Lecture 11:
Boosting and Feature Selection

Nando de Freitas
www.cs.ubc.ca/~nando/340-2009/
November 2009

About boosting

Boosting is a powerful algorithm design paradigm:

• It enables us to combine many weak learners to produce a strong learner.

• It can be easily applied to nonlinear regression or classification and is a brilliant tool for feature selection.

• It is easy to understand, implement and optimize.

• The slides used in this lecture use material from *Peter Bühlmann and Bin Yu (2009). Boosting. Wiley Interdisciplinary Reviews: Computational Statistics.*

Problem setup

Suppose that we observe

$$(X_1, Y_1), \ldots, (X_n, Y_n),$$

where $X_i \in \mathbb{R}^p$ denotes a p-dimensional predictor variable and Y_i a univariate response, for example taking values in $\mathbb{R}$ as in regression or in $\{-1, +1\}$ as in binary classification. In the sequel, we denote by $X^{(j)}$ the jth component of a vector $X \in \mathbb{R}^p$. We usually assume that the pairs (X_i, Y_i) are i.i.d. or from a stationary process. The goal is to estimate the regression function $F(x) = \mathbb{E}[Y|X = x]$ or to find a classifier $\text{sign}(F(x))$ where $F(x) : \mathbb{R}^p \to \mathbb{R}$.

Expected and Empirical Loss

loss = error = cost = -log Likelihood

The estimation performance is evaluated via a real-valued loss function in the sense that we want to minimize the expected loss or risk:

$$\mathbb{E}L(Y, F(X)), = \iint L(y, F(x))\, P(x,y)\, dx\, dy$$

based on data $(X_i, Y_i)(i = 1, ..., n)$. The loss function L is assumed to be smooth and convex in the second argument so that the gradient method can be applied. Boosting algorithms are obtainable by minimizing the empirical loss function

$$n^{-1} \sum_{i=1}^{n} L(Y_i, F(X_i)),$$

eg.

$$L(y_i, F(x_i)) = \sum_{i=1}^{n} (y_i - F(x_i))^2 \frac{1}{n}$$

From weak learners to strong ones

The boosting methodology in general builds on a user-determined base procedure or weak learner and uses it repeatedly on modified data which are typically outputs from the previous iterations. The final boosted procedure takes the form of linear combinations of the base procedures. Precisely, given a base learner $h(x, \theta)$, boosting is derivable as functional gradient descent on the loss function L.

Boosting (gradient descent view)

1. Start with $F_0(x) = 0$.
2. Given $F_{m-1}(x)$, let

$$F_{m-1}(x) = \sum_{j=1}^{m-1} \beta_j h_j(x, \theta_j)$$

$$(\beta_m, h(x, \hat{\theta}_m))) = argmin_{\beta \in \mathbb{R}, \theta} \sum_{i=1}^{n} L(Y_i, F_{m-1}(X_i) + \underset{m}{\beta} h(x, \underset{m}{\theta})).$$

3. Set

$$F_m(x) = F_{m-1}(x) + \beta_m h(x, \hat{\theta}_m).$$

4. Stop when $m = M$.

The AdaBoost classifier is $\text{sign}(F_M(x))$.

L2Boosting: coordinatewise descent for linear models

1. Start with $F_0 = 0$.

2. Given $F_{m-1}(x)$, Compute residuals $U_i = Y_i - F_{m-1}(X_i)$ $(i = 1, \ldots, n)$. Let $X_i^{(j)}$ be the jth component of $X_i \in \mathbb{R}^p$,

$$\hat{j}_m = argmin_{j=1,\ldots,p} \sum_{i=1}^{n} (U_i - \hat{\beta}_m X_i^{(j)}))^2,$$

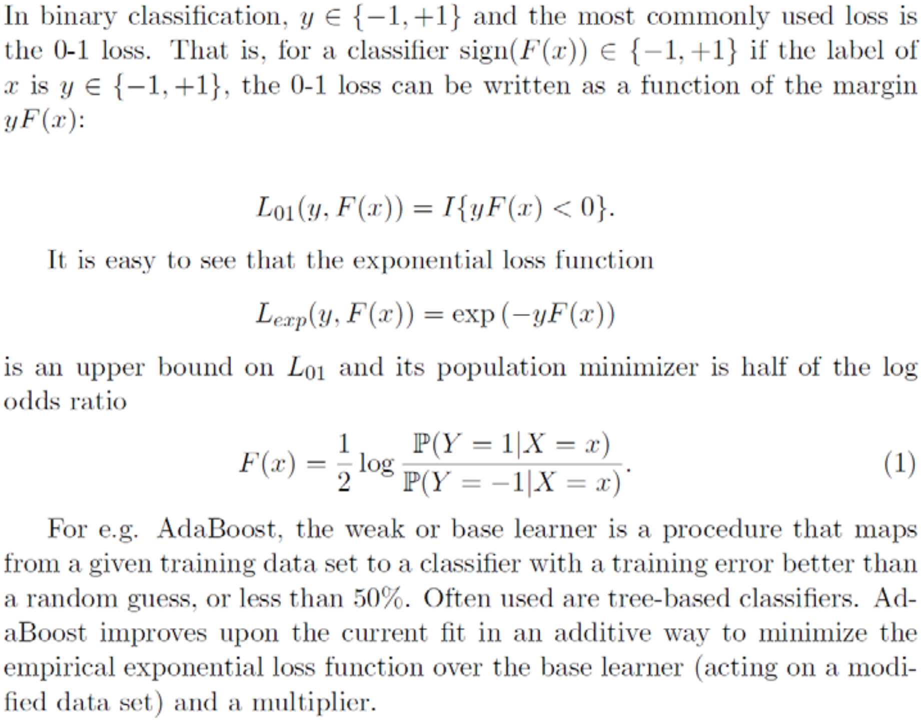
$$\hat{\beta}_m = argmin_{\beta} \sum_{i=1}^{n} (U_i - \beta X_i^{(j_m)})^2,$$

3.

$$F_m(x) = F_{m-1}(x) + \hat{\beta}_m X^{(\hat{j}_m)},$$

4. Stop when $m = M$ and $F_M(x)$ is the final estimator of the linear regression function.

Adaboost



In binary classification, $y \in \{-1, +1\}$ and the most commonly used loss is the 0-1 loss. That is, for a classifier $\text{sign}(F(x)) \in \{-1, +1\}$ if the label of x is $y \in \{-1, +1\}$, the 0-1 loss can be written as a function of the margin $yF(x)$:

$$L_{01}(y, F(x)) = I\{yF(x) < 0\}.$$

It is easy to see that the exponential loss function

$$L_{exp}(y, F(x)) = \exp(-yF(x))$$

is an upper bound on L_{01} and its population minimizer is half of the log odds ratio

$$F(x) = \frac{1}{2} \log \frac{\mathbb{P}(Y = 1|X = x)}{\mathbb{P}(Y = -1|X = x)}. \quad (1)$$

For e.g. AdaBoost, the weak or base learner is a procedure that maps from a given training data set to a classifier with a training error better than a random guess, or less than 50%. Often used are tree-based classifiers. AdaBoost improves upon the current fit in an additive way to minimize the empirical exponential loss function over the base learner (acting on a modified data set) and a multiplier.

AdaBoost

1. Start with $F_0(x) = 0$;
2. Given $F_{m-1}(x)$, let
$$w_i^{(m)} = \exp(-Y_i F_{m-1}(X_i)),$$
$$h(x, \hat{\theta}_m)) = argmin_\theta \sum_{i=1}^{n} w_i^{(m)} I(Y_i \neq h(X_i, \theta)),$$
and denote $h(\cdot, \hat{\theta}_m)$'s associated error by
$$err_m = \frac{\sum_{i=1}^{n} w_i^{(m)} I(Y_i \neq h(X_i, \hat{\theta}_m))}{\sum_{i=1}^{n} w_i^{(m)}}.$$
Furthermore let
$$\beta_m = \frac{1}{2} \log \frac{1 - err_m}{err_m}.$$
3. Set
$$F_m(x) = F_{m-1}(x) + \beta_m h(x, \hat{\theta}_m).$$
4. Stop when $m = M$.
5. The AdaBoost classifier is $y = \text{sign}(F_M(x))$.

Lecture 12: Unsupervised learning

Nando de Freitas
www.cs.ubc.ca/~nando/340-2009/

November 2009

Outline

In the absence of labels ***y***, there are many useful patterns and structures that we can still find in the data. For example, we might be interested in:

- Novelty detection (e.g. detecting new strands of HIV).

- Data association (e.g. machine translation, multi-target tracking, object recognition from annotated images).

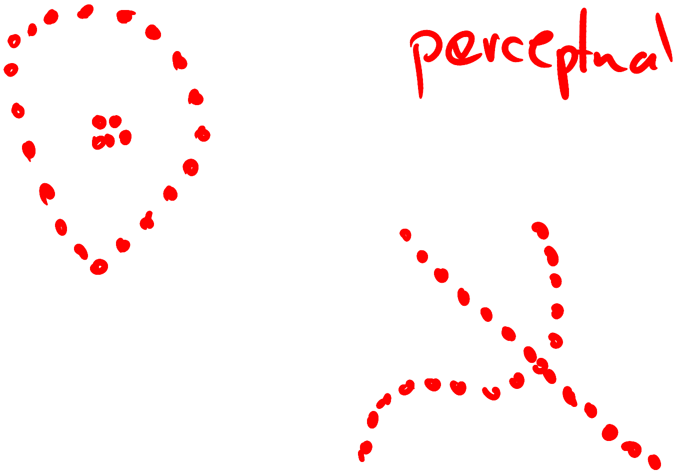
- Clustering (grouping similar items together).

In this lecture, we will introduce two of the most popular algorithms in machine learning and data mining: K-means and EM.

Clustering

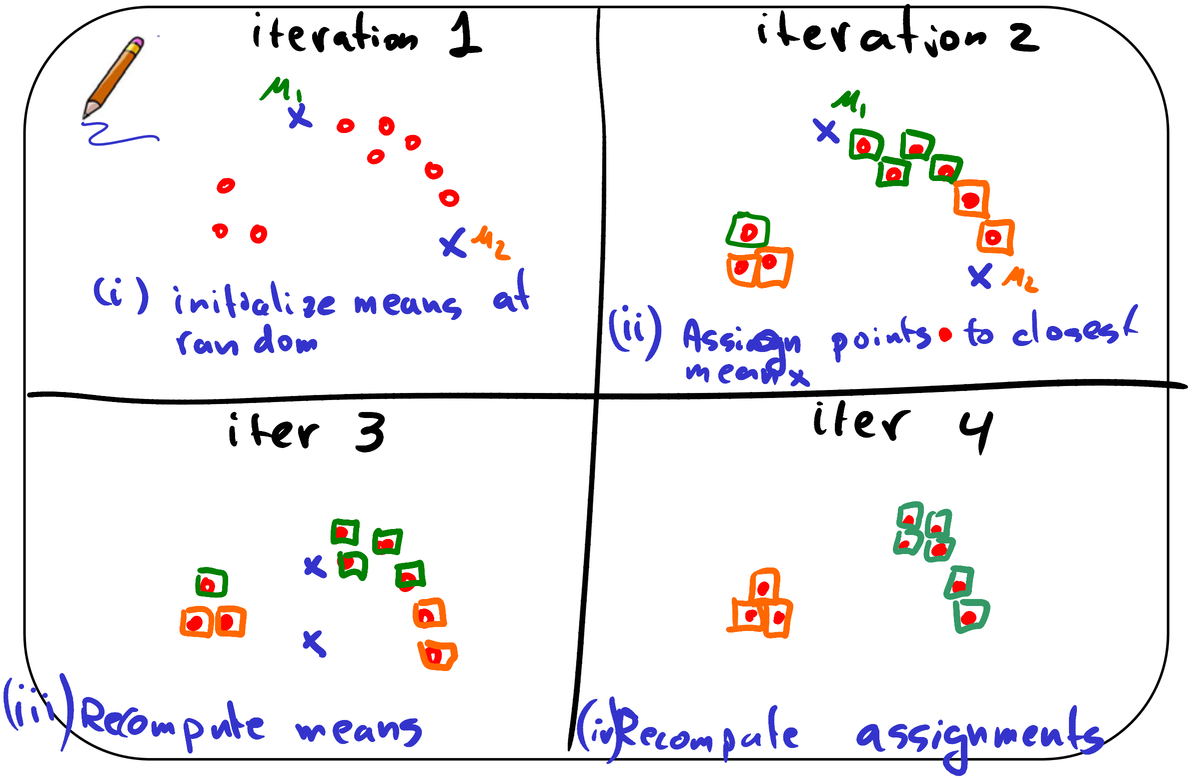
Gestalt

perceptual grouping



K-means

K=2 clusters



(i) initialize means at random

(ii) Assign points • to closest mean x

(iii) Recompute means

(iv) Recompute assignments

K-means algorithm

1. **Initialisation:** Choose $k = 2$ means $\mu_{1:2}$ at random.
2. **Compute distances:** For $c = 1, \ldots, k$ and $i = 1, \ldots, n$ compute the distance $\|x_i - \mu_c\|^2$.
3. **Assign data to nearest mean:** To keep track of assignments, introduce the indicator variable z_i, such that

$$\mathbb{I}_c(z_i) = \begin{cases} 1 & \text{if } c = \arg\min_{c'} \|x_i - \mu_{c'}\|^2 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbb{I}_c(z) = \begin{array}{ccc|c} c_1 & c_2 & c_3 & \\ \hline 1 & 0 & 0 & x_1 \\ 1 & 0 & 0 & x_2 \\ 0 & 1 & 0 & x_3 \\ 0 & 1 & 0 & x_4 \\ 0 & 0 & 1 & x_5 \\ 1 & 0 & 0 & x_6 \end{array}$$

That is, $\mathbb{I}_2(z_i) = 1$ if observation x_i is closer to cluster 2. $\mathbb{I}_c(z_i)$ end up being the entries of an $n \times k$ matrix with only one 1 per row and many zeros.

K-means algorithm (continued)

$\sum_i \mathbb{I}_c(z_i) =$ # points in cluster c

4. **Update means:**

$$\mu_c = \frac{\sum_{i=1}^{n} \mathbb{I}_c(z_i) x_i}{\sum_{i=1}^{n} \mathbb{I}_c(z_i)}$$

5. **Repeat:** Go back to step 2. until the means and assignments stop changing.

Hard Vs Soft assignments

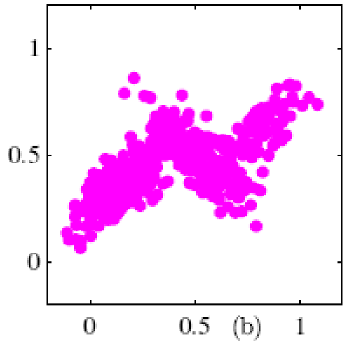
The problem with this algorithm is that the assignments are hard. Something is either this or that. Sometimes, however, we would like to say that something is this with probability 0.7 or that with probability 0.3.

We would like to find not only the means, but also the variances of each cluster and the probabilities of belonging to each cluster.

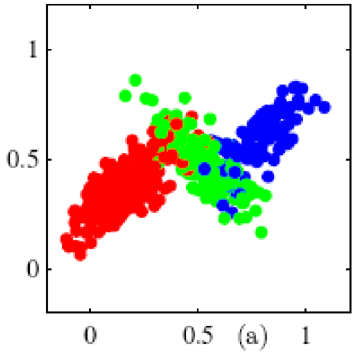
Clustering

Desired output

Input

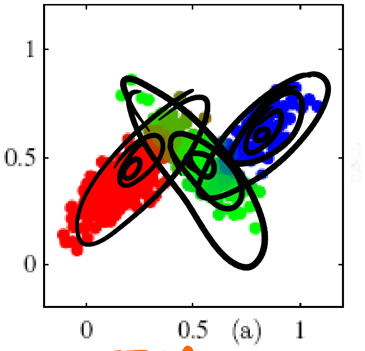


Hard labeling



K-MEANS

Soft labeling



EM
&
MIXTURES

K=3 is the number of clusters, here chosen by hand

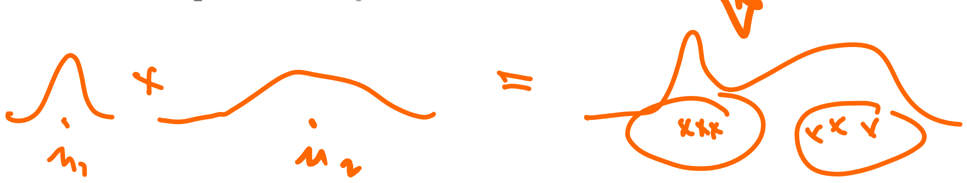
Probabilistic approach

For the 2 clusters, we approximate the probability of each data point with a weighted combination of Gaussians

Mixture:

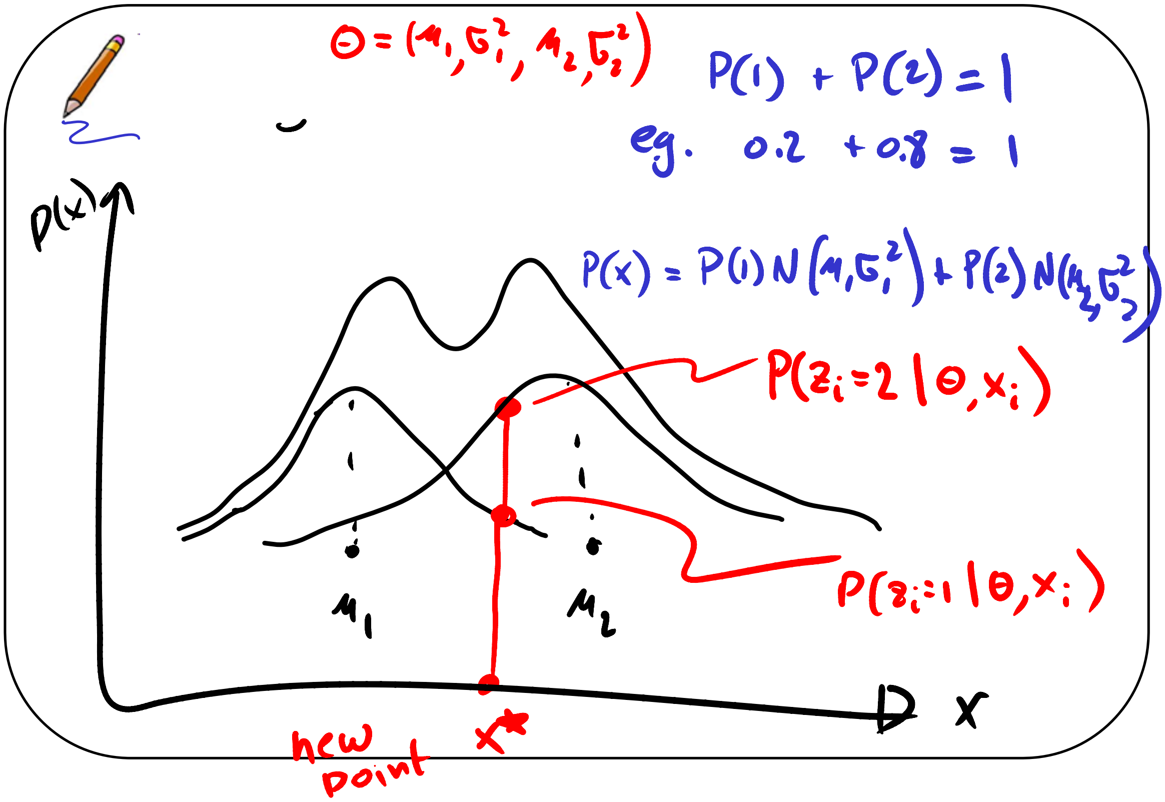
$$p(x_i|\mu_{1:2},\sigma_{1:2}) = \overset{\text{cluster 1}}{p(z_i=1)\mathcal{N}(x_i|\mu_1,\sigma_1^2)} + \overset{\text{cluster 2}}{p(z_i=2)\mathcal{N}(x_i|\mu_2,\sigma_2^2)}$$

Here, the unknown parameters are $(\mu_{1:2},\sigma_{1:2}^2)$ and the cluster probabilities $p(z_i=1)$ and $p(z_i=2)$, which we rewrite as $p(1)$ and $p(2)$ for brevity. Note that $p(1)+p(2)=1$ to ensure that we still have a probability.



Probabilistic approach in 2D

Probabilistic approach in 1D


$\Theta = (\mu_1, \sigma_1^2, \mu_2, \sigma_2^2)$
$P(1) + P(2) = 1$
eg. $0.2 + 0.8 = 1$
$p(x)$
$P(x) = P(1) N(\mu_1, \sigma_1^2) + P(2) N(\mu_2, \sigma_2^2)$
$P(z_i = 2 \mid \Theta, x_i)$
$P(z_i = 1 \mid \Theta, x_i)$
μ_1
μ_2
x
x^*
new point

Probabilistic approach

In general, we have

$$\boxed{\underline{p(x_i|\theta)} = \sum_{c=1}^{k} p(c)\mathcal{N}(x_i|\mu_c, \sigma_c^2)}$$

where $\theta = (\mu_{1:c}, \sigma_{1:c}^2)$ summarises the model parameters and $p(c) = p(z_i = c)$. Clearly, $\sum_{c=1}^{k} p(c) = 1$.

The EM algorithm

In this section, we use intuition to introduce the expectation-maximisation (EM). If we know $\mathbb{I}_c(z_i)$, then it is easy to compute (μ_c, σ_c^2) by maximum likelihood. We repeat this for each cluster. The problem is that we have a chicken and egg situation. To know the cluster memberships, we need the parameters of the Gaussians. To know the parameters, we need the cluster memberships.

One solution is to approximate $\mathbb{I}_c(z_i)$ with our expectation of it given the data and our current estimate of the parameters θ. That is, we replace $\mathbb{I}_c(z_i)$ with

$$\xi_{ic} \triangleq \mathbb{E}\left[\mathbb{I}_c(z_i)|x_i, \theta\right]$$

$$P(x_i|z_i=c, \theta) = \mathcal{N}(\mu_c, \sigma_c^2)$$

$$\xi_{ic} \triangleq \mathbb{E}\left[\mathbb{I}_c(z_i)|x_i, \theta\right] = P(z_i=c|x_i, \theta)$$

$$= \frac{P(z_i=c, x_i|\theta)}{P(x_i|\theta)}$$

$$= \frac{P(z_i=c)\,P(x_i|z_i=c, \theta)}{\underbrace{\sum_{c'} P(z_i=c)\,P(x_i|z_i=c, \theta)}_{P(x_i|\theta)}}$$

The EM algorithm

Once we know ξ_{ic}, we can compute the Gaussian mixture parameters:

$$\begin{aligned}
\mu_c &= \frac{\sum_{i=1}^n \xi_{ic} x_i}{\sum_{i=1}^n \xi_{ic}} \\
\Sigma_c &= \frac{\sum_{i=1}^n \xi_{ic}(x_i - \mu_c)(x_i - \mu_c)'}{\sum_{i=1}^n \xi_{ic}} \\
p(c) &= \frac{1}{n}\sum_{i=1}^n \xi_{ic}
\end{aligned}$$

EXAM

2 iterations!

The EM algorithm

The EM for Gaussians is as follows:

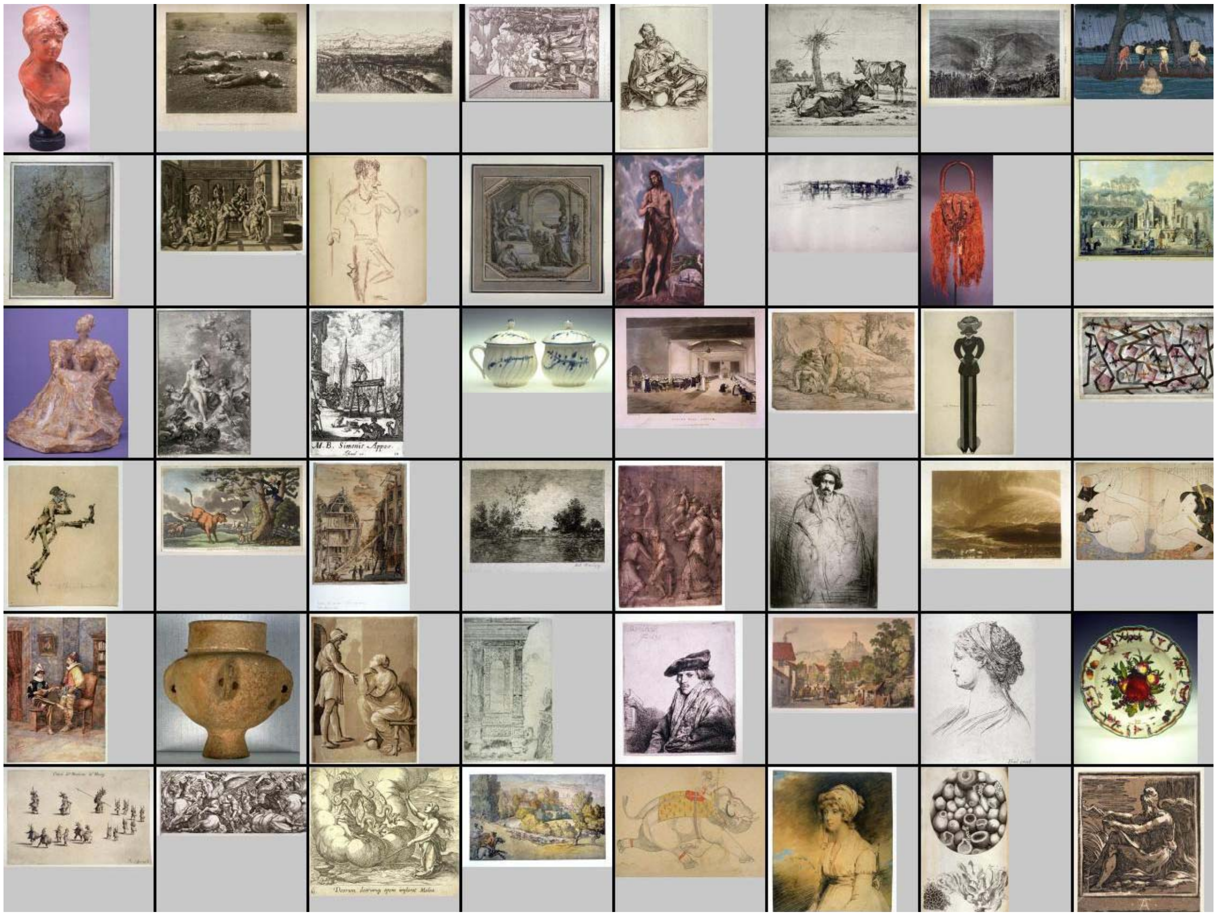
1. **Initialise.**

2. **E Step:** At iteration t, compute the expectation of the indicators for each i and c:

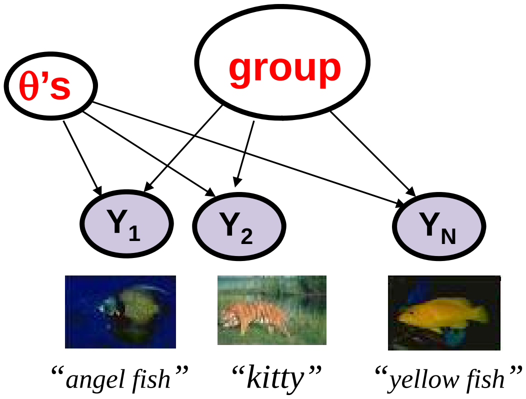
$$\xi_{ic}^{(t)} = \frac{p(c)^{(t)}\mathcal{N}(x_i|\mu_c^{(t)}, \Sigma_c^{(t)})}{\sum_{c'=1}^k p(c')^{(t)}\mathcal{N}(x_i|\mu_{c'}^{(t)}, \Sigma_{c'}^{(t)})}$$

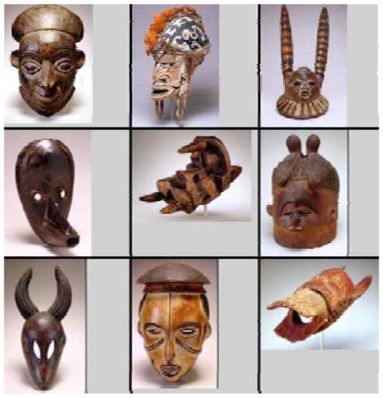
   and normalise it (divide by sum over c).

3. **M Step:** Update the parameters $p(c)^{(t)}, \mu_c^{(t)}, \Sigma_c^{(t)}$.

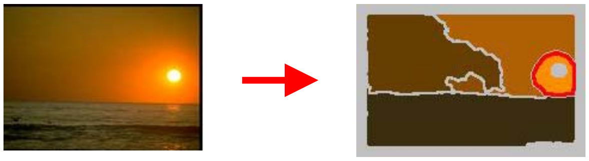


Clustering

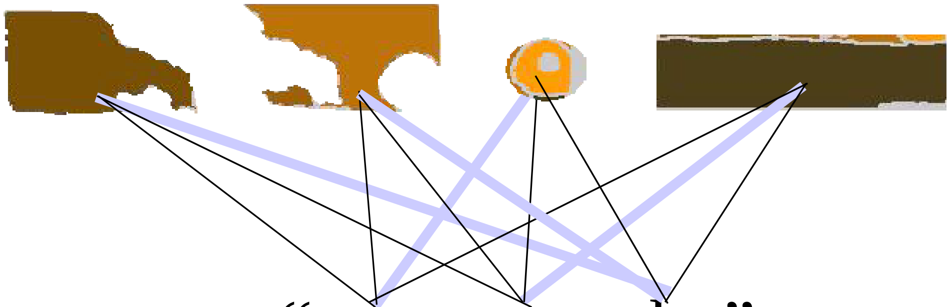




Translation and data association



“sun sea sky”



“sun sea sky”