A. Alias Sampler

A key component is the alias sampler of (Walker, 1977). Given an arbitrary discrete probability distribution on n outcomes, it allows for $O(1)$ sampling once an $O(n)$ preprocessing step has been performed. Hence, drawing n observations a distribution over n outcomes costs an amortized $O(1)$ per sample. Given probabilities π_i with $\pi \in \mathcal{P}_n$ the algorithm proceeds as follows:

- Decompose $\{1, \dots n\}$ into sets L, H with $i \in L$ if $\pi_i < n^{-1}$ and $i \in H$ otherwise.
- For each $i \in L$ pick some $j \in H$.
  - Append the triple (i, j, π_i) to an array A
  - Set residual $\pi'_j := \pi_j + \pi_i - n^{-1}$
  - If $\pi'_j > n^{-1}$ return π'_j to H, otherwise to L.

Preprocessing takes $O(n)$ computation and memory since we remove one element at a time from L.

- To sample from the array pick $u \sim U(0, 1)$ uniformly at random.
- Choose the tuple (i, j, π_i) at position $\lfloor un \rfloor$.
- If $u - n^{-1}\lfloor un \rfloor < \pi_i$ return i, else return j.

This step costs $O(1)$ operations and it follows by construction that i is returned with probability π_i. Now we need a data structure that will allow us to sample many objects *in bulk* without the need to inspect each item individually. Cover trees satisfy this requirement.

B. Rejection Sampling

B.1. Flat Clusters

The proof for the proposed rejection sampler in case of sampling a cluster for a single observation x is as follows. If we approximate $p(x|\theta_z)$ by some q_z such that

$$e^{-\epsilon} p(x|\theta_z) \leq q_z \leq e^{\epsilon} p(x|\theta_z) \tag{13}$$

then it follows that a sampler drawing z from

$$z \sim \frac{q_z p(z)}{\sum_{z'} q_{z'} p(z')} \tag{14}$$

and then accepting with probability $e^{-\epsilon} q_z^{-1} p(x|\theta_z)$ will draw from $p(z|x)$. To prove this, we begin by computing the probability of this sampler $r(z)$ to return a particular value z. The sampler returns z when it (a) samples and accepts z, or (b) samples any value, rejects it to proceed to next iteration of sampling. Using $\gamma = \sum_{z'} q_{z'} p(z')$ and $\gamma_T = \sum_{z'} p(x|\theta_{z'}) p(z')$ to denote normalization for proposal and true posterior respectively, we have:

$$\begin{aligned} r(z) &= \frac{q_z p(z)}{\gamma} e^{-\epsilon} q_z^{-1} p(x|\theta_z) + \sum_{z'} (1 - e^{-\epsilon} q_{z'}^{-1} p(x|\theta_{z'})) \frac{q_{z'} p(z')}{\gamma} r(z) \\ &= \frac{e^{-\epsilon}}{\gamma} p(z) p(x|\theta_z) + \frac{r(z)}{\gamma} \sum_{z'} q_{z'} p(z') - r(z) \frac{e^{-\epsilon}}{\gamma} \sum_{z'} p(x|\theta_{z'}) p(z') \\ &= \frac{e^{-\epsilon}}{\gamma} p(z) p(x|\theta_z) + r(z) - r(z) \frac{e^{-\epsilon}}{\gamma} \gamma_T \\ r(z) &= \frac{p(z) p(x|\theta_z)}{\gamma_T} \end{aligned} \tag{15}$$

Hence the procedure will draw from the true posterior $p(z|x)$.

B.2. Clusters Arranged in Cover Tree

We now extend the above proof strategy when the clusters are arranged in a cover tree, thereby proving the correctness of our rejection sampler in Sec. 3.2.2.

Similar to previous case, we approximate $p(x|\theta_z)$ for z in level i by some q_z such that

$$e^{-\epsilon_i} p(x|\theta_z) \leq q_z \leq e^{\epsilon_i} p(x|\theta_z). \tag{16}$$

Note that approximation error ϵ_i now depends on the location of the cluster in the cover tree. To be specific, if the cluster z is located at level i, then $\epsilon_i = 2^i \|\bar{\phi}(x)\|$. Also, we assume the path to reach the node z starting from its (grand) parent at level $\hat{\imath}$ is given by $\mathcal{T} = [\mathcal{T}(\hat{\imath}), \mathcal{T}(\hat{\imath}-1), ..., \mathcal{T}(i)]$, with $\mathcal{T}(i) = z$.

To prove the correctness of our rejection sampler in Sec. 3.2.1, we simply show that probability of this sampler to return a particular value z is equal to the true posterior. The sampler returns z when it (a) reaches the corresponding node in the cover tree and accepts it, or (b) rejects or exits to proceed to next iteration of sampling. So, the probability of this sampler to return z is given by:

$$r(z) = \underbrace{\mathcal{A}(z)}_{\substack{\text{Probability of the} \\ \text{sampler accepting } z}} + r(z) \underbrace{\mathcal{E}}_{\substack{\text{Probability of the} \\ \text{sampler rejecting or exiting}}} \tag{17}$$

We calculate these individual terms beginning with the probability of sampler accepting z. Using γ as defined in (10) and $\gamma_T = \sum_{z'} p(x|\theta_{z'})p(z')$, we have:

$$\begin{aligned}
\mathcal{A}(z) &= \underbrace{\frac{e^{\epsilon_{\hat{\imath}}}\beta(\hat{\imath}, \mathcal{T}(\hat{\imath}))p(x|\theta_{\mathcal{T}(\hat{\imath})})}{\gamma}}_{\text{Selecting the first parent (Step 4)}} \left[\underbrace{\prod_{i'=i+1}^{\hat{\imath}}}_{\substack{\text{The loop} \\ \text{(Step 5)}}} \underbrace{\left(1 - \frac{p(\mathcal{T}(i'))p(x|\theta_{\mathcal{T}(i')})}{e^{\epsilon_{i'}}\beta(i', \mathcal{T}(i'))p(x|\theta_{\mathcal{T}(i')})}\right)}_{\text{Rejecting the nodes (Step 5iii)}} \underbrace{\frac{e^{\epsilon_{i'-1}}\beta(i'-1, \mathcal{T}(i'-1))p(x|\theta_{\mathcal{T}(i'-1)})}{e^{\epsilon_{i'}}\beta(i', \mathcal{T}(i'))p(x|\theta_{\mathcal{T}(i')}) - p(\mathcal{T}(i'))p(x|\theta_{\mathcal{T}(i')})}}_{\text{Selecting the next node (Step 5ii)}} \right] \\
&\qquad \times \underbrace{\frac{p(z)p(x|\theta_z)}{e^{\epsilon_i}\beta(i, \mathcal{T}(i))p(x|\theta_{\mathcal{T}(i)})}}_{\text{Accepting node } z} \\
&= \frac{e^{\epsilon_{\hat{\imath}}}\beta(\hat{\imath}, \mathcal{T}(\hat{\imath}))p(x|\theta_{\mathcal{T}(\hat{\imath})})}{\gamma} \left[\prod_{i'=i+1}^{\hat{\imath}} \frac{e^{\epsilon_{i'-1}}\beta(i'-1, \mathcal{T}(i'-1))p(x|\theta_{\mathcal{T}(i'-1)})}{e^{\epsilon_{i'}}\beta(i', \mathcal{T}(i'))p(x|\theta_{\mathcal{T}(i')})} \right] \frac{p(z)p(x|\theta_z)}{e^{\epsilon_i}\beta(i, \mathcal{T}(i))p(x|\theta_{\mathcal{T}(i)})} \\
&= \frac{e^{\epsilon_{\hat{\imath}}}\beta(\hat{\imath}, \mathcal{T}(\hat{\imath}))p(x|\theta_{\mathcal{T}(\hat{\imath})})}{\gamma} \left[\frac{e^{\epsilon_i}\beta(i, \mathcal{T}(i))p(x|\theta_{\mathcal{T}(i)})}{e^{\epsilon_{\hat{\imath}}}\beta(\hat{\imath}, \mathcal{T}(\hat{\imath}))p(x|\theta_{\mathcal{T}(\hat{\imath})})} \right] \frac{p(z)p(x|\theta_z)}{e^{\epsilon_i}\beta(i, \mathcal{T}(i))p(x|\theta_{\mathcal{T}(i)})} \\
&= \frac{p(z)p(x|\theta_z)}{\gamma} \qquad \text{(by telescoping)}
\end{aligned} \tag{18}$$

Next, the probability of rejecting or exiting from the sampler is one minus probability of accepting any node z, *i.e.*

$$\begin{aligned}
\mathcal{E} &= 1 - \sum_{z' \in \mathcal{Z}} \mathcal{A}(z') \\
&= 1 - \sum_{z' \in \mathcal{Z}} \frac{p(z')p(x|\theta_{z'})}{\gamma} \\
&= 1 - \frac{\gamma_T}{\gamma}
\end{aligned} \tag{19}$$

Plugging back the acceptance and exit probabilities into (17):

$$\begin{aligned}
r(z) &= \mathcal{A}(z) + r(z)\mathcal{E} \\
&= \frac{p(z)p(x|\theta_z)}{\gamma} + r(z)\left(1 - \frac{\gamma_T}{\gamma}\right) \\
&= \frac{p(z)p(x|\theta_z)}{\gamma} + r(z) - r(z)\frac{\gamma_T}{\gamma} \\
r(z) &= \frac{p(z)p(x|\theta_z)}{\gamma_T}
\end{aligned} \tag{20}$$

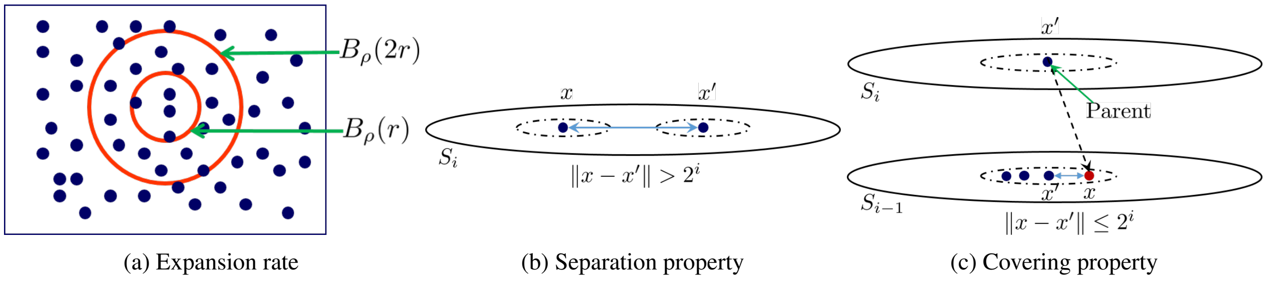


Figure 6. Illustration of various properties of covering tree.

Hence the procedure will draw from the true posterior $p(z|x)$.

The above describes a rejection sampler that keeps on upper-bounding the probability of accepting a particular parameter or any of its children. It is as aggressive as possible at retaining tight lower bounds on the *acceptance* probability such that not too much effort is wasted in traversing the cover tree to he bottom. In other words, we attempt to reject as quickly as possible. Some computational considerations are in order:

1. The computationally most expensive part is to compute the inner products $\left\langle \tilde{\phi}(x), \tilde{\theta}_z \right\rangle$.
2. As soon as we compute this value for a particular $\tilde{\theta}_z$ we cache it at the corresponding vertex of the cover tree.
3. To avoid expensive bookkeeping we attach to each vertex two variables: the value of the last compute inner product and the observation ID of x that it is associated with. +

C. Cover Trees

Cover Trees (Beygelzimer et al., 2006) and their improved version (Izbicki & Shelton, 2015) form a hierarchical data structure that allows fast retrieval in logarithmic time. The key properties for the purpose of this paper are that it allows for $O(n \log n)$ construction time, $O(\log n)$ retrieval, and that it only depends polynomially on the expansion rate (Karger & Ruhl, 2002) of the underlying space, which we refer to as c. Moreover, the degree of all internal nodes is well controlled, thus giving guarantees for retrieval (as exploited in (Beygelzimer et al., 2006)), and for sampling (as we will be using in this paper).

The expansion rate of a set, due to (Karger & Ruhl, 2002) captures several key properties.

Definition 2 (Expansion Rate) *Denote by $B_\rho(r)$ a ball of radius of r centered at ρ. Then a set S has a (l, c) expansion rate iff all $r > 0$ and $\rho \in S$ satisfy*

$$|B_\rho(r) \cap S| \ge l \Longrightarrow |B_\rho(2r) \cap S| \le c\,|B_\rho(r) \cap S|\,. \tag{21}$$

In the following we set $l = O(\log |S|)$, thus referring to c simply as the expansion rate of S.

Cover trees are defined as an infinite succession of levels S_i with $i \in \mathbb{Z}$. Each level i contains (a nested subset of) the data with the following properties:

- Nesting property: $S_i \subseteq S_{i-1}$.
- Separation property: All $x, x' \in S_i$ satisfy $\|x - x'\| \ge 2^i$.
- All $x \in S_{i-1}$ have a parent in $x' \in S_i$, possibly with $x = x'$, with $\|x - x'\| \le 2^i$.
- As a consequence, the subtree for any $x \in S_i$ has distance at most 2^{i+1} from x.

Clearly we need to reperesent each x only once, namely in terms of S_i with the largest i for which $x \in S_i$ holds. This data structure has a number of highly desirable properties, as proved in (Beygelzimer et al., 2006). We list the most relevant ones below:

- The depth of the tree in terms of its explicit representation is at most $O(c^2 \log n)$.

- The maximum degree of any node is $O(c^4)$.
- Insertion & removal take at most $O(c^6 \log n)$ time.
- Retrieval of the nearest neighbor takes at most $O(c^{12} \log n)$ time.
- The time to construct the tree is $O(c^6 n \log n)$.

The fast lookup of cover tree is built upon the implicit assumption in terms of the distinguishability of parameters θ_z, which we also borrow in Canopy. This is related to the issue that if we had many choices of θ_z that, a-priori, all looked quite relevant yet distinct, we would have no efficient means of evaluating them short of testing all by brute force. Note that this could be achieved, e.g. by using the fast hash approximation of a sampler in (Ahmed et al., 2012). This is complementary to the present paper.

D. Theoretical Analysis

Some more conclusions we can make about the algorithm Canopy I:

Remark 3 (Rejection Sampler) *The same reasoning yields a rejection sampler since*

$$\frac{p(z|\bar{x})}{p(z|x)} \geq e^{-\|\phi(x)-\phi(\bar{x})\|\|\theta_z\|} \geq e^{-2^{\bar{\jmath}+1}L}. \tag{22}$$

Here we may bound each term (and the normalization) in computing $p(z|x)$ appropriately.

Remark 4 *The efficiency of the sampler increases as the sample size m increases. In particular, an increase of m by $O(c^4)$ is guaranteed to decrease $\bar{\jmath}$ by 1, thus increasing the acceptance probability π from π to $\sqrt{\pi}$. This follows from the fact that each node in the cover tree has at most $O(c^4)$ children.*

Remark 5 *There is no need to build a cover tree to a level beyond $\bar{\jmath}$ since we do not exploit the improvement. This could be used to remove the logarithmic dependence $O(n \log n)$ in constructing the cover tree and reduce it to $O(n\bar{\jmath})$.*

E. Feature Extraction

E.1. Denoising Autoencoder for MNIST

The autoencoder consists of an encoder with fully connected layers of size (28x28)-1000-500-250-30 and a symmetric decoder. The thirty units in the code layer were linear and all the other units were logistic. The network was trained on the 8 million images using mean square error loss.

E.2. Denoising Autoencoder for CIFAR100

The autoencoder consists of an encoder with convolutional layers of size (3x32x32)-(64, 5, 5)-(32, 5, 5)-(16, 4, 4) and having a 2x2 max pooling after each convolutional layer. The decoder is symmetric with max pooling replaced by upsampling. The 256 units in the code layer were linear and all the other internal units were RelU while the final layer was sigmoid. The network was trained on the 50 thousand images using mean square error loss.

E.3. ResNet for ImageNet

We use the state of the art deep convolutional neural network (DCNN), based on the ResNet (”Residual Network”) architecture (He et al., 2015; 2016). ResNet consists of small building blocks of layers which learn the residual functions with reference to the input. It is demonstrated that ResNet is able to train networks that are substantially deeper without the problem of noisy backpropagation gradient. For feature extraction We use a 200 layer ResNet that is trained on a task of classification on ImageNet. In the process, the network learned which high-level visual features (and combinations of those features) are important. After training the model, we remove the final classification layer of the network and extract from the next-to-last layer of the DCNN, as the representation of the input image which is of dimension 2048.

F. Further Experimental Results

MNIST8m - Direct										
			Random I		Random II		KMeans++		CoverTree	
Clusters	**Method**	**s/iter**	**LLH**	**Purity**	**LLH**	**Purity**	**LLH**	**Purity**	**LLH**	**Purity**
10	EM	39.588 ± 1.801	3.04×10^7	32.39%	3.05×10^7	30.76%	3.04×10^7	30.81%	3.05×10^7	30.50%
	SEM	7.124 ± 0.241	3.04×10^7	32.33%	3.03×10^7	30.65%	3.04×10^7	30.61%	3.04×10^7	31.69%
	Canopy I	7.453 ± 0.255	1.49×10^7	42.12%	1.49×10^7	40.51%	1.49×10^7	40.41%	1.50×10^7	42.84%
	Canopy II	7.534 ± 0.320	1.49×10^7	42.85%	1.49×10^7	40.69%	1.49×10^7	40.95%	1.50×10^7	42.59%
100	EM	512.185 ± 13.295	3.27×10^7	53.20%	3.26×10^7	53.24%	3.28×10^7	52.45%	3.32×10^7	53.10%
	SEM	10.085 ± 0.162	3.34×10^7	53.19%	3.34×10^7	53.21%	3.34×10^7	52.42%	3.33×10^7	53.52%
	Canopy I	6.882 ± 0.174	2.02×10^7	53.39%	2.04×10^7	53.53%	2.01×10^7	53.88%	2.02×10^7	52.69%
	Canopy II	6.483 ± 0.298	1.91×10^7	60.19%	1.90×10^7	61.09%	1.90×10^7	60.61%	1.90×10^7	60.29%
MNIST8m - Embedding										
			Random I		Random II		KMeans++		CoverTree	
Clusters	**Method**	**s/iter**	**LLH** $(\times 10^7)$	**Purity**	**LLH** $(\times 10^7)$	**Purity**	**LLH** $(\times 10^7)$	**Purity**	**LLH** $(\times 10^7)$	**Purity**
10	EM	6.595 ± 0.230	-4.35×10^5	58.43%	-4.36×10^5	63.14%	-4.35×10^5	63.19%	-4.34×10^5	63.22%
	SEM	0.943 ± 0.037	-4.35×10^5	58.43%	-4.35×10^5	62.05%	-4.36×10^5	61.44%	-4.35×10^5	60.58%
	Canopy I	0.932 ± 0.027	-4.35×10^5	58.78%	-4.36×10^5	61.61%	-4.35×10^5	64.46%	-4.35×10^5	58.78%
	Canopy II	1.008 ± 0.053	-4.35×10^5	58.78%	-4.35×10^5	62.30%	-4.36×10^5	61.69%	-4.35×10^5	58.78%
100	EM	56.640 ± 1.060	-3.93×10^5	83.95%	-3.94×10^5	82.33%	-3.94×10^5	83.44%	-3.94×10^5	82.77%
	SEM	4.006 ± 0.050	-3.93×10^5	83.99%	-3.93×10^5	83.37%	-3.94×10^5	83.05%	-3.95×10^5	83.44%
	Canopy I	1.220 ± 0.025	-3.96×10^5	83.44%	-3.96×10^5	83.20%	-3.97×10^5	83.48%	-3.96×10^5	83.22%
	Canopy II	1.015 ± 0.029	-3.97×10^5	82.77%	-3.97×10^5	83.21%	-3.97×10^5	82.66%	-3.97×10^5	82.66%
CIFAR100 - Direct										
			Random I		Random II		KMeans++		CoverTree	
Clusters	**Method**	**s/iter**	**LLH**	**Purity**	**LLH**	**Purity**	**LLH**	**Purity**	**LLH**	**Purity**
100	EM	78.019 ± 10.702	2.86×10^6	14.27%	3.03×10^6	13.31%	3.09×10^6	13.84%	3.09×10^6	14.19%
	SEM	1.055 ± 0.095	2.93×10^6	14.08%	2.93×10^6	14.12%	2.86×10^6	14.75%	3.00×10^6	14.90%
	Canopy I	1.027 ± 0.095	3.20×10^6	12.98%	3.36×10^6	12.43%	3.21×10^6	13.55%	3.25×10^6	12.91%
	Canopy II	1.190 ± 0.099	2.99×10^6	12.87%	3.08×10^6	13.23%	3.28×10^6	13.72%	3.08×10^6	12.87%
500	EM	407.764 ± 18.160	3.37×10^6	25.19%	3.31×10^6	24.70%	3.27×10^6	26.03%	3.31×10^6	25.59%
	SEM	6.486 ± 0.613	3.39×10^6	25.14%	3.30×10^6	24.33%	3.36×10^6	26.16%	3.22×10^6	25.39%
	Canopy I	2.745 ± 0.225	3.38×10^6	22.35%	3.50×10^6	22.14%	3.45×10^6	24.03%	3.44×10^6	22.31%
	Canopy II	1.908 ± 0.152	3.17×10^6	22.68%	3.18×10^6	22.83%	3.19×10^6	24.91%	3.19×10^6	22.71%
CIFAR100 - Embedding										
			Random I		Random II		KMeans++		CoverTree	
Clusters	**Method**	**s/iter**	**LLH**	**Purity**	**LLH**	**Purity**	**LLH**	**Purity**	**LLH**	**Purity**
100	EM	12.589 ± 0.255	5.45×10^5	12.38%	5.50×10^5	12.14%	5.50×10^5	12.25%	5.46×10^5	12.59%
	SEM	0.491 ± 0.022	5.46×10^5	12.21%	5.53×10^5	11.57%	5.45×10^5	12.72%	5.47×10^5	12.68%
	Canopy I	0.315 ± 0.014	5.01×10^5	12.34%	5.04×10^5	11.96%	4.99×10^5	13.16%	5.06×10^5	12.30%
	Canopy II	0.313 ± 0.124	5.00×10^5	12.50%	5.02×10^5	11.97%	4.99×10^5	13.01%	5.02×10^5	12.29%
500	EM	62.520 ± 1.135	6.94×10^5	19.17%	6.96×10^5	18.93%	6.86×10^5	21.13%	6.86×10^5	21.05%
	SEM	2.276 ± 0.112	6.92×10^5	18.97%	6.93×10^5	18.64%	6.85×10^5	21.16%	6.85×10^5	21.20%
	Canopy I	0.963 ± 0.061	6.25×10^5	20.07%	6.21×10^5	19.19%	6.14×10^5	21.57%	6.24×10^5	20.04%
	Canopy II	0.333 ± 0.101	6.20×10^5	22.26%	6.16×10^5	21.61%	6.12×10^5	23.18%	6.18×10^5	22.25%

Table 1. Comparison of ESCA, Canopy I and Canopy II on cluster purity and loglikelihood on real, benchmark datasets–MNIST8m and CIFAR-100. Additionally, standard deviations are shown for 5 runs.

F.1. Image Clustering

We sample images from varied sized clusters, as described below, to study the semantic concept they usually represent: (a) $> 10k$ **members:** As our dataset is extracted from Flickr, a photo sharing platform, it is heavily biased towards everyday objects like humans, flowers, birds, *etc*. We found several consistent clusters containing people (sitting, standing, crowd). (b) > 5 **but** $< 10k$ **members:** These contains less common semantic groups like swings, transmission lines, *etc*, out of which some are absent as explicit concepts in underlying Resnet model. (c) < 5 **members:** We found around 15% small sized clusters which are typically outliers containing less than 5 images. Fig. 7 contains more examples.

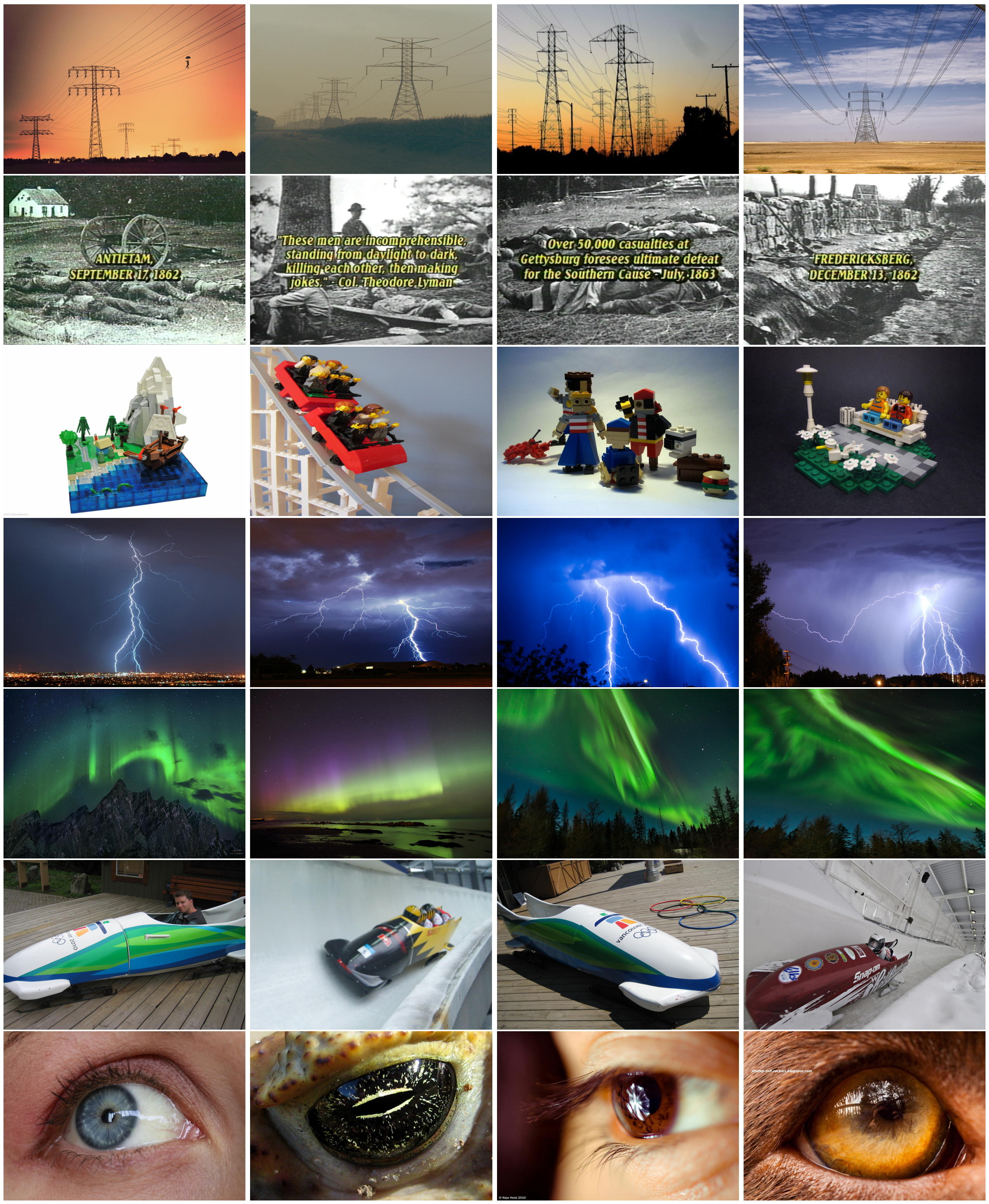

ANTIETAM, SEPTEMBER 17, 1862
"These men are incomprehensible, standing from daylight to dark, killing eachother, then making jokes." - Col. Theodore Lyman
Over 50,000 casualties at Gettysburg foresees ultimate defeat for the Southern Cause - July, 1863
FREDERICKSBERG, DECEMBER 13, 1862

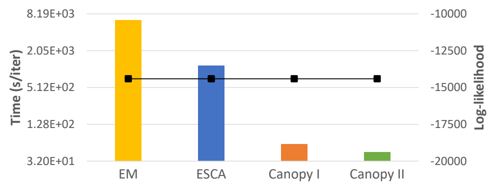
Figure 7. Illustration of concepts captured by clustering images in the feature space extracted by ResNet (He et al., 2015; 2016). Figure shows four closest images of seven more randomly selected clusters (one in each row) possibly denoting the semantic concepts of 'electrical transmission lines', 'image with text', 'lego toys', 'lightening', 'Aurora', 'buggy' and 'eyes'. Few of the concepts are discovered by clustering as Resnet received supervision only for 1000 categories (for example does not include label 'lightening', 'thunder', or 'storm'). Full set of 1000 imagenet label can be seen at `http://image-net.org/challenges/LSVRC/2014/browse-synsets`.

G. Graphical Explanation

We now present insights about our approach graphically.

Motivation

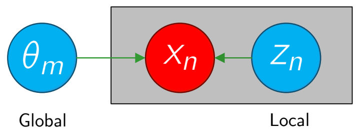
- Latent variable models (LVM), such as Mixture Models, Latent Dirichlet Allocation, are popular tools in statistical data analysis.
- They are used in diverse fields ranging from text, images, to user modelling and content recommendations.
- Inference is often slow



Inference Strategy

- Inference using Gibbs sampling, stochastic EM, or stochastic variational methods requires drawing from

$$p(z|x,\theta) \propto p(z)p(x|z,\theta)$$



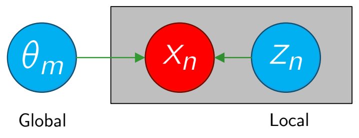
Inference Strategy

- Inference using Gibbs sampling, stochastic EM, or stochastic variational methods requires drawing from

$$p(z|x,\theta) \propto p(z)p(x|z,\theta)$$

- Assume exponential family, i.e.

$$p(x|z,\theta) = \exp(\langle\phi(x), \theta_z\rangle - g(\theta_z))$$



Insights

- For example assume we have following data:

Insights

- For example assume we have following data:

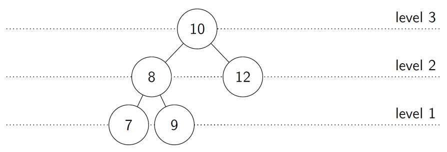
- Two key observations
  - Points close by will have similar posteriors
  - No need to consider clusters far away

Insights

- For example assume we have following data:

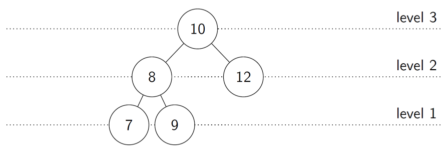
- Two key observations
  - Points close by will have similar posteriors
  - No need to consider clusters far away
- Two tools to exploit the observations
  - Cover trees
  - Metropolis Hasting sampling

Cover Tree

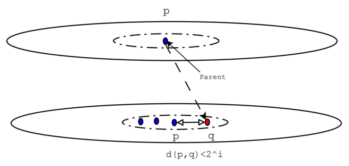


- Cover tree is a hierarchical data structure

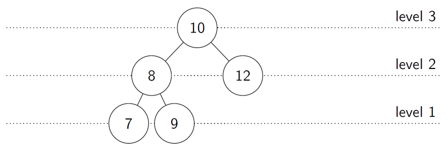
Cover Tree



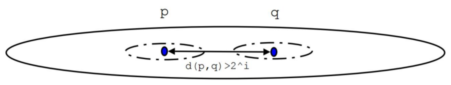
- Cover tree is a hierarchical data structure
- Covering property: $d(p,q) < 2^{\text{level}(p)}$



Cover Tree



- Cover tree is a hierarchical data structure
- Covering property: $d(p,q) < 2^{\text{level}(p)}$
- Separating property: $d(p,q) > 2^{\text{level}(p)-1}$



Computational Cost of Cover Trees

	Cover Tree	Ball Tree
Construction space	$O(n)$	$O(n)$
Construction time	$O(c^6 n \log n)$	$O(n^2)$
Insertion time (1 pt)	$O(c^6 \log n)$	$O(n)$
Query time (1 pt)	$O(c^{12} \log n)$	$O(n)$
Query time (n pts)	$O(c^{16} n)$	$O(n^2)$
	Beygelzimer *et. al.*, 2006	Omohundro, 1989

- Does not depend on the dimension of the data
- c: Expansion rate of data or Hausdorff dimension (special case of fractal dimension)

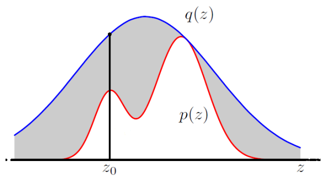
Insights

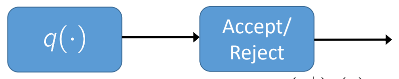
- For example assume we have following data:

- Two key observations
  - Points close by will have similar posteriors
  - No need to consider clusters far away
- Two tools to exploit the observations
  - Cover trees
  - Metropolis Hasting sampling

Metropolis Hasting Sampling

- Enables us to construct sound sampler that incorporates our intuitions





Sample from p

An easy to draw distribution

Acceptance probability $= \dfrac{p(z^+)q(z)}{p(z)q(z^+)}$

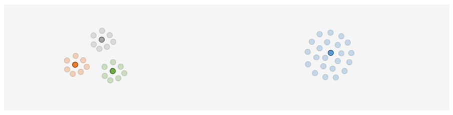
Only need to look at a few probabilities!

How to Design a Good Proposal?

- For example assume we have following data:

How to Design a Good Proposal?

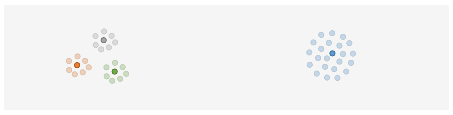
- For example assume we have following data:



- Suppose for each point x we can find surrogates $\bar{x}$
$$\|x - \bar{x}\| < \delta \Rightarrow |p(x|\theta) - p(\bar{x}|\theta)| < \epsilon$$

How to Design a Good Proposal?

- For example assume we have following data:



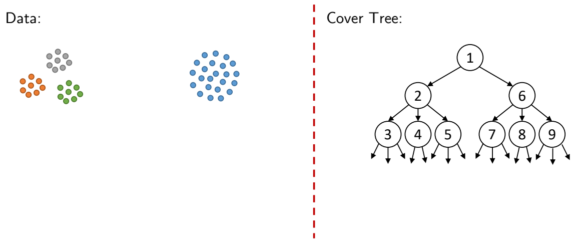
- Suppose for each point x we can find surrogates $\bar{x}$
$$\|x - \bar{x}\| < \delta \Rightarrow |p(x|\theta) - p(\bar{x}|\theta)| < \epsilon$$
- Then $p(\bar{x}|\theta)$ becomes a good proposal for $p(x|\theta)$
  - Compute alias table and re-use for many points
  - Cost for sampling from proposal given alias table is O(1)

Outline

- Background
  - Latent Variable Models
  - Cover tree
  - Metropolis Hastings
- Canopy: Proposed Method
  - Moderate number of clusters
  - Large number of clusters
- Experimental Results
  - Synthetic data
  - Images

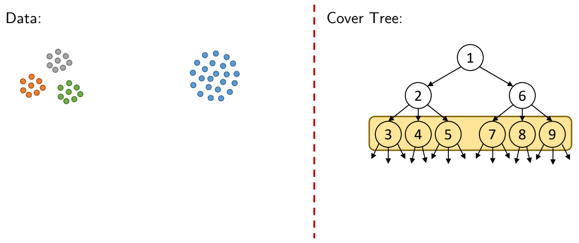
Canopy I – Method 1

- Build a cover tree on data points – Cost $O(N \log N)$



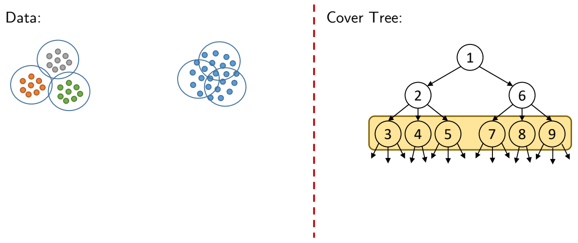
Canopy I – Method 2

- Build a cover tree on data points – Cost $O(N \log N)$
- Pick an accuracy level $\bar{j}$ having $\bar{N} = \Omega(M)$ elements



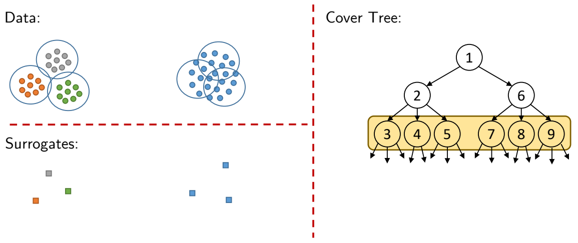
Canopy I – Method 3

- Build a cover tree on data points – Cost $O(N \log N)$
- Pick an accuracy level $\bar{j}$ having $\bar{N} = \Omega(M)$ elements



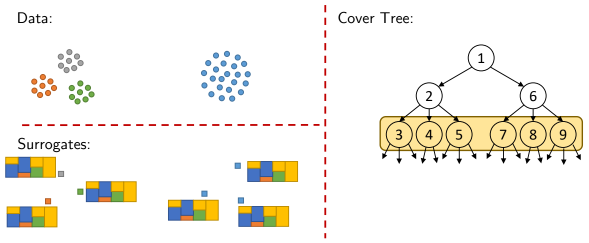
Canopy I – Method 4

- Build a cover tree on data points – Cost $O(N \log N)$
- Pick an accuracy level $\bar{j}$ having $\bar{N} = \Omega(M)$ elements



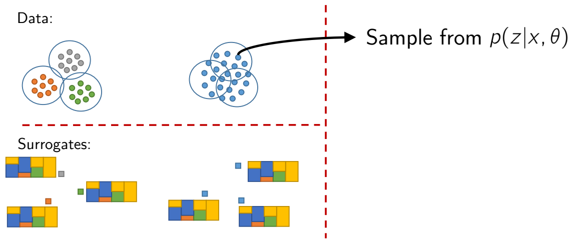
Canopy I – Method 5

- Build a cover tree on data points – Cost $O(N \log N)$
- Pick an accuracy level $\bar{j}$ having $\bar{N} = \Omega(M)$ elements
- Build alias tables for $p(z|\bar{x}, \theta)$ – Cost $O(M\bar{N})$



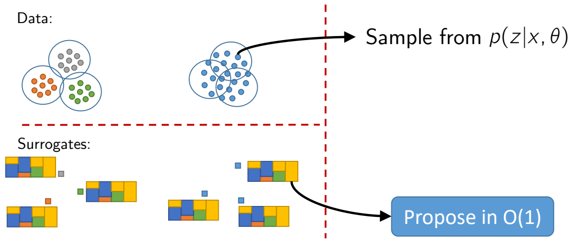
Canopy I – Method 6

- For each observation x perform Metropolis-Hastings



Canopy I – Method 7

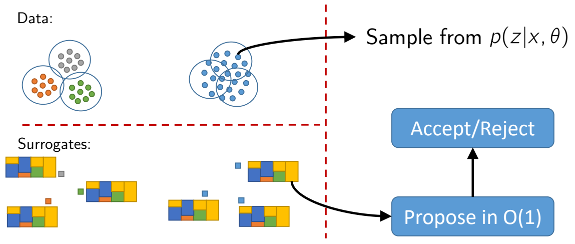
- For each observation x perform Metropolis-Hastings



Canopy I – Method 8

- For each observation x perform Metropolis-Hastings

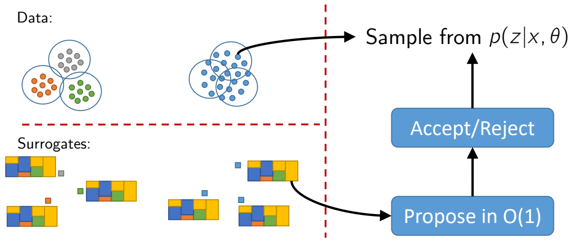
$$\pi := \min\left\{1, \frac{p(z'|x)p(z|\bar{x})}{p(z|x)p(z'|\bar{x})}\right\}$$



Canopy I – Method 9

- For each observation x perform Metropolis-Hastings

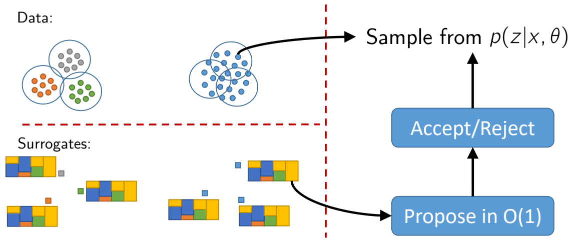
$$\pi := \min\left\{1, \frac{p(z'|x)p(z|\bar{x})}{p(z|x)p(z'|\bar{x})}\right\}$$



Canopy I – Method 10

- For each observation x perform Metropolis-Hastings
  - For exponential families:

$$\pi = \min\{1, \exp\{\langle\phi(x) - \phi(\bar{x}), \theta_{z'} - \theta_z\rangle\} \geq e^{-2^{\bar{i}+2}L}$$

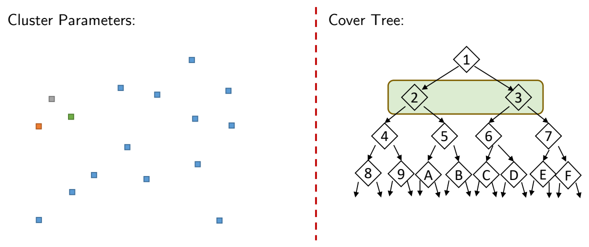


Large Number of Clusters

- Using first insight, cost reduced $O(MN) \to \tilde{O}(N + M\bar{N})$
- When moderate number of clusters, e.g. $M < \sqrt{N}$
  - Choose $\bar{N} = O(N/K)$
  - Then alias table will be used at least K times – full amortization!
  - Total cost $\tilde{O}(N)$
- When there many clusters, e.g. $M > \sqrt{N}$
  - Either high overhead of memory and computation,
  - Or granularity in x that is less precise than desired
- Use second insight: not all clusters are relevant
- Apply cover trees not only to observations but also to the clusters themselves!

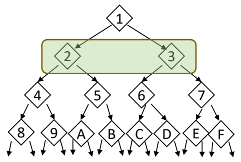
Canopy II – Method 1

- Build a cover tree on cluster parameters – Cost $O(M \log M)$
- Pick an accuracy level $\bar{i}$

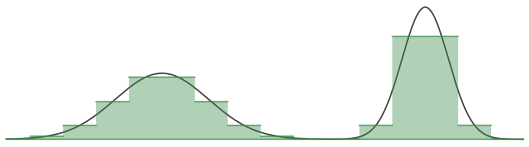


Canopy II – Method 2

- The nodes at the selected accuracy level $\bar{i}$ act as coarse approximation to the posterior

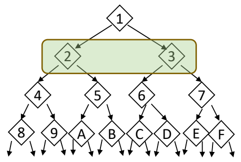


- (Sec 3.2.2 of paper)

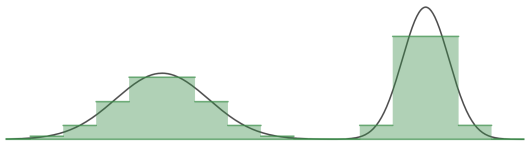


Canopy II – Method 3

- The nodes at the selected accuracy level $\bar{i}$ act as coarse approximation to the posterior
- Treat this as a proposal for a rejection sampler

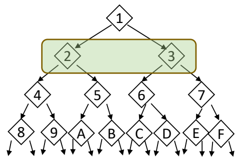


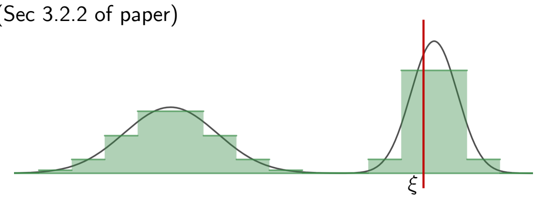
- (Sec 3.2.2 of paper)



Canopy II – Method 4

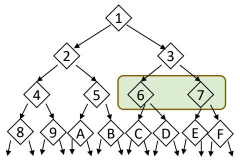
- The nodes at the selected accuracy level $\bar{i}$ act as coarse approximation to the posterior
- Treat this as a proposal for a rejection sampler
- Sample from the proposal
- (Sec 3.2.2 of paper)

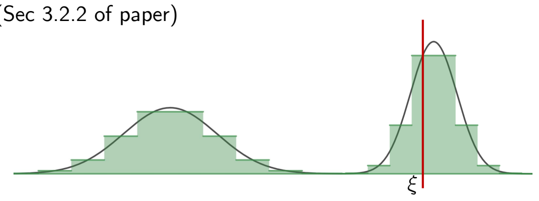




Canopy II – Method 5

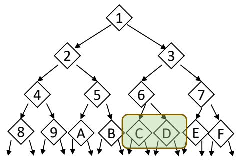
- If the proposed sample is accepted, exit
- Else descend down the tree, and obtain a finer proposal around the region of interest
- (Sec 3.2.2 of paper)

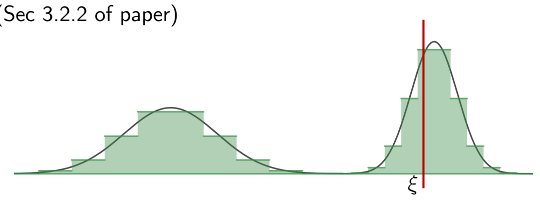




Canopy II – Method 6

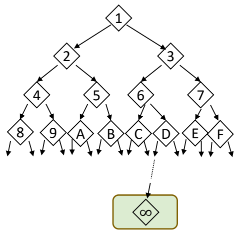
- If the proposed sample is accepted, exit
- Else descend down the tree, and obtain a finer proposal around the region of interest
- (Sec 3.2.2 of paper)

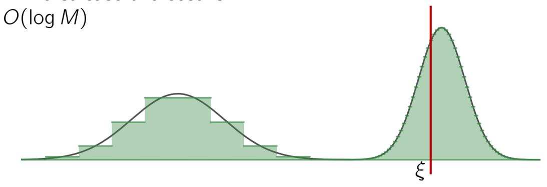




Canopy II – Method 7

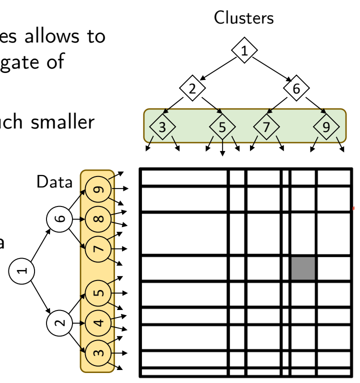
- Sampler is as aggressive as possible in rejecting early on such that not much effort is wasted in traversing the tree
- The deeper we descend into the tree, the less likely we reject
- In worst case the cost is $O(\log M)$





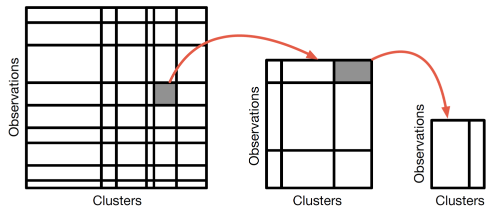
Canopy II – Full Picture

- Using both the trees allows to deal with an aggregate of clusters and data
- This leads to a much smaller observation group
- Employ a MH scheme as before
- We propose from a distribution where both observations and clusters are grouped



Canopy II – Descending both Trees

- Recursively descend in both the trees while sampling
  - Until number of observations for a given cluster is too small
  - Then use the rejection sampler as describer earlier
- Finally perform a MH accept/reject step
  - Acceptance probabilities are as high as in previous case
- This reduces total cost $O(MN) \to \tilde{O}(M+N)$


Observations
Clusters
Observations
Clusters
Observations
Clusters