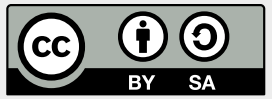
Naive Bayes, K-Means, Gaussian Mixture

Milan Straka

December 16, 2019

Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics



SVM For Regression

The idea of SVM for regression is to use an ε-insensitive error function

$$\mathcal{L}_\varepsilon\big(t, y(\boldsymbol{x})\big) = \max\big(0, |y(\boldsymbol{x}) - t| - \varepsilon\big).$$

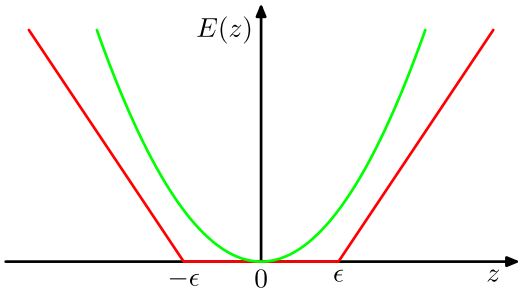


Figure 7.6 of Pattern Recognition and Machine Learning.

The primary formulation of the loss is then

$$C\sum_i \mathcal{L}_\varepsilon\big(t, y(\boldsymbol{x})\big) + \frac{1}{2}||\boldsymbol{w}||^2.$$

In the dual formulation, we ideally require every example to be withing ε of its target, but introduce two slack variables $\boldsymbol{\xi}^-$, $\boldsymbol{\xi}^+$ to allow outliers. We therefore minimize the loss

$$C\sum_i(\xi_i^- + \xi_i^+) + \tfrac{1}{2}||\boldsymbol{w}||^2$$

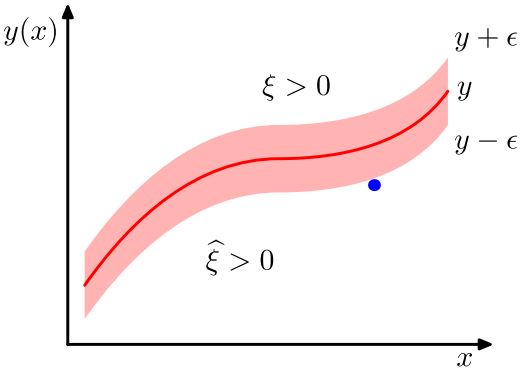


Figure 7.7 of Pattern Recognition and Machine Learning.

while requiring for every example $t_i - \varepsilon - \xi_i^- \leq y(\boldsymbol{x}) \leq t_i + \varepsilon + \xi_i^+$ for $\xi_i^- \geq 0, \xi_i^+ \geq 0$.

SVM For Regression

The Langrangian after substituting for $\boldsymbol{w}$, b, $\boldsymbol{\xi}^-$ and $\boldsymbol{\xi}^+$ we get that we want to minimize

$$L = \sum_i (a_i^+ - a_i^-) t_i - \varepsilon \sum_i (a_i^+ + a_i^-) - \frac{1}{2} \sum_i \sum_j (a_i^+ - a_i^-)(a_j^+ - a_j^-) K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

subject to

$$0 \leq a_i^+, a_i^- \leq C.$$

The prediction is then given by

$$y(\boldsymbol{z}) = \sum_i (a_i^+ - a_j^-) K(\boldsymbol{z}, \boldsymbol{x}_i) + b.$$

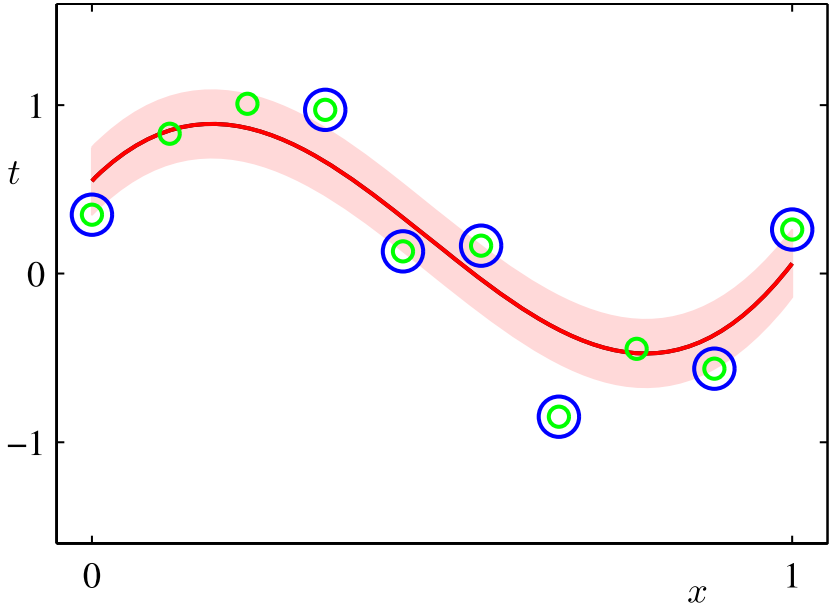


Figure 7.8 of Pattern Recognition and Machine Learning.

Term Frequency – Inverse Document Frequency

To represent a document, we might consider it a **bag of words**, and create a feature space with a dimension of every word. We might represent the words as:

- **binary indicators**: $1/0$ depending on whether a word is present in a document or not;
- **term frequency TF**: relative frequency of the term in the document;

$$TF(t) = \frac{\text{number of occurrences of } t \text{ in the document}}{\text{number of terms in the document}}$$

- **inverse document frequency IDF**: we might represent the term using its self-information, where terms with lower probability have higher weights;

$$IDF(t) = \log \frac{\text{number of documents}}{\text{number of documents containing term } t \big[\text{optionally} + 1\big]}$$

- **TF-IDF**: product of $TF(t)$ and $IDF(t)$.

Naive Bayes Classifier

Consider a discriminative classifier modelling probabilities

$$p(C_k|\boldsymbol{x}) = p(C_k|x_1, x_2, \ldots, x_D).$$

We might use Bayes theorem and rewrite it to

$$p(C_k|\boldsymbol{x}) = \frac{p(C_k)p(\boldsymbol{x}|C_k)}{p(\boldsymbol{x})}.$$

The so-called **Naive Bayes** classifier assumes all x_i are independent given C_k, so we can write

$$p(\boldsymbol{x}|C_k) = p(x_1|C_k)p(x_2|C_k, x_1)p(x_3|C_k, x_1, x_2)\cdots p(x_D|C_k, x_1, \ldots)$$

as

$$p(C_k|\boldsymbol{x}) \propto p(C_k)\prod_i p(x_i|C_k).$$

Naive Bayes Classifier

There are several used naive Bayes classifiers, depending on the distribution $p(x_i|C_k)$:

- **Gaussian NB**: the probability $p(x_i|C_k)$ is modelled as a normal distribution $\mathcal{N}(\mu_{i,k}, \sigma^2_{i,k})$;
- **Multinomial NB**: the probability $p(x_i|C_k)$ is proportional to $p_{i,k}^{x_i}$, so the

$$\log p(C_k, \boldsymbol{x}) = \log p(C_k) + \sum_i \log p_{i,k}^{x_i} = \log p(C_k) + \sum_i x_i \log p_{i,k} = b + \boldsymbol{x}^T \boldsymbol{w}$$

is a linear model in the log space with $b = \log p(C_k)$ and $w_i = \log p_{i,k}$. Denoting $n_{i,k}$ as the sum of features x_i for a class C_k, the probabilities $p_{i,k}$ are usually estimated as

$$p_{i,k} = \frac{n_{i,k} + \alpha}{\sum_j n_{j,k} + \alpha D}$$

where α is a *smoothing* parameter accounting for terms not appearing in any document of class C_k.

- **Bernoulli NB**: when the input features are binary, the $p(x_i|C_k)$ might also be a Bernoulli distribution

$$p(x_i|C_k) = p_{i,k}^{x_i} \cdot (1 - p_{i,k})^{(1-x_i)}.$$

Similarly to multinomial NB, the probabilities are usually estimated as

$$p_{i,k} = \frac{\text{number of documents of class } k \text{ with nonzero feature } i + \alpha}{\text{number of documents of class } k + 2\alpha}.$$

The difference with respect to Multinomial NB is that Bernoulli NB explicitly models also an *absence of terms*.

Given that a Multinomial/Bernoulli NB fits $p(C_k, \boldsymbol{x})$ as a linear model and a logistic regression fits $p(C_k|\boldsymbol{x})$ as a log-linear model, naive Bayes and logistic regression form a so-called *generative-discriminative* pair, where the naive Bayes is a *generative* model, while logistic regression is a *discriminative* model.

Multivariate Gaussian Distribution

Recall that

$$\mathcal{N}(x; \mu, \sigma^2) = \sqrt{\frac{1}{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

For D-dimensional vector $\boldsymbol{x}$, the multivariate Gaussian distribution takes the form

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \stackrel{\text{def}}{=} \frac{1}{\sqrt{(2\pi)^D |\Sigma|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right).$$

Multivariate Gaussian Distribution

The $\boldsymbol{\Sigma}$ is a *covariance* matrix, and it is symmetrical. If we represent it using its *eigenvectors* $\boldsymbol{u}_i$ and _eigenvalues λ_i, we get

$$\boldsymbol{\Sigma}^{-1} = \sum_i \frac{1}{\lambda_i} \boldsymbol{u}_i \boldsymbol{u}_i^T,$$

from which we can see that the constant surfaces of the multivariate Gaussian distribution are ellipsoids centered at $\boldsymbol{\mu}$, with axes oriented at $\boldsymbol{u}_i$ with scaling factors $\lambda_i^{1/2}$.

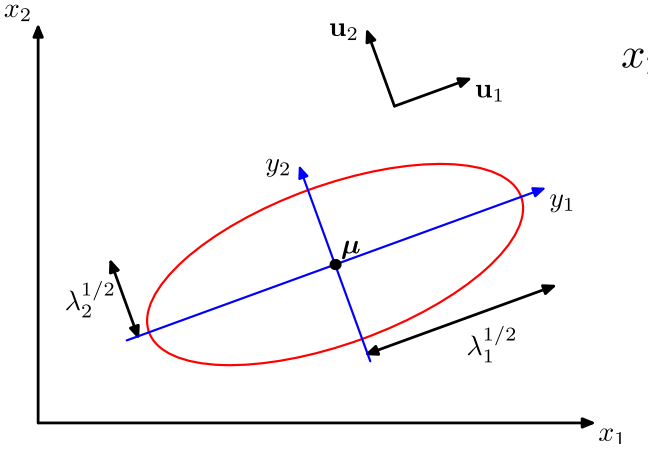


Figure 2.7 of Pattern Recognition and Machine Learning.

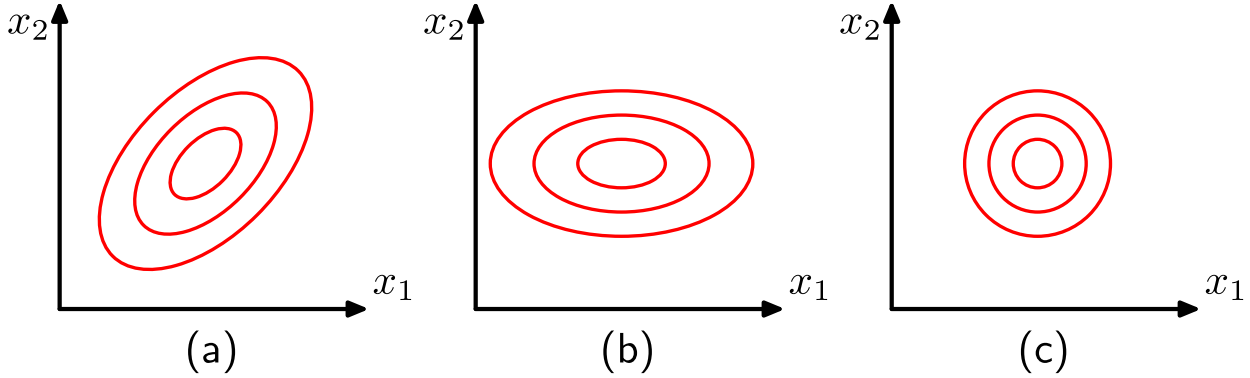


Figure 2.8 of Pattern Recognition and Machine Learning.

Clustering

Clustering is an unsupervised machine learning technique, which given input data tries to divide them into some number of groups, or *clusters*.

The number of clusters might be given in advance, or should also be inferred.

When clustering documents, we usually use TF-IDF normalized so that each feature vector has length 1 (i.e., L2 normalization).

K-Means Clustering

Let $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ be a collection of N input examples, each being a D-dimensional vector $\boldsymbol{x}_i \in \mathbb{R}^D$. Let K, the number of target clusters, be given.

Let each cluster be specified by a point $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$. Further, let $z_{i,k} \in \{0, 1\}$ be a binary indicator variables describing whether input example $\boldsymbol{x}_i$ is assigned to cluster k, and let each cluster be specified by a point $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$, usually called the cluster *center*.

Our objective function J which we aim to minimize is

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} z_{i,k} ||\boldsymbol{x}_i - \boldsymbol{\mu}_k||^2.$$

K-Means Clustering

To find out the cluster centers $\boldsymbol{\mu}_i$ and input example assignments $z_{i,k}$, we use the following iterative algorithm (which could be considered a coordinate descent):

1. compute the best possible $z_{i,k}$. It is easy to see that the smallest J is achieved by

$$z_{i,k} = \begin{cases} 1 & \text{if } k = \arg\min_j ||\boldsymbol{x}_i - \boldsymbol{\mu}_j||^2 \\ 0 & \text{otherwise.} \end{cases}$$

2. compute best possible $\boldsymbol{\mu}_k = \arg\min_{\boldsymbol{\mu}} \sum_i z_{i,k} ||\boldsymbol{x}_i - \boldsymbol{\mu}||^2$. By computing a derivative with respect to $\boldsymbol{\mu}$, we get

$$\boldsymbol{\mu}_k = \frac{\sum_i z_{i,k} \boldsymbol{x}_i}{\sum_i z_{i,k}}.$$

K-Means Clustering

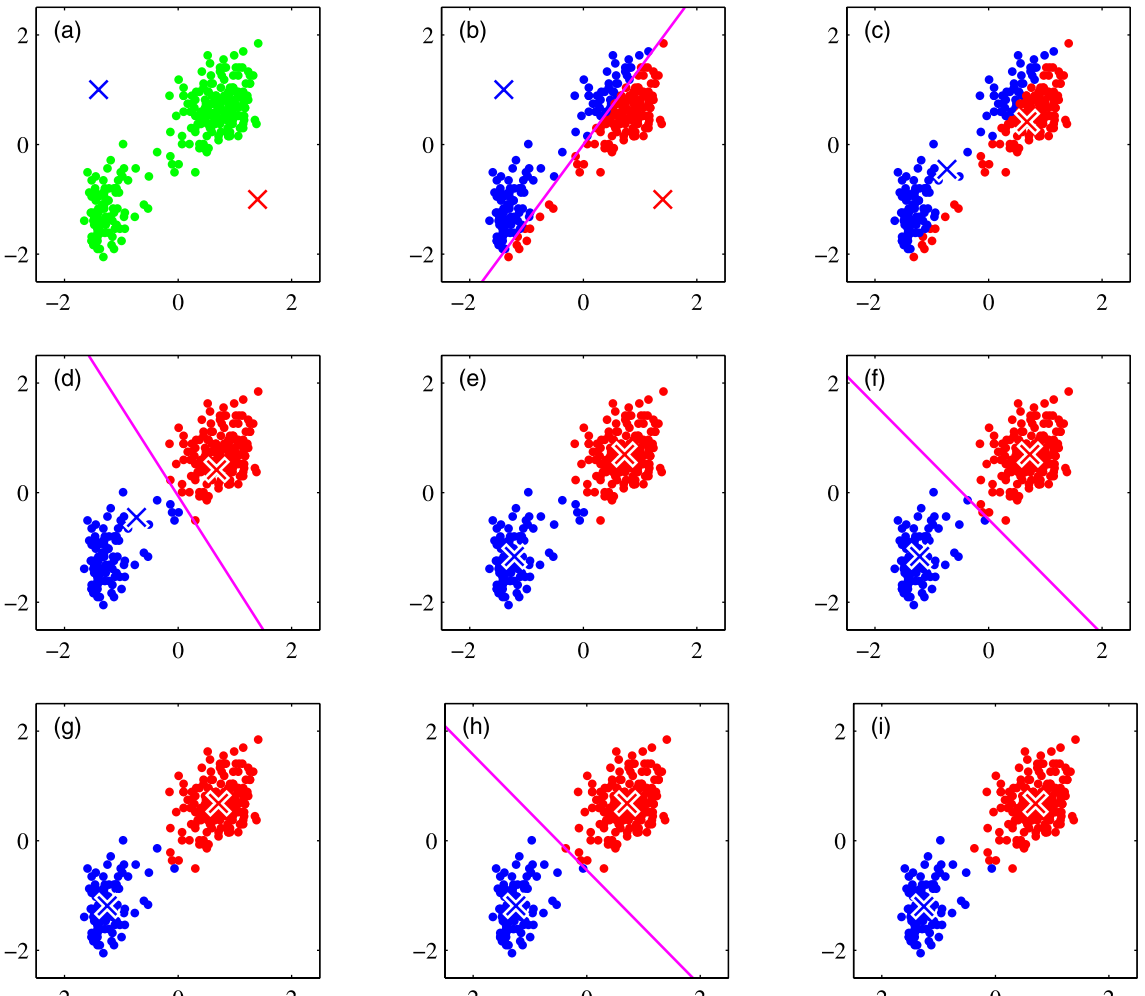


Figure 9.1 of Pattern Recognition and Machine Learning.

K-Means Clustering

It is easy to see that:

- updating the cluster assignment $z_{i,k}$ decreases the loss J or keeps it the same;
- updating the cluster centers again decreases the loss J or keeps it the same.

Plot of the cost function J given by (9.1) after each E step (blue points) and M step (red points) of the K-means algorithm for the example shown in Figure 9.1. The algorithm has converged after the third M step, and the final EM cycle produces no changes in either the assignments or the prototype vectors.

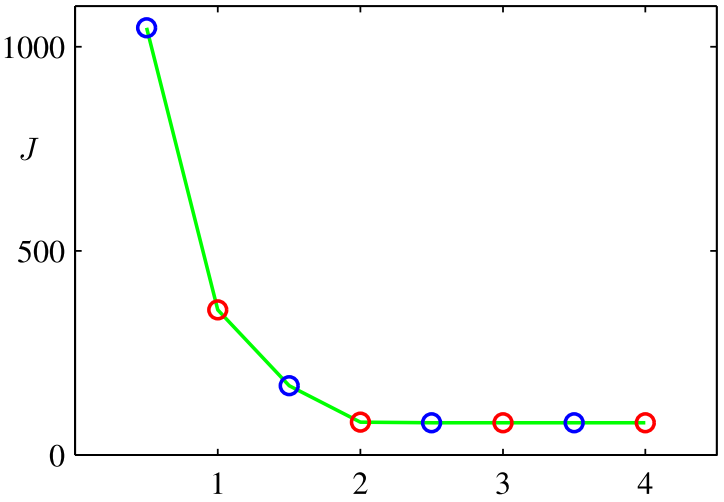


Figure 9.2 of Pattern Recognition and Machine Learning.

K-Means clustering therefore converges to a local optimum.
However, it is quite sensitive to the starting initialization:

- It is common practise to run K-Means algorithm multiple times with different initialization and use the result with lowest J (scikit-learn uses `n_init=10` by default).
- There exist better initialization schemes, a frequently used one is `k-means++`, where the first cluster center is chosen randomly and others are chosen proportionally to the square of their distance to the nearest cluster center.

K-Means Clustering

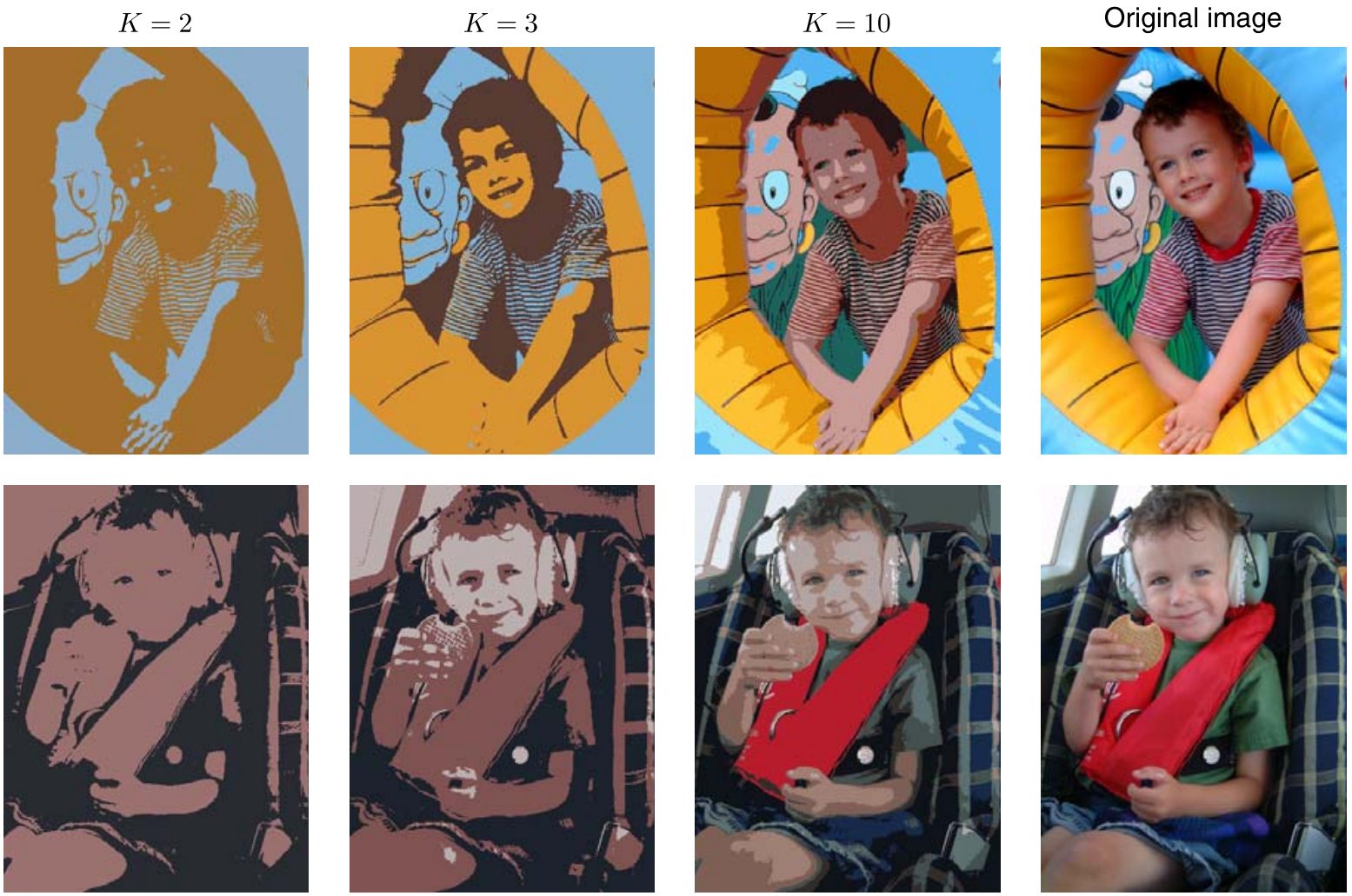


Figure 9.3 of Pattern Recognition and Machine Learning.

Gaussian Mixture

Let $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ be a collection of N input examples, each being a D-dimensional vector $\boldsymbol{x}_i \in \mathbb{R}^D$. Let K, the number of target clusters, be given.

Our goal is to represent the data as a Gaussian mixture, which is a combination of K Gaussian in the form

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

Therefore, each cluster is parametrized as $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$.

Let $\boldsymbol{z} \in \{0, 1\}^K$ be a K-dimensional random variable, such that exactly one z_k is 1, denoting to which cluster a training example belongs. Let the marginal distribution of z_k be

$$p(z_k = 1) = \pi_k.$$

Therefore, $p(\boldsymbol{z}) = \prod_k \pi_k^{z_k}$.

We can write

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{z}} p(\boldsymbol{z}) p(\boldsymbol{x}|\boldsymbol{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

and the probability of the whole clustering is therefore

$$\log p(\boldsymbol{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right).$$

To fit a Gaussian mixture model, we start with maximum likelihood estimation and minimize

$$\mathcal{L}(\boldsymbol{X}) = -\sum_{i} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

The derivative of the loss with respect to $\boldsymbol{\mu}_k$ gives

$$\frac{\partial \mathcal{L}(\boldsymbol{X})}{\partial \boldsymbol{\mu}_k} = -\sum_i \frac{\pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{l=1}^K \pi_l \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)} \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)$$

Denoting $r(z_{i,k}) = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{l=1}^K \pi_l \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}$, setting the derivative equal to zero and multiplying by $\boldsymbol{\Sigma}_k^{-1}$, we get

$$\boldsymbol{\mu}_k = \frac{\sum_i r(z_{i,k}) \boldsymbol{x}_i}{\sum_i r(z_{i,k})}.$$

The $r(z_{i,k})$ are usually called **responsibilities** and denote the probability $p(z_k = 1|\boldsymbol{x}_i)$. Note that the responsibilities depend on $\boldsymbol{\mu}_k$, so the above equation is not an analytical solution for $\boldsymbol{\mu}_k$, but can be used as an *iterative* algorithm for converting to local optimum.

For $\boldsymbol{\Sigma}_k$, we again compute the derivative of the loss, which is technically complicated (we need to compute derivative with respect a matrix, and also we need to differentiate matrix determinant) and results in an analogous equation

$$\boldsymbol{\Sigma}_k = \frac{\sum_i r(z_{i,k})(\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T}{\sum_i r(z_{i,k})}.$$

To minimize the loss with respect to $\boldsymbol{\pi}$, we need to include the constraint $\sum_k \pi_k = 1$, so we form a Lagrangian $\mathcal{L}(\boldsymbol{X}) + \lambda\left(\sum_k \pi_k - 1\right)$, and get

$$0 = \sum_i \frac{\mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{l=1}^K \pi_l \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)} + \lambda,$$

from which we get $\pi_k \propto \sum_i r(z_{i,k})$ and therefore

$$\pi_k = 1/N \cdot \sum_i r(z_{i,k}).$$

Gaussian Mixture

Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters (comprising the means and covariances of the components and the mixing coefficients).

1. Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k, and evaluate the initial value of the log likelihood.
2. **E step**. Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \tag{9.23}$$

3. **M step**. Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \tag{9.24}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}} \tag{9.25}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \tag{9.26}$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}). \tag{9.27}$$

4. Evaluate the log likelihood

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\} \tag{9.28}$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

Algorithm 9.2.2 of Pattern Recognition and Machine Learning.

Gaussian Mixture

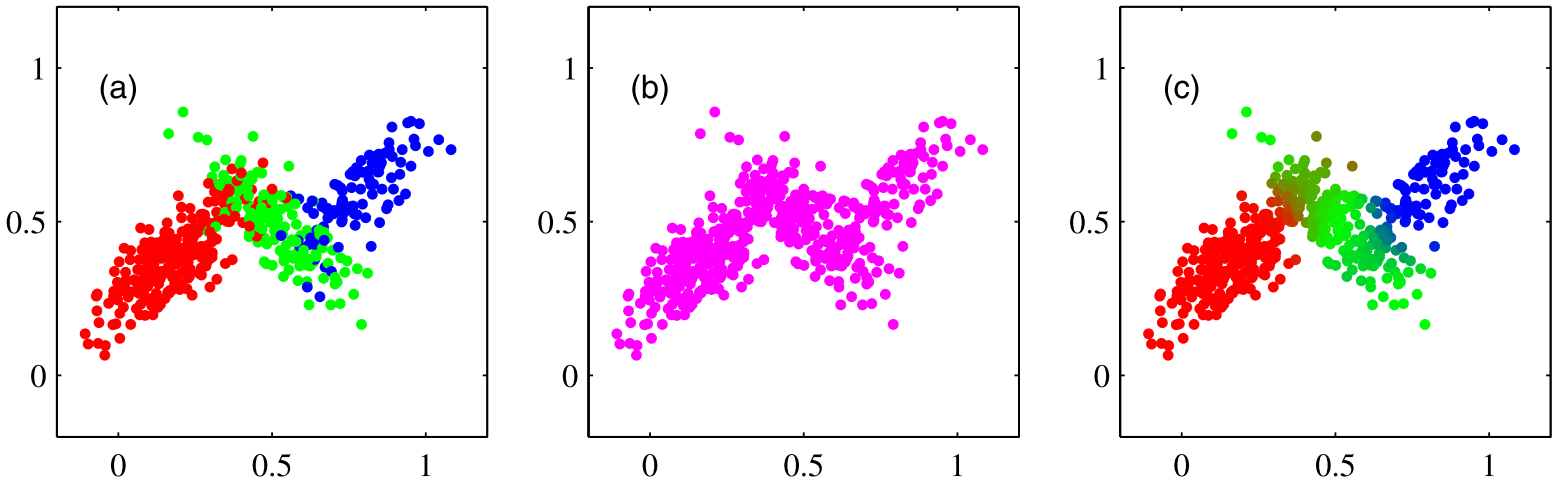


Figure 9.5 of Pattern Recognition and Machine Learning.

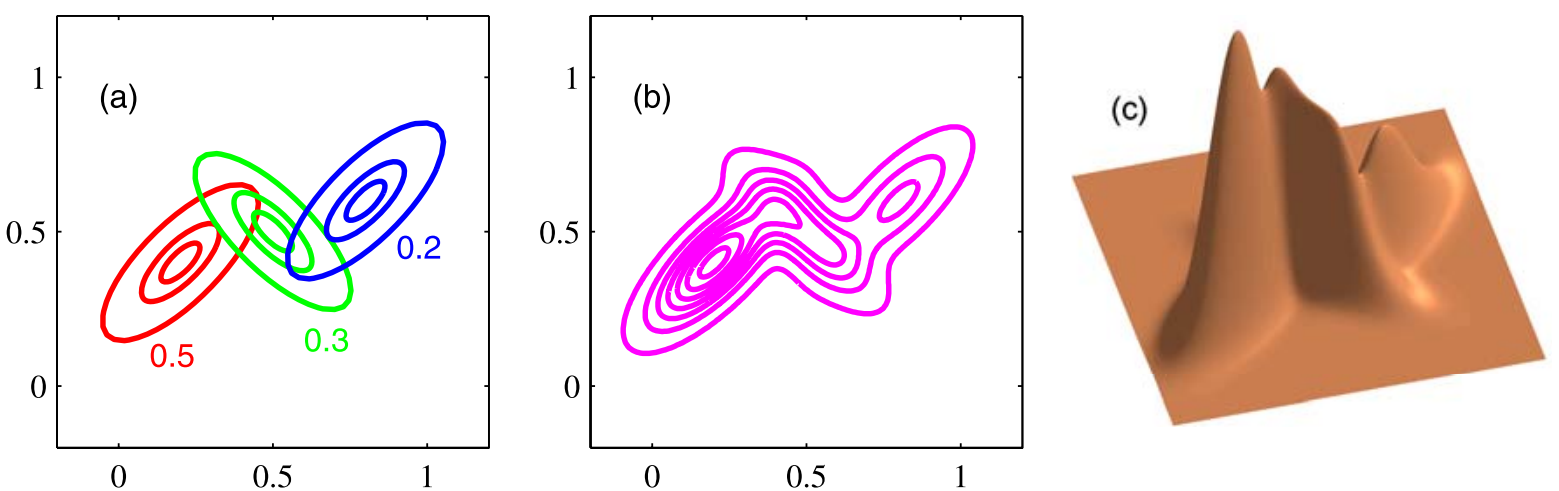


Figure 2.23 of Pattern Recognition and Machine Learning.

Gaussian Mixture

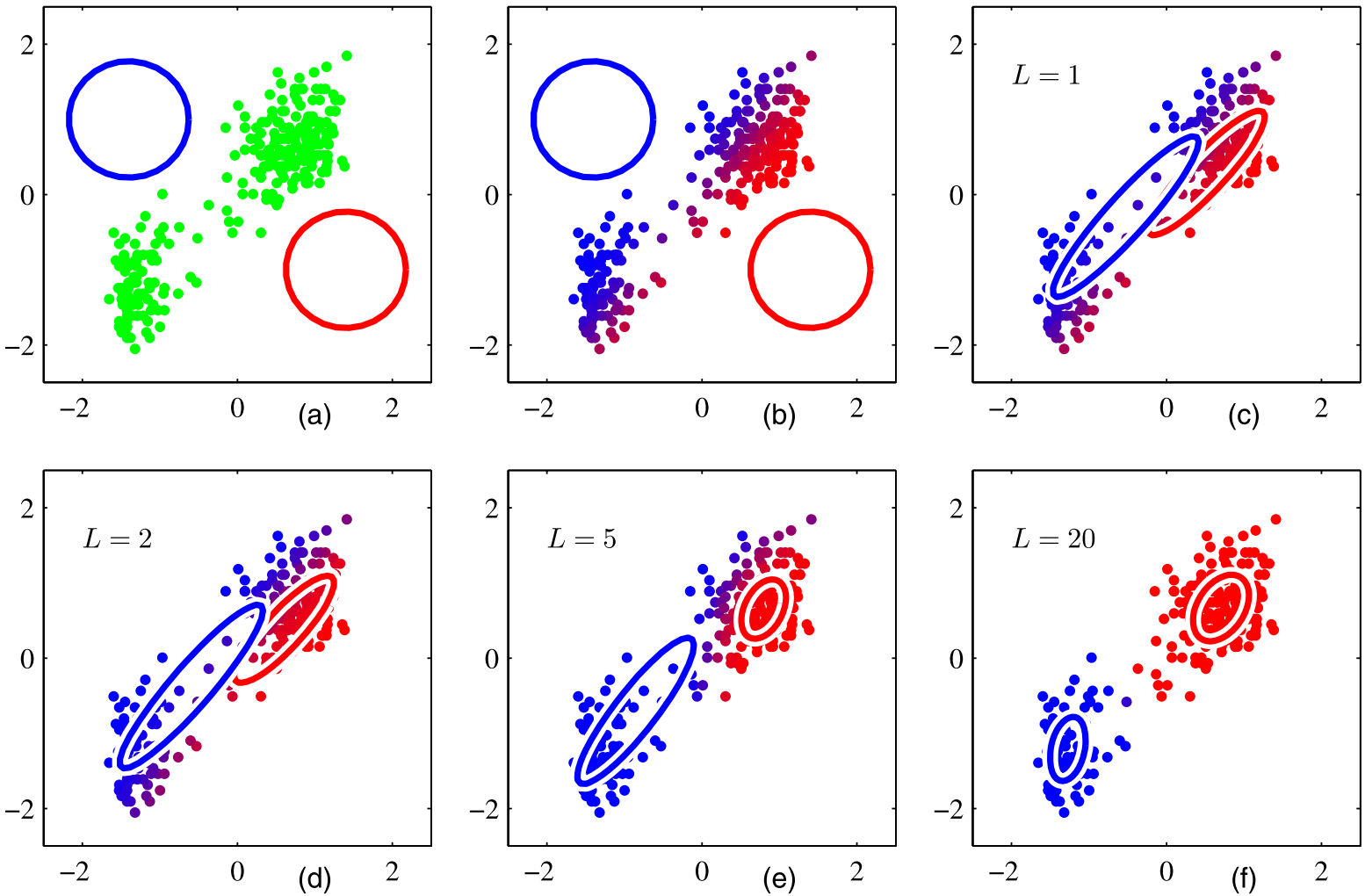


Figure 9.8 of Pattern Recognition and Machine Learning.