CHAPTER 5

TRIADIC SPLIT-MERGE SAMPLER

Contents This chapter introduces a new sampling method called the triadic split-merge sampler. The reason is that naive implementations of MCMC methods suffer from slow convergence in machine vision due to the complexity of the parameter space. Towards this blocked Gibbs and split-merge samplers have been developed that assign multiple data points to clusters at once. The triadic split-merge sampler improves on these samplers by defining split and merge steps between two and three clusters. This has two advantages. First, it reduces the asymmetry between the split and merge steps. Second, it is able to propose a new cluster that is composed out of data points from two different clusters. Both advantages speed up the convergence of the sampler on a line estimation problem.

Published in A.C. van Rossum, H.X. Lin, J. Dubbeldam, and H.J. van den Herik. Triadic Split-Merge Sampler. *The 10th International Conference on Machine Vision*, ICMV 2017, Vienna, Austria, November 13-November 15, 2017.

Outline We introduce the class of split-merge samplers as part of the MCMC samplers (Section 5.1). A conventional split-merge sampler, labeled the dyadic split-merge sampler, is detailed (Section 5.2). A new split-merge sampler, the triadic split-merge sampler is introduced (Section 5.3). The results for inference over lines are compared between the conventional dyadic sampler and the new triadic sampler (Section 5.4). Finally, we provide the chapter conclusions and describe how we can further improve the inference procedure (Section 5.5). They will be the basis of the next chapter.

5.1 The Class of Split-Merge Samplers

In clustering models there is a hierarchical structure. At the lowest level there are individual data points. At a higher level the points are grouped into clusters. Split-merge samplers are

samplers that take into account such structure as already described in Section 2.2.6. Rather than moving data points one by one from an old to a new cluster, split-merge samplers can perform moves that operate on partitions of the dataset. For example, a cluster can be split into two clusters in one single move or two clusters can be merged into once cluster in another single move.

In Section 5.1.1 we describe split-merge samplers in a bit more detail than in Section 2.2.6. In Section 5.1.2 the Dirichlet Process prior is introduced in the context of split-merge samplers.

5.1.1 Split and Merge Moves

One of the first split-merge samplers has been defined for so-called point sources in nuclear imaging (Stawinski et al., 1998). This split-merge sampler proposes split and merge moves that are defined locally. For instance, two clusters that are close to each other are a candidate for a merge step into one cluster. By defining pairs of split and merge steps with the right probabilities, a properly balanced Metropolis-Hastings step an be performed (Section 2.2.6).

In the hierarchical models of lines and segments we have used a Dirichlet process as prior. Split-merge samplers have been defined with a Dirichlet prior (Dahl, 2003; Jain and Neal, 2004). These split-merge samplers operate on one or two clusters and are defined in the thesis as dyadic split-merge samplers (Section 5.2). The split step is different per sampler: (1) the simple random split procedure does not take into account the data distribution, (2) the sequentially allocated merge-split sequentially assigns data towards one of the two clusters that is the better fit.

Other examples of split-merge samplers are a sampler that uses sub-cluster splits (Chang and Fisher III, 2013) a sampler that uses data-driven jumps besides split-merge steps (Hughes et al., 2012), a sampler that uses data-driven jumps, split-merge steps, and operates on a hierarchical Dirichlet process (Bryant and Sudderth, 2012), and a sampler that generalizes split-merge steps to birth-death steps (with multiple clusters generated simultaneously) (Hughes and Sudderth, 2013). We will introduce a sampler that generalizes the dyadic sampler to moves over two and three clusters (Section 5.3).

5.1.2 Dirichlet Process Prior

Let us first reiterate the Dirichlet process. We will consider a Dirichlet process as a prior on the distribution over parameters G. The form of this model is:

$$\begin{aligned} y_i|\theta_i &\sim F(\theta_i) \\ \theta_i|G &\sim G \\ G &\sim DP(H,\alpha) \end{aligned} \tag{5.1}$$

The split and merge steps dictate the simultaneous assignment of observations y_i unto parameters θ_i. A split assigns a set of observations to a new parameter value. A merge combines multiple sets of observations with different parameter values into one set of observations with a single parameter value.

5.2 Conventional Split-Merge Sampler

The conventional split-merge sampler (see Jain and Neal, 2004) splits a single cluster into two clusters, and merges two clusters into a single cluster. Hence, this split-merge sampler operates on two clusters at each time step. Therefore we will call their algorithm a dyadic split-merge sampler in contrast with our approach (Van Rossum et al., 2017). Below we describe this dyadic split-merge sampler in pseudo-code (see Algorithm 11).

Algorithm 11 Dyadic split-merge sampler

procedure DYADIC SPLIT-MERGE SAMPLER(c) ▷ Accepts cluster assignments c of length N (besides Metropolis-Hastings acceptance factors $a(c', c)$ and a split procedure e.g. SIMPLERANDOMSPLIT) and returns a (potentially) updated cluster assignment vector c'.
  $i \sim U(\{1, \ldots, N\})$ ▷ Sample i discrete uniformly over cluster assignments.
  $j \sim U(\{1, \ldots, N\} \setminus i)$ ▷ Sample j from the discrete uniform distribution excluding i.
  $S_R = \{c_i, c_j\}$ ▷ Sampled clusters c_i, c_j.
  $S_I = \{c_x\}$ with $c_x \in S_R$ for $x \in \{1, \ldots, N\}$ ▷ All data in clusters c_i, c_j.
  $S_E = S \setminus S_R$ ▷ All data in clusters c_i, c_j excluding S_R.
  $N_S = \text{unique}(S_R)$
  if $N_S = 1$ **then** ▷ Case: i, j belong to the same cluster.
    $c_i^{(2)} = c_k$ with $c_k \notin \{c_1, \ldots, c_N\}$ ▷ Sample new cluster for $c_i^{(2)}$.
    $c_j^{(2)} = c_j^{(1)}$ ▷ Keep c_j the same.
    $c_e^{(2)} = \text{SPLITPROCEDURE}(S_E, c_i^{(2)}, c_j^{(2)})$ ▷ After $c_i^{(2)}, c_j^{(2)}$ assign S_E.
    for all $m \notin S_I$ **do**
      $c_m^{(2)} = c_m^{(1)}$ ▷ Data points in clusters other than c_i, c_j are not adjusted.
    end for
    $c' = \{c_i^{(2)}, c_j^{(2)}, c_e^{(2)}, c_m^{(2)}\}$
    $a = a_{split}(c', c)$ according to Eq. 5.3 ▷ MH acceptance for a split.
  else ▷ Case: i, j belong to different clusters $c_i \neq c_j$ ($N_S = 2$).
    for all $q \in S_I$ **do**
      $c_q^{(1)} = c_j^{(2)}$ ▷ Assign all data points in c_i and c_j to c_j.
    end for
    for all $m \notin S_I$ **do**
      $c_m^{(1)} = c_m^{(2)}$ ▷ Data points in clusters other than c_i, c_j are not adjusted.
    end for
    $c' = \{c_q^{(1)}, c_m^{(1)}\}$
    $a = a_{merge}(c', c)$ according to Eq. 5.10 ▷ MH acceptance for a merge.
  end if
  $u \sim U(0, 1)$ ▷ Sample u between 0 or 1 uniformly.
  if $a < u$ **then**
    $c' = c$ ▷ Reject c' by setting it to c
  end if
  return c', the (updated) cluster assignment vector: $c \rightarrow c'$.
end procedure

In Algorithm 11 the notation $c_i^{(2)}$ is used to signify that the cluster assignment c_i has 2 clusters under consideration. In the dyadic algorithm we could have used c_i^{merge} and c_i^{split}, however in the triadic algorithm (see Algorithm 14) with multiple split and merge operations the latter notation would become confusing.

Algorithm 12 Simple random split

1: **procedure** SIMPLERANDOMSPLIT(S, c_0, c_1) ▷ Accepts unassigned set S and cluster indices c_0, c_1, returns cluster assignment c'_m.
2: **for all** $m \in S$ **do**
3: $c'_m \sim Cat(c_0, c_1)$ with equiprobable $p(c_0) = p(c_1) = \frac{1}{2}$.
4: **end for**
5: **return** c'_m, the cluster assignment for S.
6: **end procedure**

The dyadic split-merge sampler in Algorithm 11 samples two distinct data items. If the data items belong to the same cluster a split step is attempted. If the data items belong to different clusters a merge step is attempted. The split procedure itself is the so-called simple random split (Algorithm 12) that assigns data items with the same probability to one of the parts of the split cluster without any consideration for a proper data fit.

5.2.1 Acceptance for the Split Step

The acceptance ratio contains the Metropolis ratio to step from c to c':

$$\frac{P(c')L(c'|y)}{P(c)L(c|y)}. \tag{5.2}$$

Additionally, the Hastings correction is applied because of the asymmetry of the proposal distribution in the form of $q(c|c')/q(c'|c)$:

$$a_{split}(c^{(2)}, c^{(1)}) = \min\left[1, \frac{q(c^{(1)}|c^{(2)})}{q(c^{(2)}|c^{(1)})} \frac{P(c^{(2)})}{P(c^{(1)})} \frac{L(c^{(2)}|y)}{L(c^{(1)}|y)}\right]. \tag{5.3}$$

The notation $c^{(2)}$ is used to indicate that the cluster index vector is referencing 2 unique clusters (in this case after the split step).

The prior distribution is represented by a Chinese Restaurant Process with concentration parameter α and no discount factor. Data not yet assigned is assigned (1) with probability $\alpha/(n+\alpha)$ to a new cluster and (2) with probability $n_c/(n+\alpha)$ to an existing cluster c. Here n is the total number of assigned data points, n_c is the number of data points assigned to cluster c. There are D clusters. Hence, the prior over clusters will be:

$$P(c) = \frac{\Gamma(\alpha)}{\Gamma(\alpha+n)} \alpha^D \prod_{c_l} \Gamma(n_{c_l}) = \alpha^D \frac{\prod_{c_l}(n_{c_l}-1)!}{\prod_{k=1}^{n}(\alpha+k-1)}. \tag{5.4}$$

In the prior distribution ratio before and after the split step many of the factors drop out. There is one factor α remaining and the number of data points in the split cluster is part

of the equation. There is no dependency on other clusters or the total number of data points. We can simplify the formula using the Beta function $B(a,b) = \Gamma(a)\Gamma(b)/\Gamma(a+b)$ with $\Gamma(x) = (x-1)!$ the Euler-Gamma function:

$$\frac{P(c^{(2)})}{P(c^{(1)})} = \alpha \frac{(n_{c_i^{(2)}}-1)!(n_{c_j^{(2)}}-1)!}{(n_{c_i^{(1)}}-1)!} = \alpha B(n_{c_i^{(2)}}, n_{c_j^{(2)}}). \tag{5.5}$$

The likelihood can be written as a product over all observations y_i or as a product over clusters with each cluster a product over its observations y_k:

$$L(c|y) = \prod_{c=1}^{D} \prod_{k:c_k=c} p(y_k|\phi). \tag{5.6}$$

Here we write $p(y_k|\theta_k)$ rather than assuming conjugacy between the likelihood $F(\theta_k)$ and the prior distribution $H(\theta_k)$ (see Dahl, 2005). In the case of conjugacy we can analytically calculate $\int F(\theta_k)dH(\theta_k)$ which speeds up inference, but which restricts our choice of likelihoods and priors (see Chapter 3).

With the above formula for the likelihood, we can calculate the likelihood ratio of two clusters versus a single cluster:

$$\frac{L(c^{(2)}|y)}{L(c^{(1)}|y)} = \frac{\prod_{k:c_k^{(2)}=c_i^{(2)}} p(y_k|\phi) \prod_{k:c_k^{(2)}=c_j^{(2)}} p(y_k|\phi)}{\prod_{k:c_k^{(1)}=c_i^{(1)}} p(y_k|\phi)}. \tag{5.7}$$

The split step determines the probability of a particular split. Algorithm 11 commences with picking two random points. These two points are henceforth are already assigned to distinct clusters. Only the remaining points have to be assigned (see Figure 5.1).

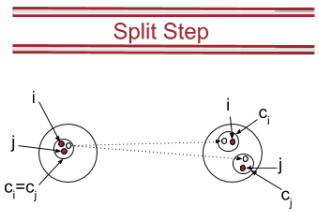


Figure 5.1: A split step. Points i and j are already assigned to separate clusters c_i and c_j. Each next point is assigned to one of the clusters with probability $\frac{1}{2}$, indicated by dotted arrows.

The remaining points are assigned with equal probability $\frac{1}{2}$ to $c_i^{(2)}$ and $c_j^{(2)}$:

$$q(c^{(2)}|c^{(1)}) = \left(\frac{1}{2}\right)^{-2+n_{c_i^{(2)}}+n_{c_j^{(2)}}} = \left(\frac{1}{2}\right)^{-2+n_{c_i^{(1)}}}. \tag{5.8}$$

The notation $n_{c_i^{(2)}}$ might seem complex, but it can be read as the number of points n in one of the clusters after the split. The split results in two clusters, indicated by the subscript (2). The cluster index is i. The probability of the reverse of the split operation is exactly 1. There is only one way in which a single cluster can be the starting state for a split operation. It must have had all points assigned to it. This means that the ratio with respect to the assignment of points over clusters in the split step becomes:

$$\frac{q(c^{(1)}|c^{(2)})}{q(c^{(2)}|c^{(1)})} = \frac{1}{\left(\frac{1}{2}\right)^{n_{c_i^{(2)}}+n_{c_j^{(2)}}-2}} = 2^{-2+n_{c_i^{(1)}}}. \tag{5.9}$$

Only basic identities are used and the fact that the number of data items does not change after a split, $n_{c_i^{(2)}} + n_{c_j^{(2)}} = n_{c_i^{(1)}}$. Note that the reverse split transition looks like a 'merging' operation. The merge step, however, is defined independently and its description can be found in the next section.

5.2.2 Acceptance for the Merge Step

Acceptance of a merge step consists of the same components as that of the split step.

$$a_{merge}(c^{(1)}, c^{(2)}) = \min\left[1, \frac{q(c^{(2)}|c^{(1)})}{q(c^{(1)}|c^{(2)})} \frac{P(c^{(1)})}{P(c^{(2)})} \frac{L(c^{(1)}|y)}{L(c^{(2)}|y)}\right]. \tag{5.10}$$

$$\frac{P(c^{(1)})}{P(c^{(2)})} = \alpha^{-1} \frac{(n_{c_i^{(1)}}-1)!}{(n_{c_i^{(2)}}-1)!(n_{c_i^{(2)}}-1)!} = \frac{1}{\alpha B(n_{c_i^{(2)}}, c_j^{(2)})}. \tag{5.11}$$

$$\frac{L(c^{(1)}|y)}{L(c^{(2)}|y)} = \frac{\prod_{k:c_k^{(1)}=c_i^{(1)}} p(y_k|\phi)}{\prod_{k:c_k^{(2)}=c_i^{(2)}} p(y_k|\phi) \prod_{k:c_k^{(2)}=c_j^{(2)}} p(y_k|\phi)}. \tag{5.12}$$

$$\frac{q(c^{(2)}|c^{(1)})}{q(c^{(1)}|c^{(2)})} = \left(\frac{1}{2}\right)^{-2+n_{c_i^{(1)}}} = 2^{2-n_{c_i^{(1)}}}. \tag{5.13}$$

The ratios of the merge step are the inverse of the ratios of the split step. That is, Eq. 5.11 is the inverse of Eq. 5.5, Eq. 5.12 is the inverse of Eq. 5.7, and Eq. 5.13 is the inverse of Eq. 5.9.

5.2.3 Sequentially-Allocated Merge-Split Sampler

A variant on the conventional split-merge sampler is the sequentially allocated merge-split (SAMS) sampler[1] (Dahl, 2003). The simple random split procedure of Algorithm 12 is replaced by a procedure that sequentially assigns observations to clusters rather than splitting the data random uniformly over the split clusters.

[1] In the naming of split-merge or merge-split samplers, the order of merge split does not bear any significance.

Algorithm 13 Sequentially Allocated Merge-Split

procedure SAMS(S, c_0, c_1) ▷ Accepts unassigned set S, cluster indices c_i, and $p(y_k|\theta_{c_i})$ with $i = 0, 1$, returns cluster assignment c'_m.
  $T =$ random_shuffle(S)
  for all $m \in T$ **do**
    $p(c_m = c_0|c_0, c_1, \theta_{c_0}, \theta_{c_1}) = \frac{N_0 p(y_k|\theta_0)}{N_0 p(y_k|\theta_0)+N_1 p(y_k|\theta_1)}$
    $p(c_m = c_1|c_0, c_1, \theta_{c_0}, \theta_{c_1}) = 1 - p(c_m = c_0|c_0, c_1, \theta_{c_0}, \theta_{c_1})$
    $c'_m \sim p(c_m|c_0, c_1, \theta_{c_0}, \theta_{c_1})$
  end for
  return c'_m, the cluster assignment for S.
end procedure

In contrast to the simple random split, observations y_k are used in the SAMS to obtain cluster assignments that correspond with the data rather than cluster assignments independent of the data.

5.3 Triadic Split-Merge Sampler

The triadic split-merge sampler that we introduce (Van Rossum et al., 2017) uses up to three clusters for a split or merge step (Fig. 5.2).

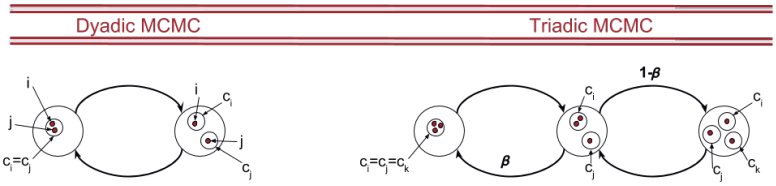


Figure 5.2: Right: dyadic MCMC picks two data items i, j random uniformly. If both are in the same cluster a split towards two clusters is attempted. If both are in distinct clusters a merge towards one cluster is attempted. Left: triadic MCMC picks three data items i, j, k random uniformly. If all three are in the same cluster a split towards two clusters is attempted. If the three items are in two clusters either a split into three (with probability $1-\beta$) or a merge into a single cluster (with probability β) is attempted. If the three data items are in three distinct clusters a merge is attempted. There are no direct transitions from a single cluster to three clusters or the other way around.

The intuition behind the triadic split-merge sampler is twofold:

- In the dyadic sampler there is a large asymmetry between split and merge steps. There is only one way in which two clusters can be merged into one single cluster, while there are many ways in which one single cluster can be split into two clusters. This asymmetry is reduced by transitioning between two and three clusters. This is a straightforward improvement in balancing split and merge steps (for alternatives, see Wang and Russell (2015)).

- In practical optimization problems it might be useful to form a third cluster out of subsets of two other clusters. The dyadic MCMC sampler requires intermediate steps in which (1) one of these clusters is split into two, (2) the other is split into two, and (3) the two new clusters are merged. This means that (a) mixing and hence convergence will be slow and (b) the intermediate steps might have very low probability and function as an unnecessary barrier between high probable states.

The algorithm is detailed in Algorithm 14. Compare with Algorithm 11. It starts with sampling three distinct points i, j, and k. These points can originate from one, two, or three distinct clusters with non-unique indices c_i, c_j, and c_k. Depending on the number of distinct clusters N_S, either a dyadic or triadic step is performed. If all points belong to the same cluster, $N_S = 1$, a split step into two clusters is attempted. If the points belong to two clusters $N_S = 2$, with probability β a merge step into one cluster is tried, with probability $1-\beta$ a triadic split into three clusters will be considered. If the points belong to three distinct clusters, $N_S = 3$, a triadic merge step into two clusters will be attempted. All steps will be accepted or rejected according to the Metropolis-Hastings probability ratios in the following section. Note that there are no steps with which a single cluster is immediately split into three, nor is there is a step that merges three clusters into one.

Sampling random uniformly for three unique items is implemented through a random shuffle algorithm, in particular the modern version of the Fisher-Yates shuffle introduced by Durstenfeld (1964) and picking the first three items.

The Metropolis-Hastings probabilities for the triadic split and merge steps are calculated in the following Sections 5.3.1 and 5.3.2.

5.3.1 Acceptance for the Split Step

In the triadic split-merge sampler there are two splitting steps. It is possible to split according to the dyadic split-merge sampler. However, given two clusters there are (split) jumps to three states as well as (merge) jumps to single states again. To account for this asymmetry another Hastings correction is applied to establish detailed balance.

$$a_{split}(c^{(2)}, c^{(1)}) = \min\left[1, \frac{r(c^{(1)}|c^{(2)})}{r(c^{(2)}|c^{(1)})} \frac{q(c^{(1)}|c^{(2)})}{q(c^{(2)}|c^{(1)})} \frac{P(c^{(2)})}{P(c^{(1)})} \frac{L(c^{(2)}|y)}{L(c^{(1)}|y)}\right]. \tag{5.14}$$

Here we have one additional term compared to the split step from one cluster to two clusters:

$$\frac{r(c^{(1)}|c^{(2)})}{r(c^{(2)}|c^{(1)})} = \frac{\beta}{1}. \tag{5.15}$$

The parameter β is free to control, as long as $0 < \beta < 1$ (to maintain ergodicity). The transition from two states to three states is another split step:

$$a_{split}(c^{(3)}, c^{(2)}) = \min\left[1, \frac{r(c^{(2)}|c^{(3)})}{r(c^{(3)}|c^{(2)})} \frac{q(c^{(2)}|c^{(3)})}{q(c^{(3)}|c^{(2)})} \frac{P(c^{(3)})}{P(c^{(2)})} \frac{L(c^{(3)}|y)}{L(c^{(2)}|y)}\right]. \tag{5.16}$$

Algorithm 14 Triadic split-merge sampler

procedure TRIADIC SPLIT-MERGE SAMPLER(c) ▷ Accepts cluster assignments c of length N (besides Metropolis-Hastings acceptance factors $a(c', c)$ and a split procedure) and returns a (potentially) updated cluster assignment vector c'.
  $i \sim U(\{1, \ldots, N\})$ ▷ Sample i discrete uniformly over cluster assignments.
  $j \sim U(\{1, \ldots, N\} \setminus i)$ ▷ Sample j from the discrete uniform distribution excluding i.
  $k \sim U(\{1, \ldots, N\} \setminus \{i, j\})$ ▷ Sample k from the discrete uniform distribution excluding $\{i, j\}$.
  $S_R = \{c_i, c_j, c_k\}$ ▷ Sampled clusters c_i, c_j, c_k.
  $S_I = \{c_x\}$ with $c_x \in S_R$ for $x \in \{1, \ldots, N\}$ ▷ All data in clusters c_i, c_j, c_k.
  $S_E = S_I \setminus S_R$ ▷ All data in clusters c_i, c_j, c_k excluding S_R.
  $N_S = \text{unique}(S_R)$
  $u \sim U(0, 1)$ ▷ Sample u between 0 or 1 uniformly.
  if $N_S = 1$ **then** ▷ Case: i, j, k belong to the same cluster.
    return $c' = $ DYADIC SPLIT-MERGE SAMPER(c)
  else if $N_S = 2$ **and** $u < \beta$ **then** ▷ Case: a cluster with one item and one with two items and $u < \beta$.
    return $c' = $ DYADIC SPLIT-MERGE SAMPER(c)
  else if $N_S = 2$ **and** $u \geq \beta$ **then** ▷ Case: a cluster with one item and one with two items and $u \geq \beta$.
    $c_i^{(3)} = c_k$ with $c_k \notin \{c_1, \ldots, c_N\}$ ▷ Sample new cluster for $c_i^{(3)}$.
    $c_j^{(3)} = c_j^{(2)}$ ▷ Keep c_j the same.
    $c_e^{(3)} = $ SPLITPROCEDURE$(S_E, c_i^{(3)}, c_j^{(3)})$ ▷ After $c_i^{(3)}, c_j^{(3)}$ assign S_E.
    for all $m \notin S_I$ **do**
      $c_m^{(3)} = c_m^{(2)}$ ▷ Data points in clusters other than c_i, c_j are not adjusted.
    end for
    $c' = \{c_i^{(3)}, c_j^{(3)}, c_e^{(3)}, c_m^{(3)}\}$
    $a = a_{split}(c', c)$ according to Eq. 5.14 ▷ MH acceptance for a split.
  else ▷ Case: i, j, k belong to thee different clusters $c_i \neq c_j \neq c_k$ ($N_S = 3$).
    $S_L = S_I \setminus \{c_i^{(3)}, c_j^{(3)}\}$ ▷ Data in clusters c_i, c_j, c_k except for i and j itself.
    $\{c_i^{(2)}, c_j^{(2)}\} = \text{SAMS}(S_L, c_i^{(3)}, c_j^{(3)})$ ▷ Assign data points in c_i, c_j, c_k to c_i, c_j.
    for all $m \notin S_L$ **do**
      $c_m^{(2)} = c_m^{(3)}$ ▷ Data points in clusters other than S_L are not adjusted.
    end for
    $c' = \{c_i^{(2)}, c_j^{(2)}, c_m^{(2)}\}$
    $a = a_{merge}(c', c)$ according to Eq. 5.21 ▷ MH acceptance for a merge.
  end if
  $u \sim U(0, 1)$ ▷ Sample u between 0 or 1 uniformly.
  if $a < u$ **then**
    $c' = c$ ▷ Reject c' by setting it to c
  end if
  return c', the (updated) cluster assignment vector: $c \to c'$.
end procedure

The fraction with r reads as follows:

$$\frac{r(c^{(2)}|c^{(3)})}{r(c^{(3)}|c^{(2)})} = \frac{1}{1-\beta}. \tag{5.17}$$

The fraction with q uses the total number of data points n_c in the clusters:

$$\frac{q(c^{(2)}|c^{(3)})}{q(c^{(3)}|c^{(2)})} = \frac{\left(\frac{1}{2}\right)^{n_c-2}}{\left(\frac{1}{3}\right)^{n_c-3}} = \left(3^{n_c-3}\right)\left(2^{2-n_c}\right) = \left(\frac{3}{2}\right)^{n_c} \frac{2^2}{3^3}. \tag{5.18}$$

To move from 2 clusters to 3 clusters the probability is a $1/3$ for each cluster index in vector c (except for the three data items already selected randomly, hence $n_c - 3$). To move back, the probability is a $1/2$ and there are only two data items randomly assigned beforehand. The fraction with P uses the number of data points in each of the clusters before and after the step:

$$\frac{P(c^{(3)})}{P(c^{(2)})} = \alpha \frac{(n_{c_i^{(3)}} - 1)!(n_{c_j^{(3)}} - 1)!(n_{c_k^{(3)}} - 1)!}{(n_{c_i^{(2)}} - 1)!(n_{c_j^{(2)}} - 1)!} = \alpha \frac{B(n_{c_i^3}, n_{c_j^3}, n_{c_k^3})}{B(n_{c_i^2}, n_{c_j^2})}. \tag{5.19}$$

Here we introduced a generalized Beta function $B(a,b,c) = \Gamma(a)\Gamma(b)\Gamma(c)/\Gamma(a+b+c)$ with $\Gamma(x) = (x-1)!$ the Gamma function. The likelihood ratio becomes:

$$\frac{L(c^{(3)}|y)}{L(c^{(2)}|y)} = \frac{\prod_{m:c_m^{(3)}=c_i^{(3)}} p(y_m|\phi) \prod_{m:c_m^{(3)}=c_j^{(3)}} p(y_m|\phi) \prod_{m:c_m^{(3)}=c_k^{(3)}} p(y_m|\phi)}{\prod_{m:c_m^{(2)}=c_i^{(2)}} p(y_m|\phi) \prod_{m:c_m^{(2)}=c_j^{(2)}} p(y_m|\phi)}. \tag{5.20}$$

5.3.2 Acceptance for the Merge Step

The merge step from two to one cluster is analogous to the split step:

$$a_{merge}(c^{(1)}, c^{(2)}) = \min\left[1, \frac{r(c^{(2)}|c^{(1)})}{r(c^{(1)}|c^{(2)})} \frac{q(c^{(2)}|c^{(1)})}{q(c^{(1)}|c^{(2)})} \frac{P(c^{(1)})}{P(c^{(2)})} \frac{L(c^{(1)}|y)}{L(c^{(2)}|y)}\right]. \tag{5.21}$$

The merge step from three clusters to two clusters is:

$$a_{merge}(c^{(2)}, c^{(3)}) = \min\left[1, \frac{r(c^{(3)}|c^{(2)})}{r(c^{(2)}|c^{(3)})} \frac{q(c^{(3)}|c^{(2)})}{q(c^{(2)}|c^{(3)})} \frac{P(c^{(2)})}{P(c^{(3)})} \frac{L(c^{(2)}|y)}{L(c^{(3)}|y)}\right]. \tag{5.22}$$

Note that all the fractions in Eq. 5.22 are the inverse of the fractions in Eq. 5.16. Inverting Eq. 5.17–5.20 will be left to the reader.

One additional issue we have to consider. When merging three clusters into two we can (1) distribute the data over all three clusters or (2) alternatively, keep the data in two clusters assigned to these clusters and only distribute the data in the third cluster over the other two clusters. The second and alternative option however would introduce unnecessary asymmetry with the merge step. In other words, Eq. 5.23 is not the inverse of Eq. 5.18. In contrast, the equation is similar to splitting one cluster across two as in Eq. 5.9:

$$\frac{q_{alt}(c^{(3)}|c^{(2)})}{q_{alt}(c^{(2)}|c^{(3)})} = 2^{-2+n_c}. \tag{5.23}$$

Hence the first option is entertained and the q-fraction is exactly the inverse of Eq. 5.18.

A second issue has to be considered, namely the inclusion or exclusion of direct operations between a single cluster and three clusters. This is because factors such as

$$\frac{P(c^{(3)})}{P(c^{(1)})} = \alpha^2 \frac{(n_{c_i^{(3)}} - 1)!(n_{c_j^{(3)}} - 1)!(n_{c_k^{(3)}} - 1)!}{(n_{c_i^{(1)}} - 1)!}, \tag{5.24}$$

become very small and although compensated by a large q fraction, remain further away from an acceptance factor of 1. Note that by the ability to split a single cluster into two and then into three, there is no ergodic argument to introduce also the immediate step.

5.4 Results

The problem we use to test our sampler is a well-known problem in computer vision, namely that of the inference of line parameters (slope and intercept) given data points. Rather than ordinary linear regression, in computer vision there is a mixture of lines that have to be estimated. Moreover, the number of lines is not known in advance. To solve this problem we use the Dirichlet process mixture (Eq. 5.1) with a normal distribution $N(0, \sigma_0^2)$ to generate the line parameters and a likelihood function that defines points to be uniformly distributed across a line of length 20 and deviating from the line according to a normal distribution $N(0, \sigma_1^2)$.

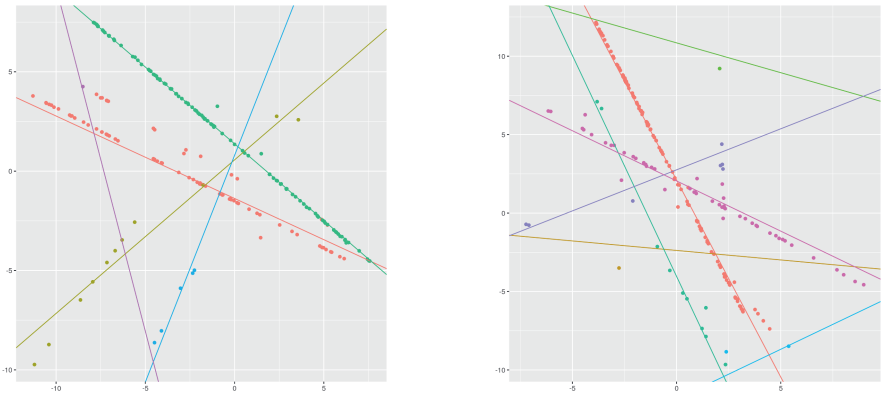


Figure 5.3: Two examples of fitting a mixture of lines to data items scattered over a two-dimensional space. The lines drawn are inferred using the triadic sampler. The lines are not the ground truth, but are meant to demonstrate the typical errors made by fitting methods. Note, for example, that there are mistakes in both the assignment of points to lines as well as the line parameters (slope and intercept). Left: In example 1 two lines with similar slope are seen as the same line. Right: In example 2 points on one vertical line are assigned to multiple lines. In example 1 and 2 slopes are not always through the points.

In Section 5.4.1 the triadic sampler's implementation is discussed. In Section 5.4.2 the triadic sampler is compared with the conventional Jain-Neal (dyadic) sampler and the auxiliary variable sampler of the previous chapter.

5.4.1 Implementation

The sampler is open-source[2] and is implemented in C++ which means that (a) it is computationally fast, (b) it can be run on embedded devices (if a cross-compiler is available and the Eigen3 library is ported). Note, that due to the fact that the simulator uses a lot of random numbers the system should use a modern compiler (g++-6 or newer) and should have enough entropy available[3]. Rather than a random scan, the implementation uses a fixed scan as advocated in the literature (MacEachern, 2007).

To speed up the sampler most calculations are done in log-space. Consider $v = u + 1$. The ratio with probabilities (Eq. 5.5 and 5.19) becomes:

$$\log \frac{P(c^{(v)})}{P(c^{(u)})} = \log(\alpha) + \sum_i \log \Gamma(n_{c_i^{(v)}}) - \sum_i \log \Gamma(n_{c_i^{(u)}}). \tag{5.25}$$

The fraction with $q(\cdot)$ (Eq. 5.9 and 5.18) becomes:

$$\log \frac{q(c^{(v-1)}|c^{(v)})}{q(c^{(v)}|c^{(v-1)})} = (v - n_c - 1)\log(v-1) - (v - n_c)\log(v). \tag{5.26}$$

The fraction with r becomes, for example, (Eq. 5.17):

$$\log \frac{r(c^{(2)}|c^{(3)})}{r(c^{(3)}|c^{(2)})} = -\log(1-\beta). \tag{5.27}$$

The log-probability to calculate the likelihood given by a multivariate Normal distribution is well-known.

5.4.2 Comparison

The Triadic sampler using SAMS is compared with the Jain-Neal Dyadic sampler using SAMS and an auxiliary variable sampler with $m = 3$ (see algorithm 8 in Neal (2000)).

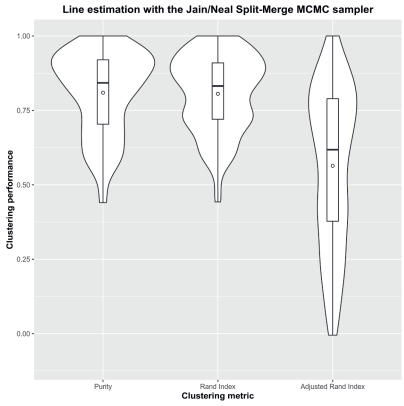
In Table 5.1 the line estimation problem is compared for the dyadic sampler, an auxiliary variables sampler, and the proposed triadic sampler. The simulation is run with $\beta = 0.1$ so that a significant number of steps are tried between two and three clusters (rather than only between one and two clusters).

In Figures 5.4a to 5.4c the different metrics are visualized in the form of violin plots.

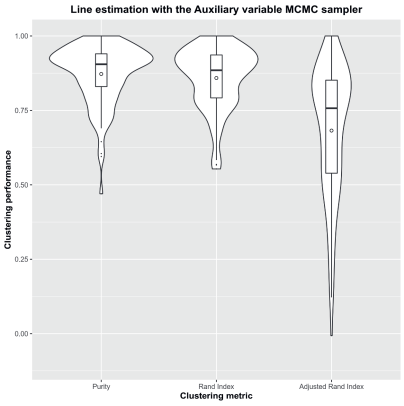
[2] Code can be found at `https://code.annevanrossum.nl/noparama`.
[3] On Linux this can be checked in /proc/sys/kernel/random/entropy_avail.

Table 5.1: The purity, rand index, and adjusted rand index establishing the quality of the clustering method. The closer the values to one, the better the method performed. The purity metric assigns high values to clusters that do not have data points from other clusters (but does not penalize the number of clusters). The rand index index computes similarity between clusters taking false negatives and false positives into account. The adjusted rand index accounts for chance. The adjusted rand index is most useful in our comparison.

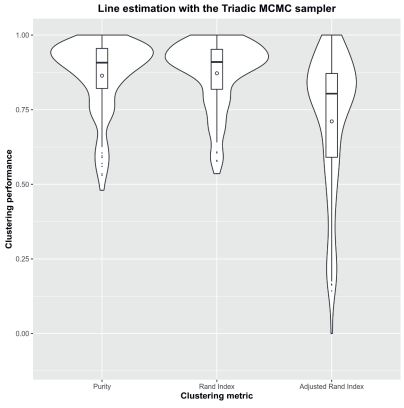
Method	Purity	Rand Index	Adjusted Rand Index
Dyadic sampler	0.80960	0.80580	0.56382
Auxiliary variables	0.87235	0.85879	0.68224
Triadic sampler	0.86405	0.87188	0.71067



(a) Line estimation with the dyadic split-merge sampler.



(b) Line estimation with the auxiliary variable sampler.



(c) Line estimation with the triadic split-merge sampler (our inference method). The values are all shifted up towards one.

Figure 5.4: The different inference methods for line estimation compared. The same results as in Table 5.1, but visualized in a violin plot. The distribution over metric values are displayed in a vertical fashion.

The improvement in clustering is especially visible with the adjusted rand index.

5.5 Chapter Conclusions

A new split-merge sampler has been introduced, implemented, and applied to the computer vision problem of line estimation. The sampler outperforms existing samplers, such as the ordinary (dyadic) split-merge sampler (Jain and Neal, 2004) and auxiliary variable sampler (Neal, 2000).

The triadic split-merge sampler has been used with likelihood functions that correspond to line fitting. It therefore estimates the number of lines and simultanously performs line fitting. Moreover, the sampler is optimized to reassign points from three lines to two lines and the other way around. This means a hypothesized third line can be composed at once from two existing lines. These triadic steps accelerate the inference process as shown in Section 5.4.

This chapter answers our second research question.

> **RQ 2** How can we optimize inference over both the number of objects and fitting of those objects in the robotic vision domain?

The triadic sampler optimizes the inference process by using spatial properties inherent to the robotic vision domain. To gain some intuition for this consider the following simplified line recognition problem. There are three lines: two vertical and one horizontal line. The horizontal line intersects both vertical lines. At some step in the inference process the vertical lines have been assigned points. The horizontal line has not been recognized yet. Now it would make sense to suggest a step where points currently assigned to both vertical lines will be combined to propose a horizontal line. The triadic sampler uses such steps. This benefits robotic vision problems where this spatial property of "object intersection" happens regularly.

Although the proposed split-merge sampler is able to mix considerably faster through a mixture model, it does not use global jumps directly based on the data. It is reasonable to suggest that MCMC methods benefit from combining the local jumps with global jumps, for example by a mixture of the local Metropolis-Hastings sampler with a Metropolized independence sampler (Jampani et al., 2015).

However, if the data is used to introduce adaptations to the sampler in this way, there are other data-driven methods that might improve performance considerably (Barbu and Zhu, 2005; Hughes et al., 2012). One of the data-driven methods that has caught considerable attention in the literature is a part of machine learning now known as deep learning methods (LeCun et al., 2015; Schmidhuber, 2015). We will introduce such methods in chapter Chapter 6.