theta^{(t)}) \leq \ell(\theta^{(t+1)})$ to prove convergence.
- We now that

$$\ell^{(t)} = \sum_m \sum_{\mathbf{z}_m} Q_m^{(t)}(\mathbf{z}_m) \log \frac{p(\mathbf{x}_m, \mathbf{z}_m; \theta^{(t)})}{Q_m^{(t)}(\mathbf{z}_m)}$$

- where $Q_m^{(t)}(\mathbf{z}_m) \leftarrow p(\mathbf{z}_m | \mathbf{x}_m; \theta^{(t)})$.
- According to equation (4) we have

$$\ell^{(t+1)} \geq \sum_m \sum_{\mathbf{z}_m} Q_m^{(t)}(\mathbf{z}_m) \log \frac{p(\mathbf{x}_m, \mathbf{z}_m; \theta^{(t+1)})}{Q_m^{(t)}(\mathbf{z}_m)}.$$

# EM convergence (2)

- Moreover, $\ell^{(t+1)}$ is chosen as in step 5 on The General EM algorithm.
- Thus,

$$\ell^{(t+1)} \geq \sum_m \sum_{\mathbf{z}_m} Q_m^{(t)}(\mathbf{z}_m) \log \frac{p(\mathbf{x}_m, \mathbf{z}_m; \theta^{(t)})}{Q_m^{(t)}(\mathbf{z}_m)} = \ell^{(t)}.$$

- If we define

$$J(Q, \theta) = \sum_m \sum_{\mathbf{z}^{(m)}} Q_m(\mathbf{z}_m) \log \frac{p(\mathbf{x}_m, \mathbf{z}_m; \theta)}{Q_m(\mathbf{z}_m)}$$

- It can be proved that the EM is a condinate descent algorithm on $J$.
- In which the $E-$step maximizes it with respect to $Q$
- And the $E-$step maximizes it with respect to $\theta$.

# Revisiting Mixture of gaussians

- Appliying the general EM to our mixture of gaussians model:
- The $E-$step is

$$Q_m(\mathbf{z}_m) = p(\mathbf{z}_m = k|\mathbf{x}_m; \phi, \mu, \Sigma) = w_k^m$$

- For the $M-$step we need to maximize the quantity

$$\sum_{m=1}^{M} \sum_{\mathbf{z}_m} Q_m(\mathbf{z}_m) \log \frac{p(\mathbf{x}_m, \mathbf{z}_m; \phi, \mu, \Sigma)}{Q_m(\mathbf{z}_m)}$$

$$= \sum_{m=1}^{M} \sum_{k=1}^{K} Q_m(\mathbf{z}_m = k) \log \frac{p(\mathbf{x}_m|\mathbf{z}_m = k; \mu, \Sigma)p(\mathbf{z}_m = k; \phi)}{Q_m(\mathbf{z}_m = k)}$$

$$= \sum_{m=1}^{M} \sum_{k=1}^{K} w_j^m \log \frac{\frac{1}{(2\pi)^{n/2}|\Sigma_k|^{1/2}} \exp(-\frac{1}{2}(\mathbf{x}_m - \mu_k)^T \Sigma_k^{-1}(\mathbf{x}_m - \mu_k))\phi_k}{w_j^m}$$

# Revisiting Mixture of gaussians (2)

- Maximizing it with respect to $\mu_\ell$

$$\nabla\mu_\ell \sum_{m=1}^{M}\sum_{k=1}^{K} w_j^m \log \frac{\frac{1}{(2\pi)^{n/2}|\Sigma_k|^{1/2}}\exp(-\frac{1}{2}(\mathbf{x}_m-\mu_k)^T\Sigma_k^{-1}(\mathbf{x}_m-\mu_k))\phi_k}{w_j^m}$$

$$= \nabla\mu_\ell \sum_{m=1}^{M}\sum_{k=1}^{K} w_j^m \frac{1}{2}(\mathbf{x}_m-\mu_k)^T\Sigma_k^{-1}(\mathbf{x}_m-\mu_k)$$

$$= \frac{1}{2}\sum_{m=1}^{M} w_j^m \nabla_{\mu_\ell} 2\mu_\ell^T\Sigma_\ell^{-1}\mathbf{x}_m - \mu_\ell^T\Sigma_\ell^{-1}\mu_\ell$$

$$= \sum_{m=1}^{M} w_j^m(\Sigma_\ell^{-1}\mathbf{x}_m - \Sigma_\ell^{-1}\mu_\ell)$$

# Revisiting Mixture of gaussians (3)

- Setting this to zero and solving for $\mu_\ell$ we have

$$\mu_\ell = \frac{\sum_{m=1}^{M} w_j^m \mathbf{x}_m}{\sum_{m=1}^{M} w_j^m}$$

- To update $\phi_k$ we need to minimize

$$\sum_{m=1}^{M} \sum_{k=1}^{K} w_j^m \log \phi_k.$$

- However, we have an additional constraint that $\sum_k \phi_k = 1$.

# Revisiting Mixture of gaussians (4)

- Hence, we must construct the Lagrangian

$$\mathcal{L}(\phi) = \sum_{m=1}^{M} \sum_{k=1}^{K} w_j^m \log \phi_k + \beta \left( \sum_k \phi_k - 1 \right)$$

- Taking derivative we obtain

$$\frac{\partial}{\partial \phi_k} \mathcal{L}(\phi) = \sum_{m=1}^{M} \frac{w_j^m}{\phi_k} + 1$$

- Setting this to zero and solving for $\phi_k$

$$\phi_k = \frac{\sum_{m=1}^{M} w_j^m}{-\beta}$$

# Revisiting Mixture of gaussians (5)

- Therefore, $-\beta = \sum_{m=1}^{M} \sum_{k=1}^{K} w_j^m = \sum_{m=1}^{M} 1 = M$.
- Then,

$$\phi_k = \frac{1}{M} \sum_{k=1}^{K} w_j^m.$$

# Principal component analysis (PCA)

- PCA tries to identify the subspace in which the data approximately lies.
- Typically PCA uses standardized data.
- We would like to choose a direction $v$ that retains the biggest amount of variance of the data.
- To maximize the variance of the projections, we would like to choose a unit-length $v$ so as to maximize

$$\begin{aligned}\frac{1}{M}\sum_{m=1}^{M}(\mathbf{x}_m^T\mathbf{u})^2 &= \frac{1}{M}\sum_{m=1}^{M}\mathbf{u}^T\mathbf{x}_m\mathbf{x}_m^T\mathbf{u} \\ &= \mathbf{u}^T\left(\sum_{m=1}^{M}\mathbf{x}_m\mathbf{x}_m^T\right)\mathbf{u}\end{aligned}$$

# Principal component analysis (PCA) (2)

- Note that maximizing this subject to $||u|| = 1$ give us the principal eigenvector of $\frac{1}{M}X^TX$ (which is the covariance matrix because the data was centered)
- Recall if we write $X = UDV^T$, $X^TX = VD^2V^T$ where $v_j$ (columns of V) are the eigenvectors, also called the principal components
- $d_j$ are the eigenvalues, where $d_1 \geq d_2 \geq \ldots$.
- The first principal component direction $v_1$ has the property that $z_1 = Xv_1$ has the largest sample variance among all normalized linear combinations of the columns of X.
- Note that the new basis $\mathbf{z}_j = X\mathbf{v}_j = \mathbf{u}_j d_j$. Thus, $\text{cov}(\mathbf{u}_j d_j) = \frac{d_j^2}{M}$
- PCA chooses the $k$ eigenvectors with larger variance.
- PCA is a dimensionality reduction technique.

# Independent Components Analysis

- This approach is motivated by the cocktail party problem.
- Here $I$ speakers are speaking simultaneously at a party and $I$ microphones placed in the room records only an overlapping combination of the $I$ speaker's voices.
- We would like to separate out the $I$ speakers' speech signals using $I$ microphones
- We could model this problem assuming that we observe the $I$ linear mixtures $x^{(1)}, x^{(2)}, \ldots, x^{(I)}$ as a combination of the $I$ independent latent variables (independent components)

$$x^{(i)} = a_{i1}s^{(1)} + a_{i2}s^{(2)} + \cdots + a_{iI}s^{(I)},\ i = 1, \ldots, I \tag{5}$$

- This can be written in a matrix form as

$$x = As,$$

- where $s = (s^{(1)}, \ldots, s^{(I)})^T$.

# Independent Components Analysis (2)

- Considering the whole training set we have

$$x_m = As_m$$

- Let $W = A^{-1}$ be the unmixing matrix. So, our goal is to find

$$s_m = Wx_m$$

- Each row of $W$ is denoted by $w^{(i)^T}$, which it turns that

$$s_m^{(i)} = w^{(i)^T} x_m$$

# Ambiguities of ICA

- We cannot determine the order of the latent variables
- Since $A$ and $s$ are unknown, we can freely change the order of the terms of equation (5).
- Formally, a permutation matrix $\mathbf{P}$ and its inverse can be substituted in the model to give $\mathbf{x} = \mathbf{A}\mathbf{P}^{-1}\mathbf{P}\mathbf{s}$.
- The elements of $\mathbf{P}\mathbf{s}$ are the original independent variables $s^{(i)}$, but in another order. The matrix $\mathbf{A}\mathbf{P}^{-1}$ is just a new unknown mixing matrix, to be solved by the ICA algorithms.
- Further, we can't recover the correct scaling of the $w^{(i)}$'s.
- Because we can multiply $A$ by a constant $c$, i.e, we have $cA$ instead of $A$.
- At the same time we multiply $s$ by $1/c$.
- This As remains unalterable. Moreover, $s_m^{(i)}$ and $-s_m^{(i)}$ are not distinguishable.

# Densities and linear transformations

- Suppose we have a random variable $s$ drawn according to some density $p_s(s)$.
- Let $x$ be a random variable defined according to $x = As$.
- What is $p_x$, the density of $x$?
- Let $W = A^{-1}$.
- Recall that $f_Y(y) = f_X(g^{-1}(y))|\frac{\partial}{\partial y}g^{-1}(y)|$.
- Thus,

$$p_x(x) = p_s(Wx)|W|.$$

# Densities and linear transformations (2)

- Let's suppose that each independent source $s^{(i)}$ has a density distribution $p_s$. Hence, the joint distribution of the sources $s$ is given by

$$p(s) = \prod_{i=1}^{I} p_s(s^{(i)}).$$

- This implies that

$$p_x(x) = \prod_{i=1}^{I} p_s(w^{(i)^T} x)|W|.$$

- Now we need to choose $p_s$.

# Why ICA doesn't work on Gaussian data

- To see why $s_i$ are nongaussians.
- Let's assume that the mixing matrix is orthogonal and $s^{(i)}$ are gaussians.
- Consider $I = 2$ and $s \sim N(o, \mathbf{I})$.
- where $\mathbf{I}$ is the identity matrix.
- Their joint density is given by

$$p(x_1, x_2) = \frac{1}{2\pi} \exp\left(-\frac{x_1^2 + x_2^2}{2}\right)$$

- It is rotationally symmetric about the origin.

# Why ICA doesn't work on Gaussian data (2)

- One can prove that the distribution of any rotation/reflection of the gaussian $(x_1, x_2)$ has exactly the same distribution as $(x_1, x_2)$.
- In other words, the matrix $\mathbf{A}$ is not identifiable for gaussian independent components. (Actually, if just one of the independent components is gaussian, the ICA model can still be estimated.)
- Instead, we will choose the sigmoid function as the cumulative distribution $g(s) = 1/(1 + e^{-s})$ then $p_s(s) = g'(s)$
- The underlying assumption here is that the data has been preprocessed to have zero mean.
- Note that $E[s] = 0$ if $p_s(s) = g'(s)$, which implies $E[x] = E[As] = 0$

# ICA algorithm

- Given a training set $\mathbf{x}_1, \ldots, \mathbf{x}_M$, we want to maximize the log likelihood function

$$\ell(W) = \sum_{m=1}^{M} \left( \sum_{i=1}^{I} \log g'(w^{(i)^T} x_m) + \log |W| \right)$$

- By taking derivatives and using the fact that $\nabla_W |W| = |W|(W^{-1})^T$ we obtain the update rule

$$W \leftarrow \alpha \left( \begin{bmatrix} 1 - 2g(w^{(1)^T} x_m) \\ 1 - 2g(w^{(2)^T} x_m) \\ \vdots \\ 1 - 2g(w^{(I)^T} x_m) \end{bmatrix} x^{(i)^T} + (W^T)^{-1} \right)$$

- Visiting examples in randomly permuted order accelerate convergence.

# Any questions?