# Lecture 8 - *Classification and Support Vector Machines*

**OBJECTIVE:** Introduce classification and support vector machines.

### A Simple Linear Classifier

Consider the following classification task:




That is, only one of $c$ outputs (classes) is 1 and the remaining outputs are 0. We could use our standard linear regression estimators with:

$$\begin{bmatrix} y_{11} & \cdots & y_{1c} \\ \vdots & \vdots & \vdots \\ y_{n1} & \cdots & y_{nc} \end{bmatrix} = \begin{bmatrix} x_{10} & \cdots & x_{1d} \\ \vdots & \vdots & \vdots \\ x_{n0} & \cdots & x_{nd} \end{bmatrix} \begin{bmatrix} \theta_{01} & \cdots & \theta_{0c} \\ \vdots & \vdots & \vdots \\ \theta_{d1} & \cdots & \theta_{dc} \end{bmatrix}$$

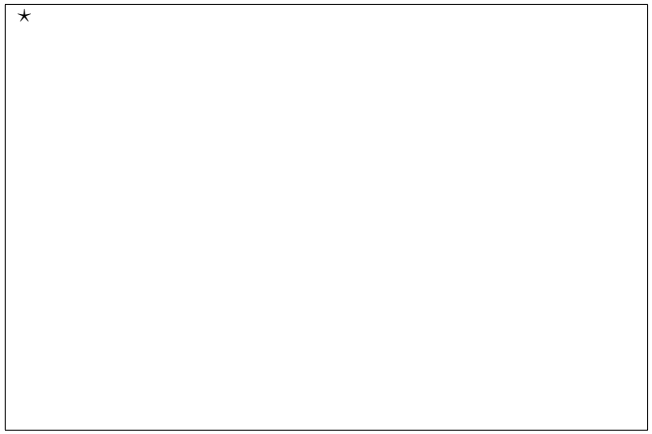
We can then generate predictions $\widehat{Y} = X\widehat{\theta}$ and choose a class according to

$$class = \arg\max_c \quad \widehat{Y}_c$$

This is an easy to implement classifier. One can start the analysis using it. However, we will learn more sophisticated ways of doing this.

## Support Vector Machines

Assume we are given the training data $\{\mathbf{x}_i, y_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$. For example, for $d = 2$:




For a new point $\mathbf{x}^\star$(without label), the optimal linear classifier is:

$$\widehat{y}(\mathbf{x}^\star) = sign(\mathbf{x}^\star \boldsymbol{\theta} + b)$$

and it should maximise the **margin**.



As in ridge, we introduce the dual parameters $\alpha$ and recast the problem in terms of dot products:

$$\widehat{y} = sign\left[\sum_{i=1}^{N_s} \alpha_i y_i \mathbf{x}^\star \mathbf{x}_i^T + b\right]$$

It can be shown (this is the topic of a more advanced class[1]) that the $\boldsymbol{\alpha}$'s that ensure that each input is assigned to the right class, while ensuring that the margin is maximized is given by the solution to the following quadratic programming cost function:

$$\min_{C \geq \boldsymbol{\alpha} \geq 0} \frac{1}{2} \boldsymbol{\alpha}^T diag(y) \mathbf{X}\mathbf{X}^T \text{diag}\,(y) \boldsymbol{\alpha} - \mathbf{1}^T \boldsymbol{\alpha}$$

where $C$ is a very large upper-bound, $\boldsymbol{\alpha} \in \mathbb{R}^n$ are the unknown coefficients, $\mathbf{1}$ is a vector of ones, and $diag(y) \in \mathbb{R}^{n \times n}$ is a diagonal matrix with the vector of training labels $\mathbf{y}$ on the diagonal.

[1] Alex Smola and Bernard Schőlkopf., "A Tutorial on Support Vector Regression.", Neuro-COLT Technical Report NC-TR-98-030, Royal Holloway College, University of London, UK, 1998.

In matlab, $\boldsymbol{\alpha}$ is obtained by calling the function quadprog:

```
[alpha, fval, exitFlag] =

quadprog(diag(y)*(X*X')*diag(y),-ones(N,1),[],[],y',0,zeros(N,1),C*ones(N,1));
```

One important property of SVMs is that the weights $\alpha$ are equal to zero for points that are not near the boundary as these far away points do not affect the shape of the classifier. (The Karush-Kuhn-Tucker Theorem tells us that only the $\alpha_i$ corresponding to points in the boundaries can be different from 0.) That is, only the $N_s$ points in the boundaries are needed for specifying the optimal classifier. These $N_s$ points are called the **support vectors**.



Finally, to compute $b$, we proceed as follows. First, we compute it for each training data point using the estimate of $\boldsymbol{\alpha}$:

$$b_i = y_i - \sum_{j=1}^{N_s} \alpha_j y_j \mathbf{x}_i \mathbf{x}_j^T$$

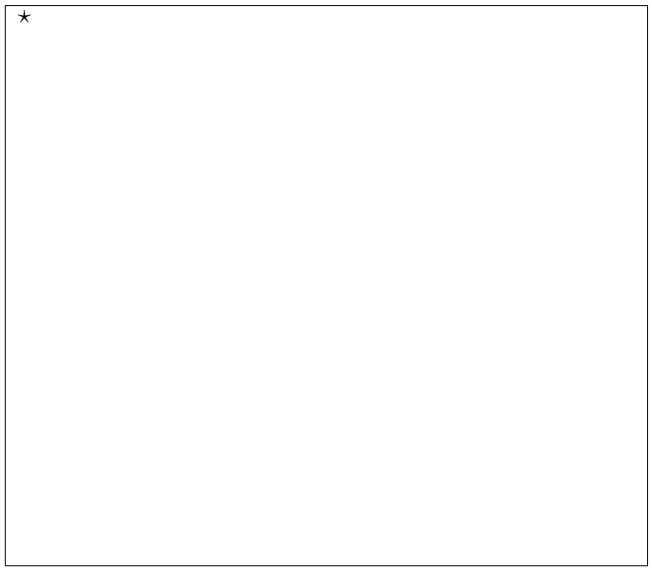
Next, we average

$$b = \frac{1}{N_s} \sum_{i=1}^{N_s} b_i$$

or take the median. Note that we only need to store the $N_s$ support vectors and not the entire data set.

## Nonlinear SVMs

Note that $x$ allways appears in the form of a dot product $x_i' x_j$. This enables us to use the kernel tricks again to transform nonlinear classification in low dimensions to linear classification in high dimensions.






In the nonlinear setting, the optimization problem is:

$$\min_{C \geq \alpha \geq 0} \frac{1}{2} \boldsymbol{\alpha}^T diag(y) \mathbf{K} \text{diag}\,(y) \boldsymbol{\alpha} - \mathbf{1}^T \boldsymbol{\alpha}$$

and the classifications are given by:

$$\widehat{y} = sign\left[\sum_{i=1}^{N_s} \alpha_i y_i k(\mathbf{x}^\star, \mathbf{x}_i) + b\right]$$

# Lecture 10 - *Unsupervised Learning*

**OBJECTIVE:** When there are no labels $\mathbf{y}$, there are many useful patterns we can still find in the data. Here we introduce two of the most popular algorithms in machine learning and data mining: K-means and EM.

### Unsupervised Learning

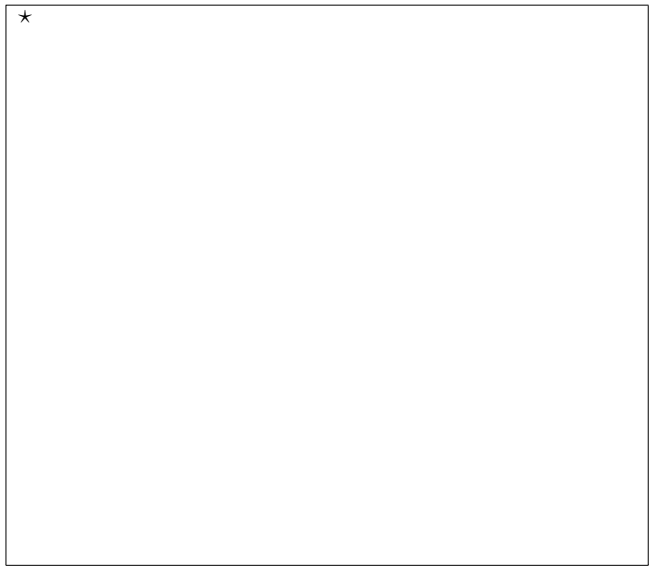
Our goal is to automatically discover patterns and structure in the data. Some examples include

- 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).



### Clustering

Assume we are given the data $x_{1:n}$, with $x_i \in \mathbb{R}^2$.

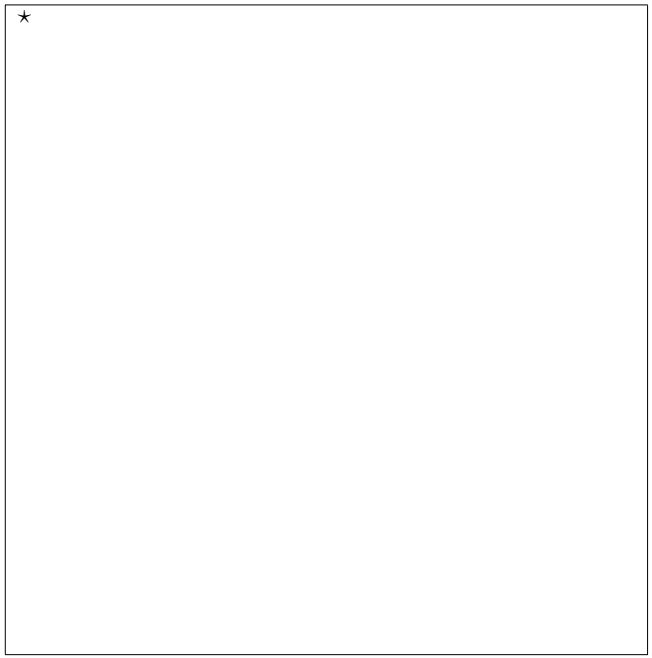


We want to find clusters (groups where data items are similar).

## K-means

K-means is a simple iterative algorithm for clustering data.

In our 2D example, it proceeds as follows:





1. **Initialisation:** Choose $k = 2$ means $\mu_{1:2}$ at random.
2. **Compute distances:** For $c = 1, \dots, k$ and $i = 1, \dots, n$ 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

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

   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.
4. **Update means:**

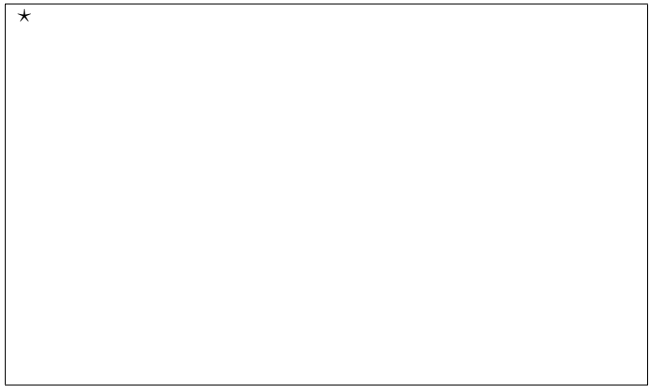
$$\mu_c = \frac{\sum_{i=1}^n \mathbb{I}_c(z_i) x_i}{\sum_{i=1}^n \mathbb{I}_c(z_i)}$$

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

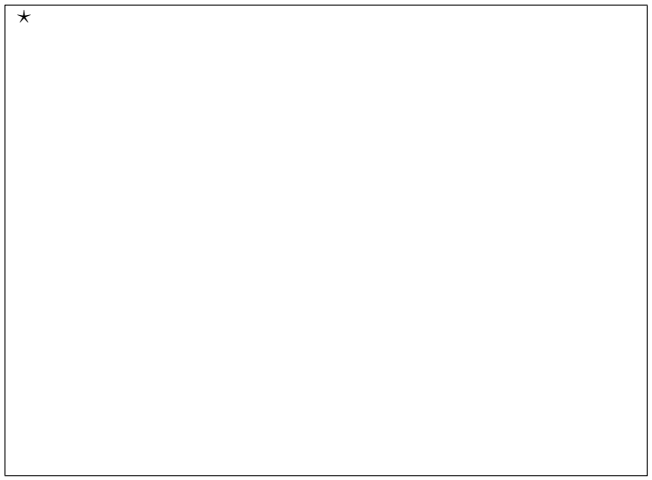
The problem with this algorithm is that the assignments are hard. Something is either this or that. Sometimes, however, we would like to say that something is this with probability 0.7 or that with probability 0.3.

### Finite Mixtures of Gaussians

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







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

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

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.

In general, we have

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

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

## Probability Review

Before introducing EM, we need to review some key concepts from probability: conditioning, marginalisation, Bayes rule and multinomial (discrete) distributions. This material will be useful for us to discuss models for multimodal (eg music and text) search engines and for us to design probabilistic expert systems (of of the core ideas in modern AI).



## Conditional Probability

$$P(A|B) \triangleq \frac{P(AB)}{P(B)}$$

where $P(A|B)$ is the **conditional probability** of $A$ given that $B$ occurs, $P(B)$ is the **marginal probability** of $B$ and $P(AB)$ is the **joint probability** of $A$ and $B$. In general, we obtain a **chain rule**

$$P(A_{1:n}) = P(A_n|A_{1:n-1})P(A_{n-1}|A_{1:n-2})\dots P(A_2|A_1)P(A_1)$$

⋆ Assume we have an urn with 3 red balls and 1 blue ball: $U = \{r, r, r, b\}$. What is the probability of drawing (without replacement) 2 red balls in the first 2 tries?

## Marginalisation

Let the sets $B_{1:n}$ be disjoint and let their union cover the set of all possible events $\Omega$, that is $\bigcup_{i=1}^{n} B_i = \Omega$. Then

$$P(A) = \sum_{i=1}^{n} P(A, B_i)$$



⋆ What is the probability that the second ball drawn from our urn will be red?

## Bayes Rule

Bayes rule allows us to reverse probabilities:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Combinining this with marginalisation, we obtain a powerful tool for statistical modelling:

$$P(model_i|data) = \frac{P(data|model_i)P(model_i)}{\sum_{j=1}^{M} P(data|model_j)P(model_j)}$$

That is, if we have **prior** probabilities for each model and generative data models, we can compute how likely each model is **a posteriori** (in light of our prior knowledge and the evidence brought in by the data).

### Discrete random variables

Let E be a discrete set, e.g. $E = \{0, 1\}$. A **discrete random variable** (r.v.) is a map from $\Omega$ to $E$:

$$X(w) : \Omega \mapsto E \qquad \text{where} w \in \Omega$$



⋆ Assume we are throwing a die and are interested in the events $E = \{even, odd\}$. Here $\Omega = \{1, 2, 3, 4, 5, 6\}$. The r.v. takes the value $X(w) = even$ if $w \in \{2, 4, 6\}$ and $X(w) = odd$ if $w \in \{1, 3, 5\}$. We describe this r.v. with a **probability distribution** $p(even) = P(X = even)$

The **cumulative distribution function** is defined as $F(x) = P(X \leq x)$ and would for this example be:

⋆

## Bernoulli Random Variables

Let $E = \{0, 1\}$, $P(X = 1) = \lambda$, and $P(X = 0) = 1 - \lambda$.

We now introduce the *set indicator variable*. (This is a very useful notation.)

$$\mathbb{I}_A(w) = \begin{cases} 1 \;\; if & w \in A; \\ 0 \;\; otherwise. & \end{cases}$$

Using this convention, the probability distribution of a **Bernoulli** random variable reads:

$$p(x) = \lambda^{\mathbb{I}_{\{1\}}(x)}(1 - \lambda)^{\mathbb{I}_{\{0\}}(x)}.$$



⋆

## Multinomial Distribution

Now suppose we have a set of $k$ boxes and a set of $n$ balls. Each ball is dropped from a certain height and will fall randomly into one of the boxes. The probability that a ball falls into box $i$ is given by $\lambda_i$. The $\lambda_i$s are constrained such that

$$\sum_{i=1}^{k} \lambda_i = 1, \quad 0 \leq \lambda_i \leq 1$$

If we define $X_i = m_i$ to be the event that there are $m_i$ balls in box $i$ then the probability that the $n$ balls are distributed in the boxes in a specified arrangement $m_{1:k} = (m_1, m_2, \ldots, m_k)$ is

$$p(m_{1:k}) = \left(\frac{n!}{m_1! m_2! \ldots m_k!}\right) \prod_{i=1}^{k} \lambda_i^{m_i} \quad \text{where} \quad \sum_{i=1}^{k} m_i = n$$

The term to the left of the product is the *normalizing term* and we need not worry about it here. Note that the binomial is a subcase of the Multinomial; if we set $k = 2$, $\lambda_1 = \lambda$ and $\lambda_2 = (1 - \lambda)$ then we get the binomial distribution.



⋆ Say we are given a text "*I am a teacher I I teacher*".
We convert this text to a frequency table $T = [3\ 1\ 1\ 2]$.
The probability $p(T)$ for this text is

where $\lambda_i$ is the probability of the $i$-th word appearing in the text.

Instead of just one text, we usually have a database of texts. We use this database to learn the word probabilities. This is usefull for clustering documents and designing information retrieval algorithms.

We also encounter frequency tables with missing values

$$\begin{bmatrix} 3 & 5 & 6 & 2 \\ 0 & ? & 3 & 1 \\ 1 & 2 & ? & 6 \end{bmatrix}$$

These tables arise in data mining and collaborative filtering. Here the columns represent items (say movies) and each row has the movie ratings assigned by a particular movie expert. By learning the Multinomial parameters, we will be able to complete the table and to recommend movies to viewers.



## EM for Mixtures of Gaussians

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 $(\mu_c, \sigma_c^2)$ 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 $\theta$. That is, we replace $\mathbb{I}_c(z_i)$ with

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

⋆

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

Note that $\xi_{ic}$ is a soft-assignments matrix:

⋆



Once we know $\xi_{ic}$, we can compute the Gaussian mixture parameters:

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

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 $p(c)^{(t)}, \mu_c^{(t)}, \Sigma_c^{(t)}$.

## EM for Categorical Data

Matrix data with categorical entries is ubiquitous on the web. Examples include word-document matrices used for information retrieval and item-judge ratings for collaborative filtering:

$$\begin{matrix} & words_{1:L} \\ documents_{1:n} & \begin{bmatrix} 3 & 1 & \dots & 6 \\ 7 & 4 & \dots & 2 \\ \vdots & & \ddots & \end{bmatrix} \end{matrix} = T_{1:n}$$

$$\begin{matrix} & judges_{1:L} \\ movies_{1:n} & \begin{bmatrix} 1 & ? & \dots & 5 \\ 5 & 4 & \dots & ? \\ \vdots & & \ddots & \end{bmatrix} \end{matrix} = M_{1:n}$$

Each row of $N$ can be modelled with a multinomial distribution.

$$p(T_i) \propto \prod_{w=1}^{n_w} \delta_w^{T_{iw}}$$



We assume that the text documents are i.i.d.

$$p(T_{1:n}) = \prod_{i=1}^{n} p(T_i)$$

Suppose the documents come from $n_c$ distinct clusters. Then they are distributed according to a mixture of multinomials:

$$p(T_i|\theta) = \sum_{c=1}^{n_c} p(c) \prod_{w=1}^{n_w} \delta_{wc}^{T_{iw}}$$

The maximum likelihood EM for Categorical data involves iterating the following two steps:

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

$$\xi_{ic} \propto p(c) \prod_{w=1}^{n_w} \delta_{wc}^{T_{iw}}$$

   and normalise it (divide by sum over $c$).

2. **M Step:** Update the parameters:

$$\begin{aligned} \delta_{wc} &= \frac{\sum_{i=1}^{n} \xi_{ic} T_{iw}}{\sum_{i=1}^{n} \sum_{w=1}^{n_w} T_{iw} \xi_{ic}} \\ p(c) &= \frac{1}{n} \sum_{i=1}^{n} \xi_{ic} \end{aligned}$$

### EM for Multimodal Data

A document with an image (represented by a vector $x_i \in \mathbb{R}^d$) and text $T_i$ can be modelled as follows:

$$p(d_i|\theta) = \sum_{c=1}^{n_c} p(c) \mathcal{N}(x_i|\mu_c, \Sigma_c) \prod_{w=1}^{n_w} \delta_{wc}^{T_{iw}}$$

The maximum likelihood EM is as follows:

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

$$\xi_{ic} \propto p(c) \mathcal{N}(x_i|\mu_c, \Sigma_c) \prod_{w=1}^{n_w} \delta_{wc}^{T_{iw}}$$

and normalise it (divide by sum over $c$).



2. **M Step:** Update the parameters:

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