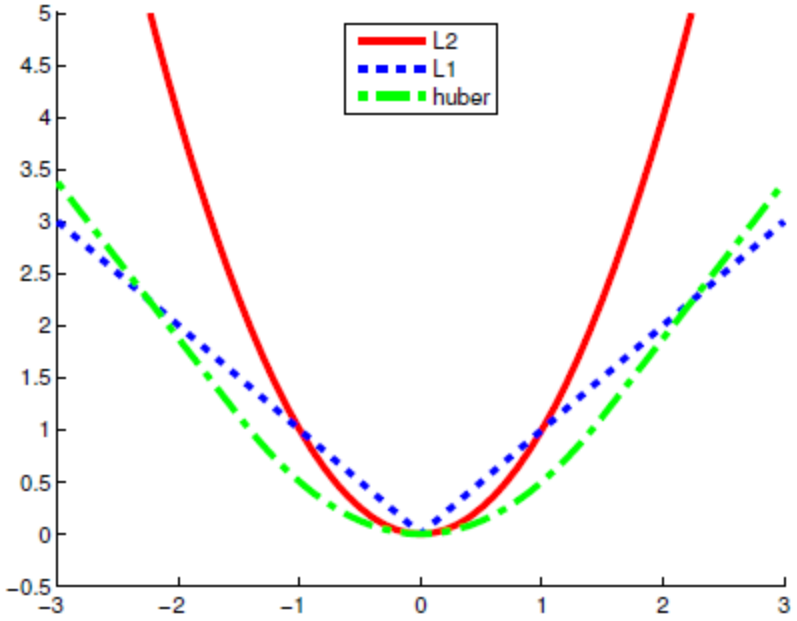
Sparsity

- L2 regularization
  - Sparsity - ?
  - Optimization – Easy
- L1 regularization
  - Sparsity - ?
  - Optimization – more difficult, not differentiable.
- Huber regularization
  - Robust to outliers
  - Optimization – Easy, differentiable everywhere



Kernel Methods

- Kernel function: $k(x_i, x_j)$
  - $k : \chi * \chi \rightarrow R; \quad x_i \in \chi$
  - Usually Symmetric, Non-negative
  - Measure of similarity between x and x'
- A kernel is positive semi definite

Gram Matrix

$$\mathbf{K} = \begin{pmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \cdots & \kappa(\mathbf{x}_1, \mathbf{x}_N) \\ & \vdots & \\ \kappa(\mathbf{x}_N, \mathbf{x}_1) & \cdots & \kappa(\mathbf{x}_N, \mathbf{x}_N) \end{pmatrix}$$

if it gives rise to PSD gram matrix for any N

Kernel Methods

- Mercer's theorem
  - Loosely speaking it states that every PD kernel can be expressed as

$$\kappa(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$$

$\phi(x)$ could be infinite dimensional

- For certain kernels (e.g., polynomial) $\phi(x)$ is finite.

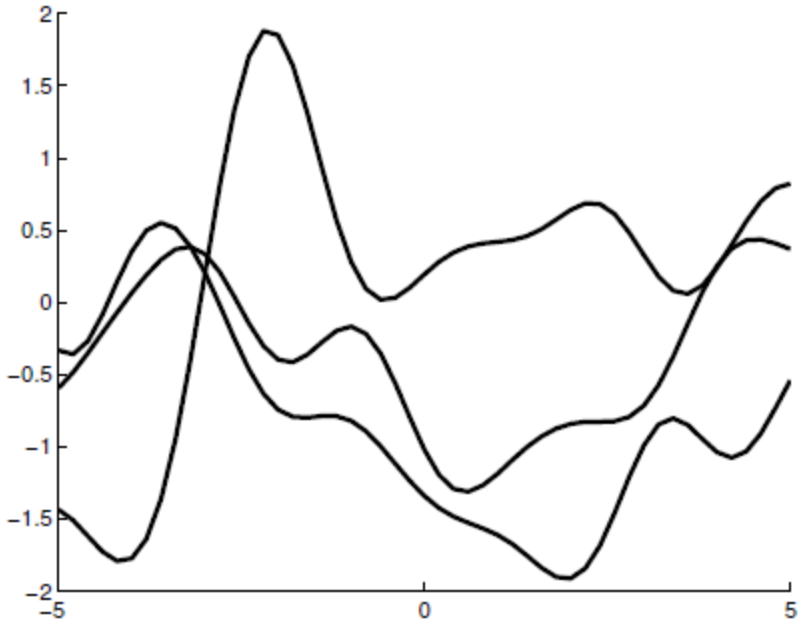
$$\begin{aligned}(1 + \mathbf{x}^T\mathbf{x}')^2 &= (1 + x_1x_1' + x_2x_2')^2 \\ &= 1 + 2x_1x_1' + 2x_2x_2' + (x_1x_1)^2 + (x_2x_2')^2 + 2x_1x_1'x_2x_2'\end{aligned}$$

$$\phi(\mathbf{x}) = [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2]^T$$

Gaussian Process

GP is a collection of random variables, any finite number of which are jointly Gaussian.

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}'))$$



Gaussian Process Models

- $p(\boldsymbol{y}, \boldsymbol{f}|\boldsymbol{x}) = p(\boldsymbol{f}|\boldsymbol{x})\prod p(y_i|f_i)$

- Regression : $y_i = f_i + \epsilon; \epsilon \sim N(0, \sigma^2)$

  $: p(y_i|f_i) = N(0, \sigma^2 I)$

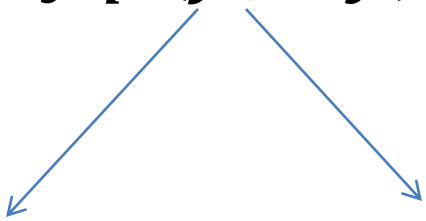
- Classification : $y_i = \pm 1$

  $: p(y_i|f_i) = \phi(y_i f_i)$

Prediction

- $p(y_*|\boldsymbol{x}, \boldsymbol{y}, x_*) = \int p(y_*|f_*)\boldsymbol{p}(f_*|\boldsymbol{x}, \boldsymbol{y}, x_*)df_*$

- $p(f_*|\boldsymbol{x}, \boldsymbol{y}, x_*) = \int p(\boldsymbol{f}|\boldsymbol{x}, \boldsymbol{y})\boldsymbol{p}(f_*|\boldsymbol{x}, \boldsymbol{f}, x_*)\boldsymbol{df}$



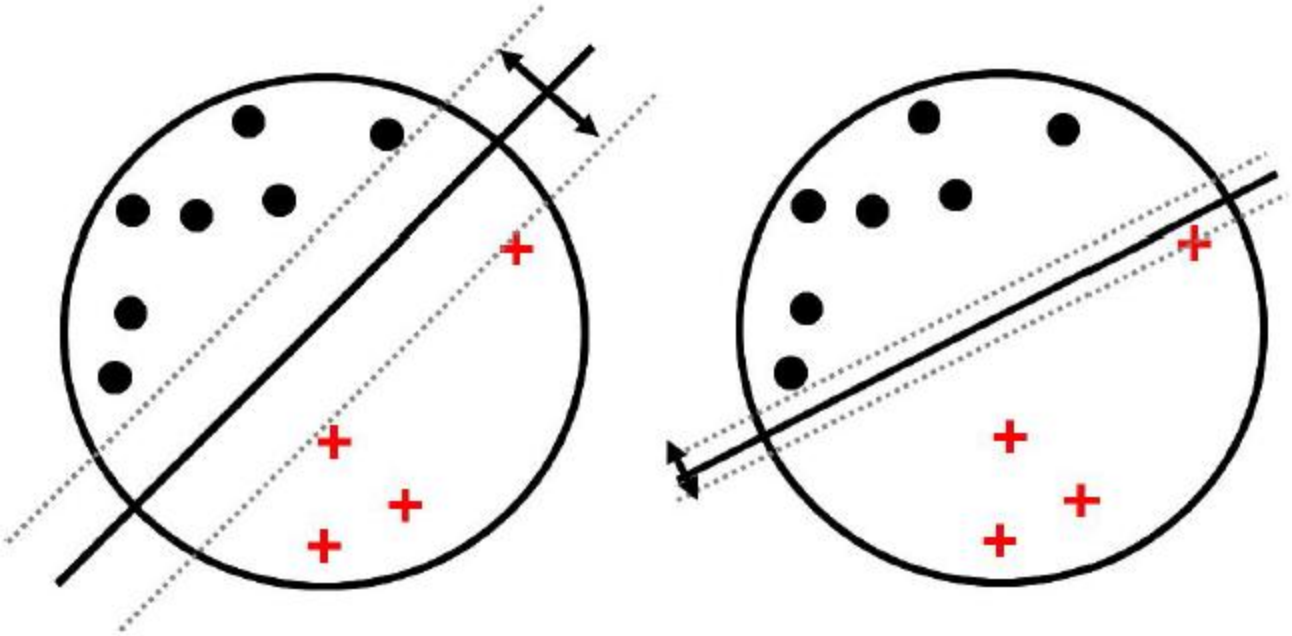
Regression
= Gaussian

Classification = Non – Gaussian, needs approximations

Laplace Approximation

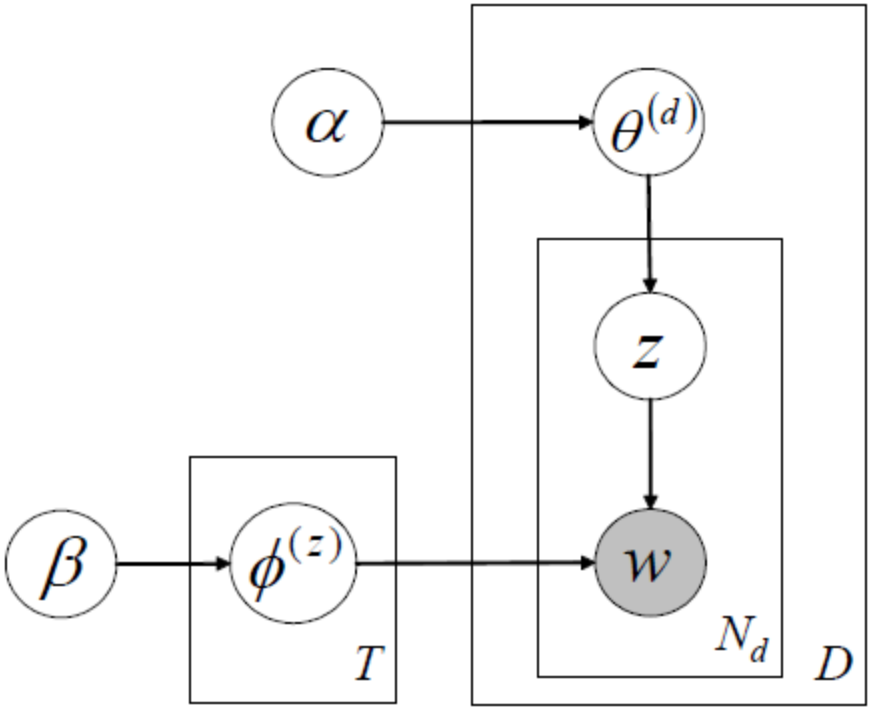
- $p(\boldsymbol{f}|\boldsymbol{x},\boldsymbol{y}) \approx N(\boldsymbol{f}|\boldsymbol{m},\boldsymbol{\Sigma})$
  - Laplace approximation – Taylor series approximation of the log posterior around the mode.
  - $\boldsymbol{m} = \boldsymbol{f}_{\mathrm{mode}}$
  - $\boldsymbol{\Sigma} = \mathrm{H}^{-1}$ at the mode.

Support Vector Machines



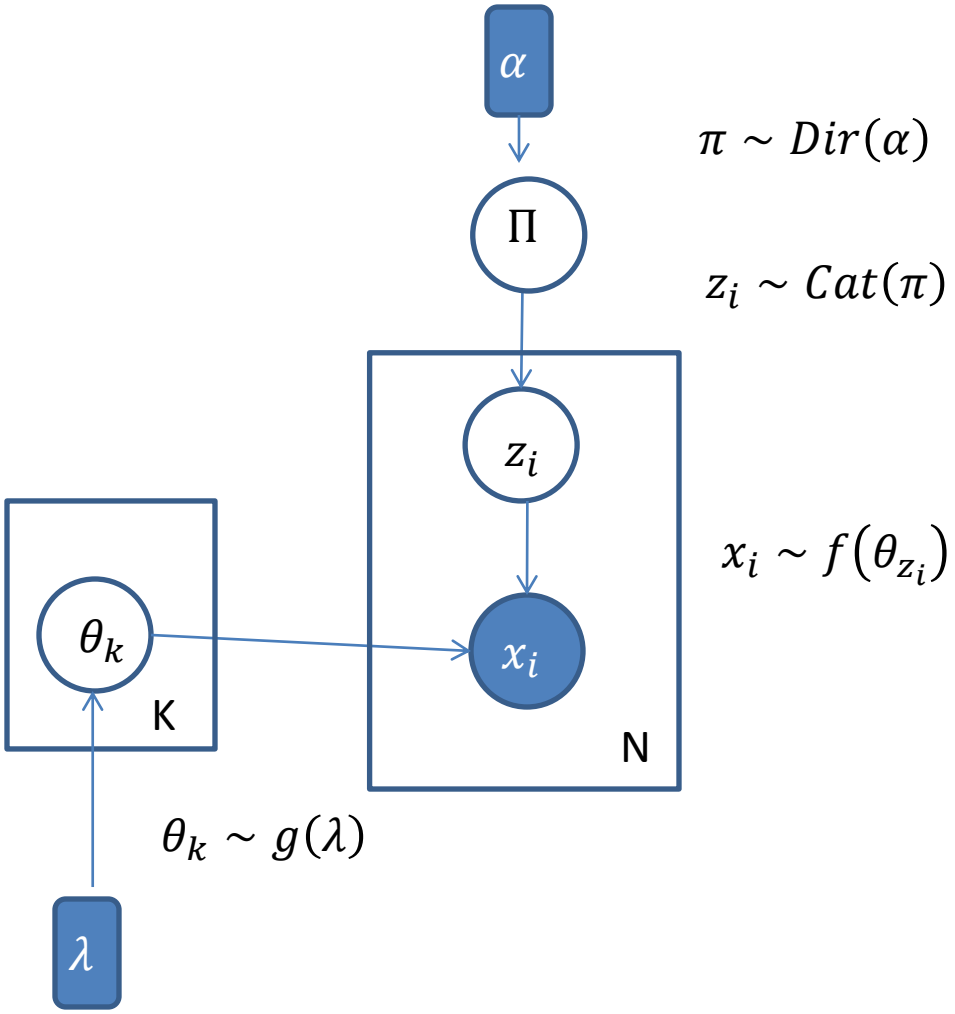
- Maximize Margin between classes
- Data instances closest to the decision boundary are the support vectors.
- Dual weights of all but SVs = 0

Topic Models



- Each document is a mixture of topics.
- Each topic is a cluster, with a distribution over words.
- All documents share the same topics, but the mixing proportions are different.

Gibbs Sampling



- For Gibbs Sampling we need

$$z_i \sim p(z_i \,|z_{-i}, x, \pi, \theta, \alpha, \lambda)$$
$$= p(z_i|x_i, \pi, \theta)$$
$$\propto p(z_i|\pi) f(x_i|\theta_{zi})$$

$$\Pi \sim p(\Pi \,|z, x, \theta, \alpha, \lambda)$$
$$\propto p(\Pi, z|\alpha)$$

$$\theta_k \sim p(\theta_k \,|\theta_{-k}, \Pi, x, \alpha, \lambda)$$
$$= p(\theta_k|x_m : \{z_m = k\}, \lambda)$$
$$\propto p(x_m, \theta_k|\lambda)$$

Mixture Models

- Defined as a convex combination of probability distributions,

$$p(x_i|\theta) = \sum_{k=1}^{K} \pi_k p_k(x_i|\theta)$$

- For $x_i \in \mathbb{R}^D \quad \forall i = 1, \ldots, N$
- Where the mixture weights $\sum_k \pi_k = 1$ and $\pi_k \geq 0$

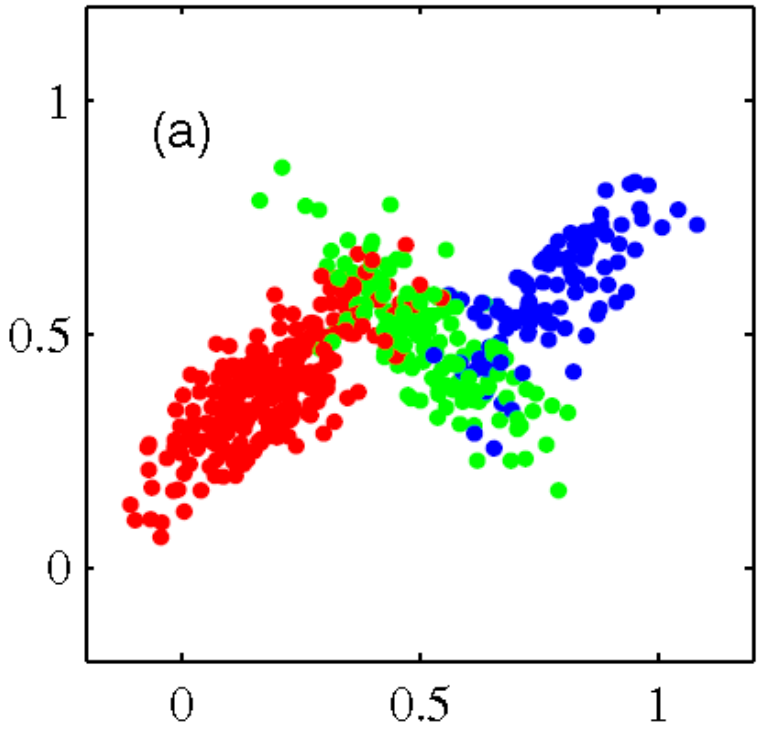


Figure 1: Example 500 points drawn from a three component Gaussian mixture. (Bishop)

Mixture Models

e.g. (Mixture of Bernoulli)

D-dimensional binary vectors which represent "coinflips"

$$x_i \quad : \quad \text{Binary variables } i = 1, \ldots, D$$

Conditional distribution:

$$x_i \mid \mu_i \sim Ber(\mu_i)$$

Mixture distribution for k=1,...,K clusters:

$$p(x|\mu, \pi) = \sum_{k=1}^{K} \pi_k \prod_{i=1}^{D} \mu_{ki}^{x_i} (1 - \mu_{ki})^{1-x_i}$$

Expectation Maximization

- Exploits interpretation that ML/MAP would be "easy" if data were fully observed
- Constructs a lower bound on log-likelihood

$$\log p(x \mid \theta) \geq \mathbb{E}_q \left[\log p(x, z \mid \theta)\right] + H(q)$$

where z is *missing data*, expectation is over $q(z)$ and $H(q)$ is *entropy*

- Terminology:

$$\begin{aligned} \log p(x \mid \theta) &: \text{Marginal Likelihood} \\ \log p(x, z \mid \theta) &: \text{Complete data log-likelihood} \\ \mathbb{E}_q \left[\log p(x, z \mid \theta)\right] &: \text{Expected complete data log-likelihood} \end{aligned}$$

Expectation Maximization

E-Step:

For iteration *(i)*:

Compute: $q^{(i)}(z) = p(z \mid x, \theta^{(i-1)})$

Evaluate *sufficient statistics* of: $\mathbb{E}_{q^{(i)}}\left[\log p(x, z \mid \theta)\right]$

M-Step:

Update parameter estimate:

$$\theta^{(i)} = \arg\max_{\theta} \mathbb{E}_{q^{(i)}}\left[\log p(x, z \mid \theta)\right]$$

Dimensionality Reduction

Principle Component Analysis:

- Orthogonal projection onto lower dimensional linear subspace, known as *principle subspace*
- Projection *maximizes variance*
- Goal: Represent high dimensional data $x \in \mathbb{R}^D$ by $z \in \mathbb{R}^M \quad \text{where } M << D$

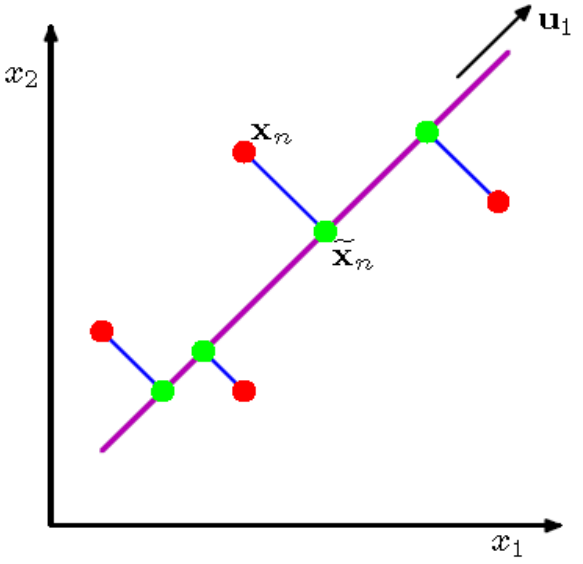


Figure 2: Example projection of 2D data X down to 1D representation. (Bishop)

Dimensionality Reduction

- **Step 1:**

  Compute sample mean / variance:

$$\mu = \frac{1}{N}\sum_{n=1}^{N} x_n \qquad S = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu)(x_n - \mu)^T$$

- **Step 2:**
  Compute M eigenvectors of S with largest eigenvalues $U = (u_1, u_2, \ldots, u_M)$
- **Step 3:**
  Project to low dimension: $z_n = x_n^T U$

Dimensionality Reduction

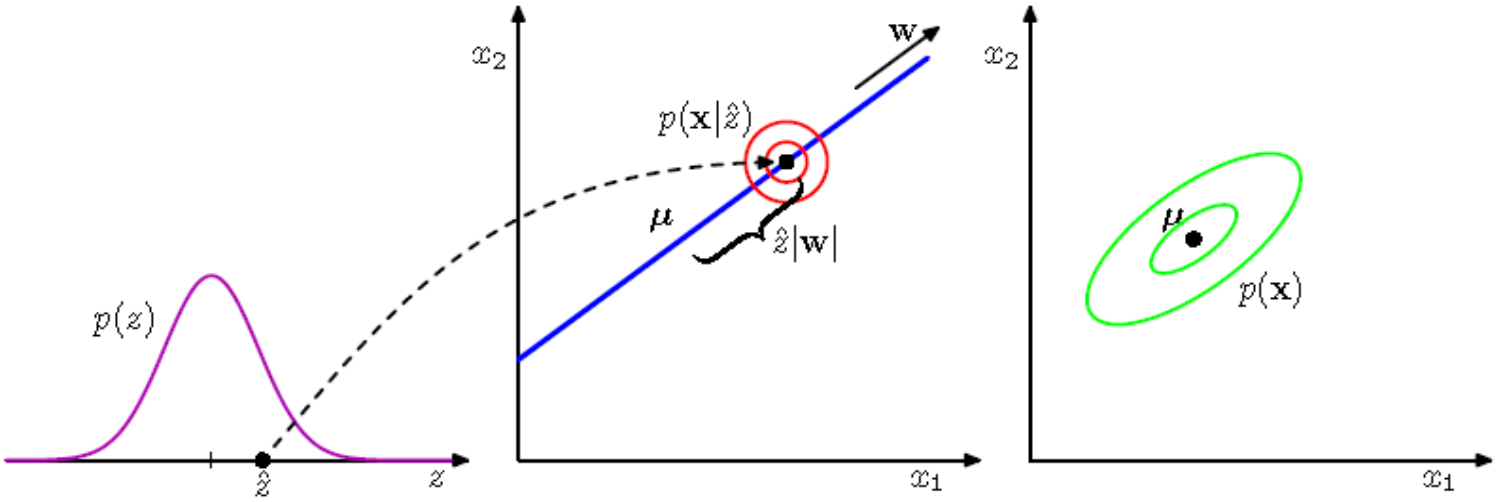


Figure 3: Illustration of the generative process of PPCA for two dimensional data and a one dimensional projection. (Bishop)

Probabilistic PCA (PPCA):

- Probabilistic extension of PCA
- Models high dimensional data as “noisy” projections of low dimensional data
- Assumes *spherical* Gaussian variance

Dimensionality Reduction

- PPCA models projection as,

$$x = Wz + \mu + \epsilon$$

where,

$$z \sim N(0, I)$$
$$\epsilon \sim N(0, \sigma^2 I_D)$$
$$x \mid z \sim N(Wz + \mu, \sigma^2 I_D)$$

- Need to learn parameters W and σ^2
- Closed form ML solution available, but can use EM if sample covariance is too large

Dimensionality Reduction

Factor Analysis (FA):

- Similar to PPCA, but allows *diagonal covariance*

$$x \mid z \sim N(Wz + \mu, \Psi)$$

- No closed form ML solution for parameters, can use EM.
- Note:
  - In PPCA & FA all *rotations* of input have the same likelihood (e.g. basis is not meaningful)
  - In FA all element-wise rescalings of input have equal likelihood (good for data at different scales, e.g. inches vs. feet)

Hidden Markov Model

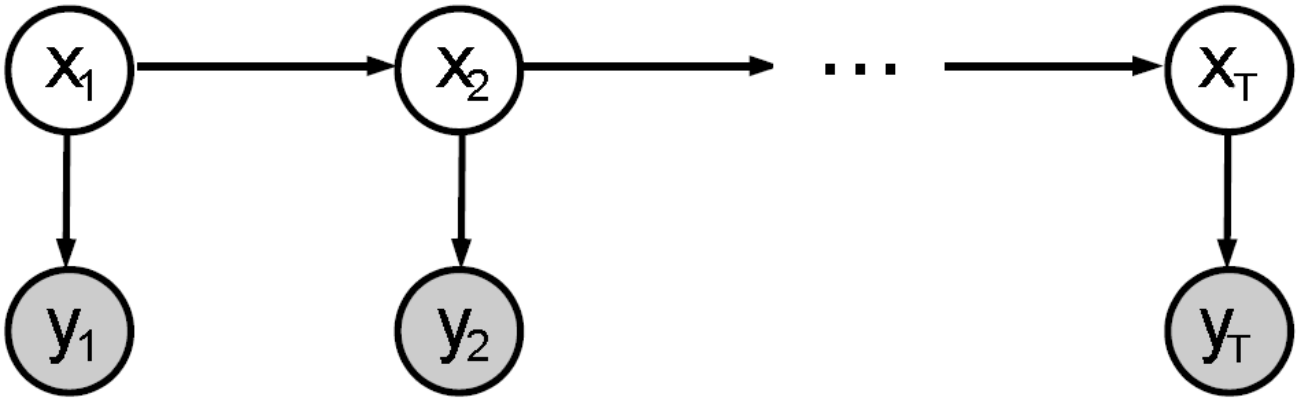


Figure 4: Example HMM for latent state X and observations Y

- Used to model *sequential data* where x_t depends only on x_{t-1}
- Used as extension to mixture model where mixture assignments are not iid, but ordered
- Defined by:

$$p(x_t \mid x_{t-1}) \quad : \quad \text{Transition probability}$$

$$p(y_t \mid x_t) \quad : \quad \text{Emission probability}$$

Forward Backward

- Used to perform inference in HMMs
- Compute forward/backward *messages*

$$\alpha(x_t) = p(x_t, y_1, \ldots, y_T) \quad : \quad \text{Forward message}$$

$$\beta(x_t) = p(y_{t+1}, \ldots, Y_t \mid x_t) \quad : \quad \text{Backward message}$$

- Compute messages recursively as,

$$\alpha(x_t) = p(y_t \mid x_t) \sum_{x_{t-1}} \alpha(x_{t-1}) p(x_t \mid x_{t-1})$$

$$\beta(x_t) = \sum_{x_{t+1}} \beta(x_{t+1}) p(y_{t+1} \mid x_{t+1}) p(x_{t+1} \mid x_t)$$

- Multiply them to yield posterior marginals,

$$p(x_t \mid y_1, \ldots, y_T) \propto \alpha(x_t)\beta(x_t)$$