Lecture 2 — October 11

Lecturer: Guillaume Obozinski *Scribes: Aymeric Reshef, Claire Vernade*

- Course webpage: `http://www.di.ens.fr/~fbach/courses/fall2013/`

2.1 Single node models (last part)

The previous course introduced the notion of *Maximum Likelihood Estimator* (MLE). Basic examples on Bernoulli model, multinomial model and Gaussian model were explicited, and side notes detailed the use of Lagrangian operators and of differentials. The last example was using the multivariate Gaussian model. We recall it briefly in the next subsection.

2.1.1 The Multivariate Gaussian model

If X is a random variable taking values in $\mathbb{R}^d$. Let $\mu \in \mathbb{R}^d$ and $\Sigma \in \mathbb{R}^{d\times d}$ be a positive definite matrix. X follows a *multivariate Gaussian model* (denoted by $X \sim \mathcal{N}(\mu, \Sigma)$) if

$$p_{\mu,\Sigma}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d \det \Sigma}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu)\right).$$

Let $X_1, \cdots, X_n \sim \mathcal{N}(\mu, \Sigma)$, *iid*. Then, the negative *log-likelihood* of the joint distribution is

$$\begin{aligned} -l(\mu,\Sigma) &= -\sum_{i=1}^n \log p_{\mu,\Sigma}(\mathbf{x}_i) \\ &= \frac{nd}{2}\log(2\pi) + \frac{n}{2}\log(\det\Sigma) + \frac{1}{2}\sum_{i=1}^n (\mathbf{x}_i-\mu)^\top \Sigma^{-1}(\mathbf{x}_i-\mu). \end{aligned}$$

Its gradient with respect to μ is given by

$$\begin{aligned} -\nabla_\mu l(\mu,\Sigma) &= \sum_{i=1}^n \Sigma^{-1}(\mathbf{x}_i-\mu) \\ &= \Sigma^{-1}\left(\sum_{i=1}^n x_i - n\mu\right) = \Sigma^{-1}(n\bar{x}-n\mu), \end{aligned}$$

which leads to $\boxed{\hat{\mu} = \frac{1}{n}\bar{x}}$, the empirical mean.

In order to compute the gradient with respect to Σ, we first write $A = \Sigma^{-1}$, so that

$$\begin{aligned} -l(\mu,\Sigma) &= \frac{nd}{2}\log(2\pi) - \frac{n}{2}\log(\det A) + \frac{1}{2}\sum_{i=1}^n (\mathbf{x}_i-\mu)^\top A(\mathbf{x}_i-\mu) \\ &= \frac{nd}{2}\log(2\pi) - \frac{n}{2}\log(\det A) + \frac{n}{2}\mathrm{Tr}(A\tilde{\Sigma}), \end{aligned}$$

where we introduced the empirical covariance matrix $\tilde{\Sigma}$ defined as

$$\tilde{\Sigma} = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}_i - \mu)^\top (\mathbf{x}_i - \mu).$$

The matrix A appears in the expression of the log-likelihood in two terms: $\frac{n}{2}\log\det A$ and $\frac{n}{2}\operatorname{Tr}(A\tilde{\Sigma})$.
Denote by $f(A) = \operatorname{Tr}(A\tilde{\Sigma})$. Then $f(A+H) - f(A) = \operatorname{Tr}(H\tilde{\Sigma})$, which leads to $\nabla f(A) = \tilde{\Sigma}$. Now, write $\log\det A$ as

$$\log\det(A+H) = \log\det\left(A^{\frac{1}{2}}\left(I + A^{-\frac{1}{2}}HA^{-\frac{1}{2}}\right)A^{\frac{1}{2}}\right) = \log\det A + \log\det(I+\tilde{H})$$

where $A^{\frac{1}{2}}$ stands for the square root matrix of A (it exists, since A is positive definite) and $\tilde{H} = A^{-\frac{1}{2}}HA^{-\frac{1}{2}}$. Let's see how $\log\det(I+\tilde{H})$ looks like. Noting that $\log\det I = 0$, and denoting by $(\lambda_1, \cdots, \lambda_d)$ the eigenvalues of $\tilde{H}$, we have that

$$\log\det(I+\tilde{H}) = \log\det(I+\tilde{H}) - \log\det I = \sum_{j=1}^{d}\log(1+\lambda_j) \approx \sum_{j=1}^{d}\lambda_j + o(\|\tilde{H}\|).$$

But then,

$$\sum_{j=1}^{d}\lambda_j = \operatorname{Tr}(\tilde{H}) = \operatorname{Tr}(A^{-\frac{1}{2}}HA^{-\frac{1}{2}}) = \operatorname{Tr}(HA^{-1}).$$

We conclude that $\nabla_A \log\det A = A^{-1}$.

Plugging these results into the gradient of the log-likelihood with respect to A, we have

$$\nabla_A l(A) = -\frac{n}{2}A^{-1} + \frac{n}{2}\tilde{\Sigma}.$$

The optimality condition $\nabla_A l(A)$ leads to $A^{-1} = \tilde{\Sigma}$, which means that

$$\boxed{\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \mu)^\top(\mathbf{x}_i - \mu)}$$

is the empirical covariance matrix.

Note that we assumed that A was invertible, which is an implicit condition when writing $\log\det A$. This implies that in a rigorous sense the maximum likelihood estimator is undefined when $\tilde{\Sigma}$ is not invertible. In practice, the MLE is extended by continuity to the rank deficient case.

2.2 Models with two nodes

In this section, we work with two nodes: one node corresponds to an input X, and one node corresponds to an output Y.

Recall that when dealing with two random variables X and Y, one can use a *generative* model, i.e. which models the joint distribution $p(X,Y)$, or one can use instead a *conditional* model (often considered equivalent to the slightly different concept of *discriminative model*), which models the conditional probability of the output, given the input $p(Y|X)$. The two following models, *linear regression* or a *logistic regression*, are *conditional models*.

2.2.1 Linear regression

Let's assume that $Y \in \mathbb{R}$ depends linearly on $X \in \mathbb{R}^p$. Let $w \in \mathbb{R}^p$ be a weighting vector and $\sigma^2 > 0$. We make the following assumption:

$$Y \mid X \sim \mathcal{N}(\mathbf{w}^\top X, \sigma^2),$$

which can be rewritten as

$$Y = \mathbf{w}^\top X + \epsilon,$$

with $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Note that if there is an offset $w_0 \in \mathbb{R}^p$, that is, if $Y = \mathbf{w}^\top X + w_0 + \epsilon$, one can always redefine a weighting vector $\tilde{\mathbf{w}} \in \mathbb{R}^{p+1}$ such that

$$Y = \tilde{\mathbf{w}}^\top \begin{pmatrix} x \\ 1 \end{pmatrix} + \epsilon.$$

Let $\mathcal{D} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$ be a training set of i.i.d. random variables. Each y_i is a *label* (a decision) on observation $\mathbf{x}_i$. We consider the *conditional* distribution of all outputs given all inputs, which is a product of terms because of the independence of the pairs forming the training set:

$$p(y_1, \cdots, y_n | x_1, \cdots, x_n; \mathbf{w}, \sigma^2) = \prod_{i=1}^{n} p(y_i | \mathbf{x}_i; \mathbf{w}, \sigma^2).$$

The associated log-likelihood has the following expression:

$$-l(\mathbf{w}, \sigma^2) = -\sum_{i=1}^{n} \log p(y_i | \mathbf{x}_i) = \frac{n}{2} \log(2\pi\sigma^2) + \frac{1}{2} \sum_{i=1}^{n} \frac{(y_i - \mathbf{w}^\top \mathbf{x}_i)^2}{\sigma^2}.$$

The minimization problem with respect to w can now be reformulated as:

$$\text{find} \quad \hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \tfrac{1}{2n} \textstyle\sum_{i=1}^{n} (y_i - \mathbf{w}^\top \mathbf{x}_i)^2.$$

Define the so-called *design matrix* $\mathbf{X}$ as

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix} \in \mathbb{R}^{n \times p}$$

and denote by $\mathbf{y}$ the vector of coordinates $(y_1, \cdots, y_n)$. The minimization problem over w can be rewritten in a more compact way as:

$$\text{find} \quad \hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \tfrac{1}{2n} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2.$$

Let $f : w \mapsto \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 = \frac{1}{2n}(\mathbf{y}^\top\mathbf{y} - 2\mathbf{w}^\top\mathbf{X}^\top\mathbf{y} + \mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w}$. f is strictly convex if and only if its Hessian matrix is invertible. This is never the case when $n < p$ (in this case, we deal with underdetermined problems). Most of the time, the Hessian matrix is invertible when $n \geq p$. When this is not the case, we often use the Tikhonov regularization, which adds

a penalization of the ℓ_2-norm of w by minimizing $f(\mathbf{w}) + \lambda\|\mathbf{w}\|^2$ with some hyperparameter $\lambda > 0$.

The gradient of f is

$$\nabla f(\mathbf{w}) = \frac{1}{n}\mathbf{X}^\top(\mathbf{X}\mathbf{w} - \mathbf{y}) = 0 \iff \mathbf{X}^\top\mathbf{X}\mathbf{w} = \mathbf{X}^\top\mathbf{y}.$$

The equation $\boxed{\mathbf{X}^\top\mathbf{X}\mathbf{w} = \mathbf{X}^\top\mathbf{y}}$ is known as the *normal equation*. If $\mathbf{X}^\top\mathbf{X}$ is invertible, then the optimal weighting vector is

$$\boxed{\hat{\mathbf{w}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} = \mathbf{X}^\dagger\mathbf{y}}$$

where $\mathbf{X}^\dagger = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$ is the *Moore-Penrose pseudo-inverse* of X. If $\mathbf{X}^\top\mathbf{X}$ is not invertible, the solution is not unique anymore, and for any $\mathbf{h} \in \ker(\mathbf{X})$, $\hat{\mathbf{w}} = (\mathbf{X}^\top\mathbf{X})^\dagger\mathbf{X}^\top\mathbf{y} + \mathbf{h}$ is an admissible solution. In that case however it would be necessary to use regularization.

The computational cost to evaluate the optimal weighting vector from $\mathbf{X}$ and $\mathbf{y}$ is $O(p^3)$ (use a Cholesky decomposition of matrix $\mathbf{X}^\top\mathbf{X}$ and solve two triangular systems).

Now, let's differentiate $l(\mathbf{w}, \sigma^2)$ with respect to σ^2: we have

$$\nabla_{\sigma^2} l(\mathbf{w}, \sigma^2) = \frac{n}{2\sigma^2} - \frac{n}{2\sigma^4}\frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{w}^\top\mathbf{x}_i)^2.$$

Setting $\nabla_{\sigma^2} l(\mathbf{w}, \sigma^2)$ to zero gives

$$\boxed{\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{w}^\top\mathbf{x}_i)^2.}$$

In practice, whenever we use a data matrix $\mathbf{X}$ in machine learning, we first preprocess it to try and avoid that it would be too badly conditioned, so to avoid numerical issues. Two main operations are applied columnwise: first, a centering (remove the mean of the coefficients) and a normalization (divide coefficients from a column by the standard deviation of the column vector). Note that this preprocessing *does not guarantee* that the matrix we obtain is well-conditioned: in particular, it can be low rank...

2.2.2 Logistic regression

Let $X \in \mathbb{R}^p$, $Y \in \{0, 1\}$. We assume that Y follows a Bernoulli distribution with parameter θ. The problem is to find θ. Let's define the *sigmoid* function σ defined on the real axis and taking values in $[0, 1]$, such that

$$\forall z \in \mathbb{R}, \sigma(z) = \frac{1}{1 + e^{-z}}.$$

The sigmoid function is plot on Figure 2.1.

One can easily prove that

$$\begin{aligned} \forall z \in \mathbb{R}, \quad \sigma(-z) &= 1 - \sigma(z), \\ \forall z \in \mathbb{R}, \quad \sigma'(z) &= \sigma(z)(1 - \sigma(z)) = \sigma(z)\sigma(-z). \end{aligned}$$

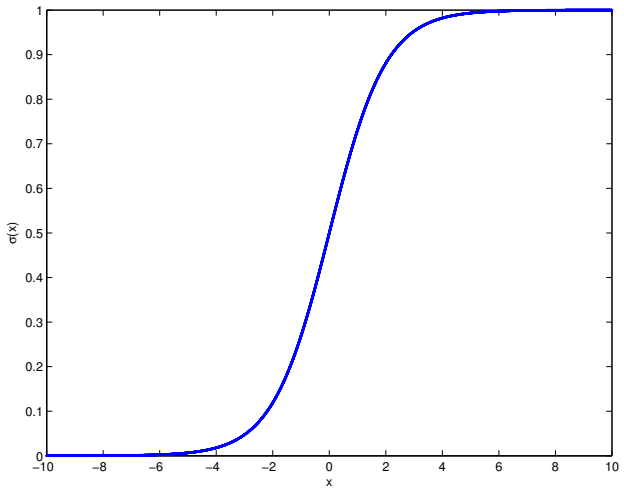

Figure 2.1. Sigmoid function.

We now assume that, for a given observation $X = \mathbf{x}$, the output $Y|X = \mathbf{x}$ follows a Bernoulli law with parameter $\theta = \sigma(\mathbf{w}^\top \mathbf{x})$, where w is again a weighting vector. In practice, we still can add an offset $\mathbf{w}^\top \mathbf{x} + w_0$. Then, the conditional distribution is given by

$$p(Y = y|X = \mathbf{x}) = \theta^y (1-\theta)^{1-y} = \sigma(\mathbf{w}^\top \mathbf{x})^y \sigma(-\mathbf{w}^\top \mathbf{x})^{1-y}.$$

Given a training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$ of *iid* random variables , we can compute the log-likelihood

$$l(\mathbf{w}) = \sum_{i=1}^{n} y_i \log \sigma(\mathbf{w}^\top \mathbf{x}_i) + (1 - y_i) \log \sigma(-\mathbf{w}^\top \mathbf{x}_i).$$

In order to minimize the log-likelihood, since $z \mapsto \log(1 + e^{-z})$ is a convex function and $\mathbf{w} \mapsto \mathbf{w}^\top \mathbf{x}_i$ is linear, we calculate its gradient. We write $\eta_i = \sigma(\theta^\top \mathbf{x}_i)$:

$$\nabla_{\mathbf{w}} l(\mathbf{w}) = \sum_{i=1}^{n} y_i \mathbf{x}_i \frac{\sigma(\mathbf{w}^\top \mathbf{x}_i)\sigma(-\mathbf{w}^\top \mathbf{x}_i)}{\sigma(\mathbf{w}^\top \mathbf{x}_i)} - (1 - y_i)\mathbf{x}_i \frac{\sigma(\mathbf{w}^\top \mathbf{x}_i)\sigma(-\mathbf{w}^\top \mathbf{x}_i)}{\sigma(-\mathbf{w}^\top \mathbf{x}_i)} = \sum_{i=1}^{n} \mathbf{x}_i (y_i - \eta_i)$$

Thus, $\nabla_{\mathbf{w}} l(\mathbf{w}) = 0 \iff \sum_{i=1}^{n} \mathbf{x}_i (y_i - \sigma(\theta^\top \mathbf{x}_i)) = 0$. This equation is nonlinear and we need an iterative optimization method to solve it. For this purpose, we derive the Hessian matrix of l:

$$Hl(\mathbf{w}) = \sum_{i=1}^{n} \mathbf{x}_i (0 - \sigma'(\mathbf{w}^\top \mathbf{x}_i)\sigma'(-\mathbf{w}^\top \mathbf{x}_i)\mathbf{x}_i^\top)$$

$$= \sum_{i=1}^{n} (-\eta_i(1 - \eta_i))\mathbf{x}_i \mathbf{x}_i^\top = -\mathbf{X}^\top \mathrm{Diag}(\eta_i(1 - \eta_i))\mathbf{X}$$

where $\mathbf{X}$ is the design matrix defined previously.

In the following we discuss first- and second-order optimization methods and apply them to logistic regression.

First-order methods

Let $f : \mathbb{R}^p \to \mathbb{R}$ be the convex C^1 function that we want to minimize. A *descent direction* at point $\mathbf{x}$ is a vector d such that $\langle \mathbf{d}, \nabla f(\mathbf{x}) \rangle < 0$. The minimization of f can be done by applying a *descent algorithm*, which iteratively takes a step in a descent direction, leading to an iterative scheme of the form

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \varepsilon^{(k)} \mathbf{d}^{(k)},$$

where $\varepsilon^{(k)}$ is the *stepsize*. The direction $\mathbf{d}^{(k)}$ is often chosen as the opposite of the gradient of f at point $\mathbf{x}^{(k)}$: $\mathbf{d}^{(k)} = -\nabla f(\mathbf{x}^{(k)})$.

There are several choices for $\varepsilon^{(k)}$:

1. Constant step: $\varepsilon^{(k)} = \varepsilon$. But the scheme does not necessarily converge.
2. Decreasing step size: $\varepsilon^{(k)} \propto \frac{1}{k}$ with $\sum_k \varepsilon^{(k)} = \infty$ and $\sum_k (\varepsilon^{(k)})^2 < \infty$. The scheme is guaranteed to converge.
3. One can determine $\varepsilon^{(k)}$ by doing a *Line Search* which tries to find $\min_\varepsilon f(\mathbf{x}^{(k)} + \varepsilon \mathbf{d}^{(k)})$:
   - either exactly but this is costly and rather useless in many situations;
   - or approximately (see the Armijo linesearch). This is a better method.

Second-order methods

This time, let $f : \mathbb{R}^p \to \mathbb{R}$ be the C^2 function that we want to minimize. We write the second-order Taylor-expansion of f:

$$f(\mathbf{x}) = f(\mathbf{x}^t) + (\mathbf{x} - \mathbf{x}^t)^\top \nabla f(\mathbf{x}^t) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^t)^\top Hf(\mathbf{x}^t)(\mathbf{x} - \mathbf{x}^t) + o(\left\|\mathbf{x} - \mathbf{x}^t\right\|^2) \stackrel{def}{=} g_t(\mathbf{x}) + (\left\|\mathbf{x} - \mathbf{x}^t\right\|^2)$$

A local optimum $\mathbf{x}^*$ is then reached when

$$\begin{cases} \nabla f(\mathbf{x}^*) = 0 \\ H(f(\mathbf{x}^*) \succeq 0 \end{cases}$$

In order to solve such a problem, we are going to use *Newton's method*. If f is a convex function, then $\nabla g_t(\mathbf{x}) = \nabla f(\mathbf{x}^t) + Hf(\mathbf{x}^t)(\mathbf{x} - \mathbf{x}^t)$ and we only need to find $\mathbf{x}^*$ so that $\nabla g_t(\mathbf{x}) = 0$, *ie.* we set $\mathbf{x}^{t+1} = \mathbf{x}^t - [Hf(\mathbf{x}^t]^{-1} \nabla f(\mathbf{x}^t)$. If the Hessian matrix is not invertible, we can regularize the problem and minimize $g_t(\mathbf{x}) + \lambda \|\mathbf{x} - \mathbf{x}^t\|^2$ instead.

In general the previous update, called the *Pure Newton step* does not lead to a convergent algorithm even if the function is convex!

In general it is necessary to use the so-called *Damped Newton method*, to obtain a convergent algorithm which consists in doing the following iterations:

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \varepsilon^t (Hf(\mathbf{x}^t))^{-1} \nabla f(\mathbf{x}^t),$$

where ε^t is set with the Armijo *Line Search*

This method may be computationally costly in high dimension because of the inverse of the hessian matrix that needs to be computed at each iteration. For some functions, however, the pure Newton's method does converge. This is the case for logistic regression.

In the context of non-convex optimization, the situation is more complicated because the Hessian can have negative eigenvalues. In that case, so-called trust region methods are typically used.

Application to logistic regression

We will write the form that Newton's algorithm takes for logistic regression. We had :

$$\begin{aligned} l(\mathbf{w}) &= \sum_{i=1}^{n} y_i \log \sigma(\mathbf{w}^\top \mathbf{x}_i) + (1-y_i) \log \sigma(-\mathbf{w}^\top \mathbf{x}_i) \\ \nabla_{\mathbf{w}} l(\mathbf{w}) &= \sum_{i=1}^{n} \mathbf{x}_i (y_i - \eta_i) = \mathbf{X}^\top (\mathbf{y} - \boldsymbol{\eta}) \\ Hl(\mathbf{w}) &= -\mathbf{X}^\top \mathrm{Diag}(\eta_i(1-\eta_i))\mathbf{X} \end{aligned}$$

The second-order Taylor expansion of the loss function leads to

$$l(\mathbf{w}) = l(\mathbf{w}^t) + (\mathbf{w} - \mathbf{w}^t)^\top \nabla l(\mathbf{w}^t) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^t)^\top Hl(\mathbf{w}^t)(\mathbf{w} - \mathbf{w}^t).$$

Let us set $\mathbf{h} = \mathbf{w} - \mathbf{w}^t$. The minimization problem becomes:

$$\min_{\mathbf{h}} \left\{ \mathbf{h}^\top \mathbf{X}^\top (\mathbf{y} - \boldsymbol{\eta}) - \frac{1}{2}\mathbf{h}^\top \mathbf{X}^\top \mathrm{Diag}(\eta(1-\eta))\mathbf{X}\mathbf{h} \right\} \iff \min_{\mathbf{h}} \mathbf{h}^\top \nabla_{\mathbf{w}} l(\mathbf{w}) + \frac{1}{2}\mathbf{h}^\top Hl(\mathbf{w})\mathbf{h}.$$

This leads, according to the previous part, to set $\mathbf{w}^{t+1} = \mathbf{w}^t + Hl(\mathbf{w}^t)^{-1}\nabla_{\mathbf{w}} l(\mathbf{w})$. The minimization problem above can be seen as some *weighted* linear regression over h of some function of the form $\sum_i \frac{(\tilde{y}_i - \tilde{\mathbf{x}_i}^\top \mathbf{h})^2}{\sigma_i^2}$, where $\tilde{y}_i = y_i - \eta_i$ and $\sigma_i^2 = [\eta_i(1-\eta_i)]^{-1}$. Thus, this method is often refered as the *iterative reweighted least squares* algorithm (IRLS).

We may run into a classification problem with more than two classes : $Y \in \{1, \cdots, K\}$ with $Y \sim \mathcal{M}(1, \pi_1(\mathbf{x}), \cdots, \pi_K(\mathbf{x}))$ where We will need to define a rule over the classifiers (softmax function, one-versus-all, etc.) in order to make a decision.

2.2.3 Generative models

This section briefly presents the *Fisher linear discriminant* also known as the linear discriminant analysis. Suppose that we have $X \in \mathbb{R}^p$ and $Y \in \{0, 1\}$.

$$P(Y = 1 \mid X = \mathbf{x}) = \frac{P(X = \mathbf{x} \mid Y = 1)P(Y = 1)}{P(X = \mathbf{x} \mid Y = 1)P(Y = 1) + P(X = \mathbf{x} \mid Y = 0)P(Y = 0)}$$

The assumption then consists in considering $P(X = \mathbf{x} \mid Y = 1) \sim \mathcal{N}(\mathbf{x}, \mu_1, \Sigma_1)$ and $P(X = \mathbf{x} \mid Y = 0) \sim \mathcal{N}(\mathbf{x}, \mu_0, \Sigma_0)$. Fisher's assumption is the assumption that $\Sigma_1 = \Sigma_0 = \Sigma$.

2.3 Unsupervised classification

Unsupervised learning consists in finding a label prediction function based on unlabeled training data only. In the case where the learning problem is a classification problem, and under the assumption that the classes form clusters in input space, the problem reduces to a clustering problem, which consists in finding groups of points that form denser clusters. When the clusters are assumed to be isotropic the formulation of the K-means algorithm is appropriate.

The K-means algorithm

We start from a set of data points $(\mathbf{x}_1, \cdots, \mathbf{x}_n)$ (where $\mathbf{x}_i \in \mathbb{R}^p$), that are unlabelled. We wish to divide this set into K clusters defined by their centroids $(\mu_1, \cdots, \mu_K)$. The problem can be formulated as:

$$\min_{\mu_1,\cdots,\mu_K} \frac{1}{n} \sum_{i=1}^{n} \min_k \|\mathbf{x}_i - \mu_k\|^2.$$

The minimization step inside the summation leads to a nonconvex problem. The K-means algorithm is a greedy algorithm which consists in iteratively apply two steps:

$$\begin{aligned} C_k &\leftarrow \left\{ i \mid \|\mathbf{x}_i - \mu_k\|^2 = \min_j \|\mathbf{x}_i - \mu_j\|^2 \right\} \\ \mu_k &\leftarrow \frac{1}{\mid C_k \mid} \sum_{i \in C_k} \mathbf{x}_i. \end{aligned}$$

The first step defines the clusters C_k by assigning each data point to its closest centroid. The second step then updates the centroids given the new cluster.

Two remarks:

- It can be shown that K-means converges in a finite number of steps.
- The algorithm however typically get stuck in local minima and it practice it is necessary to try several restarts of the algorithm with a random initialization to have chances to obtain a better solution.