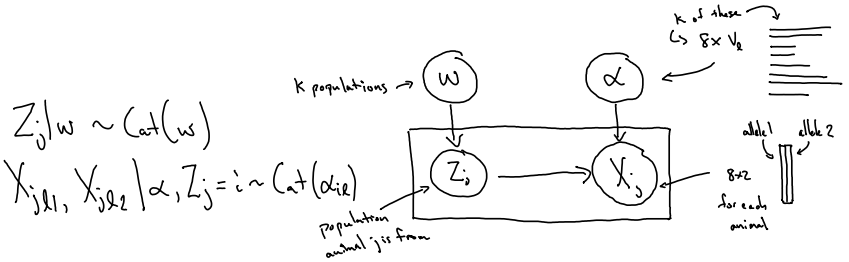
Kevin Liang

Homework 9 - Variational inference

Thursday, April 14, 2016 4:35 PM



$Z_j | w \sim Cat(w)$

$X_{j\ell 1}, X_{j\ell 2} | \alpha, Z_j = i \sim Cat(\alpha_{i\ell})$

$$\begin{aligned}\pi(z,w,\alpha) &= p(z,w,\alpha|x) \\ &\propto p(x,z,w,\alpha) \\ &\propto p(x|z,\alpha)\, p(z|w) \\ &= \prod_{j=1}^{n}\left(\prod_{\ell=1}^{L}\prod_{c=1}^{2}\alpha_{z_j\ell}(x_{j\ell c})\right) w_{z_j}\end{aligned}$$

$$\log(\pi(z,w,\alpha)) = \sum_{j=1}^{n}\sum_{i=1}^{k}\mathbb{1}(z_j=i)\left[\log w_i + \sum_{\ell=1}^{L}\sum_{c=1}^{2}\sum_{v=1}^{V_\ell}\mathbb{1}(x_{j\ell c}=v)\log\alpha_{i\ell}(v)\right] + \text{const}$$

$$q(z,w,\alpha) = q(z)\,q(w,\alpha)$$

Update $q(w,\alpha)$ with $q(z)$ fixed:

$$q(w,\alpha) \propto e^{h(w,\alpha)}$$

$$\begin{aligned}h(w,\alpha) &= \mathbb{E}_q\left[\log\pi(z,w,\alpha)\,|\,w,\alpha\right] \\ &= \sum_{j=1}^{n}\sum_{i=1}^{k}\underbrace{\mathbb{P}_q(z_j=i|w,\alpha)}_{r_j(i)}\left[\log w_i + \sum_{\ell=1}^{L}\sum_{c=1}^{2}\sum_{v=1}^{V_\ell}\mathbb{1}(x_{j\ell c}=v)\log\alpha_{i\ell}(v)\right] + \text{const} \\ &= \sum_{i=1}^{k}\left[\underbrace{\left(\sum_{j=1}^{n}r_j(i)\right)}_{R_i}\log w_i + \sum_{\ell=1}^{L}\sum_{c=1}^{2}\sum_{v=1}^{V_\ell}\underbrace{\left(\sum_{j=1}^{n}r_j(i)\mathbb{1}(x_{j\ell c}=v)\right)}_{S_{i\ell}(v)}\log\alpha_{i\ell}(v)\right]\end{aligned}$$

$$\begin{aligned}q(w,\alpha) &= \left[\prod_{i=1}^{k}w_i^{R_i}\right]\left[\prod_{i=1}^{k}\prod_{\ell=1}^{L}\prod_{c=1}^{2}\prod_{v=1}^{V_\ell}\alpha_{i\ell}(v)^{S_{i\ell}(v)}\right] \\ &= Dirichlet(w|R_1+1,\ldots,R_k+1)\left[\prod_{i=1}^{k}\prod_{\ell=1}^{L}\prod_{c=1}^{2}Dirichlet(\alpha_{i\ell}|S_{i\ell}(1)+1,\ldots,S_{i\ell}(V_\ell)+1)\right]\end{aligned}$$

Update $q(z)$ with $q(w,\alpha)$ fixed

$$q(z) \propto e^{h(z)}$$

$$\begin{aligned}h(z) &\propto \mathbb{E}_q\left[\log\pi(z,w,\alpha)\,|\,z\right] \\ &= \sum_{j=1}^{n}\sum_{i=1}^{k}\mathbb{1}(z_j=i)\Big[\underbrace{\mathbb{E}_q(\log w_i)}_{*} + \overbrace{\mathbb{E}\Big(\sum_{\ell=1}^{L}\sum_{c=1}^{2}\sum_{v=1}^{V_\ell}\mathbb{1}(x_{j\ell c}=v)\underbrace{\log\alpha_{i\ell}(v)}_{\triangle}\Big)}^{u_j(i)}\Big] + \text{const}\end{aligned}$$

$$p(w|R) = \frac{\Gamma\left(\sum_{i=1}^{k}R_i+1\right)}{\sum_{i=1}^{k}\Gamma(R_i+1)}\prod_{i=1}^{k}w_i^{R_i}$$

$$= \exp\left\{ \sum_{i=1}^{k} R_i \log w_i + \underbrace{\log\Gamma\left(\sum_{i=1}^{k} R_i + 1\right) - \sum_{i=1}^{k} \log\Gamma\left(R_i + 1\right)}_{a(R)} \right\}$$

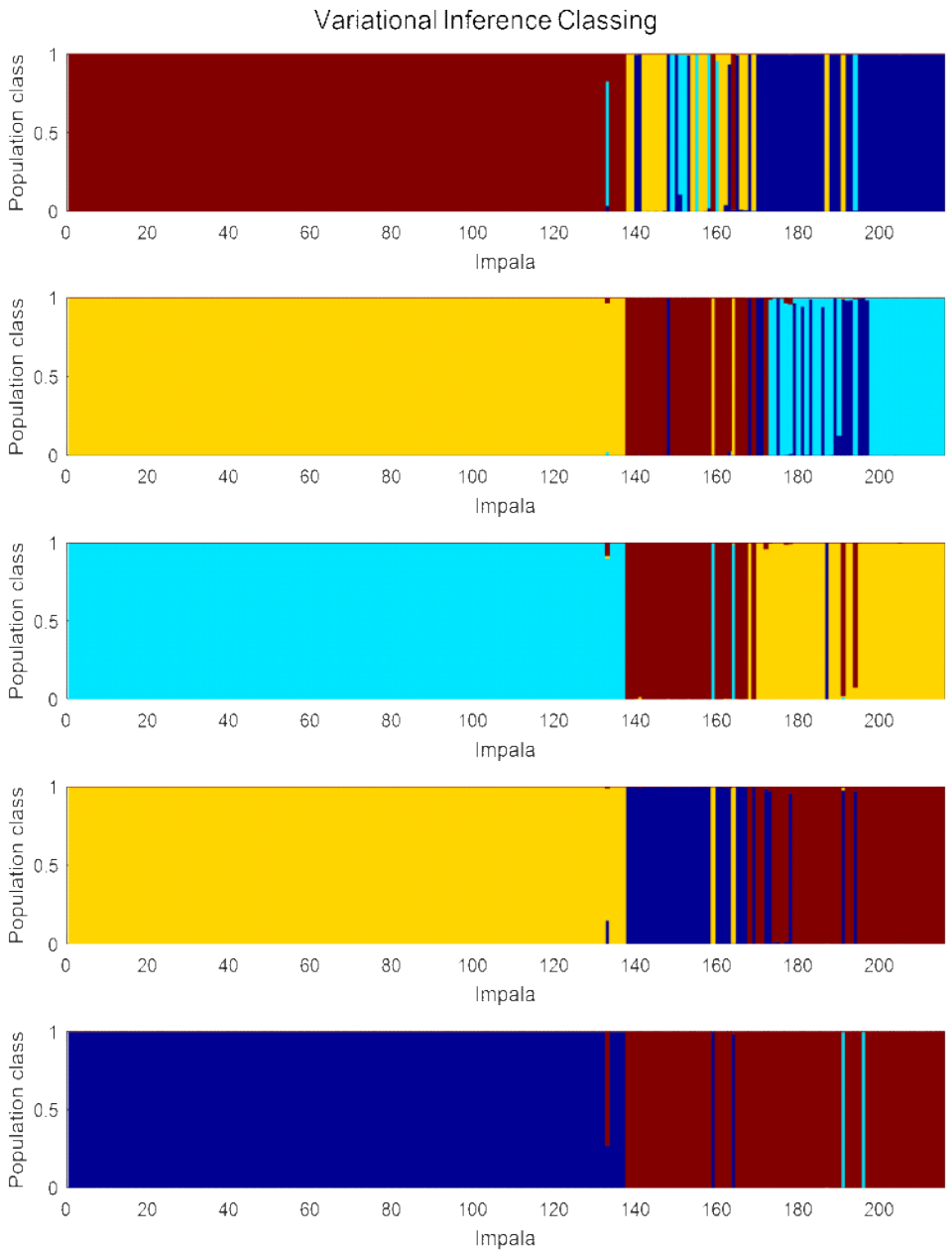
$$\begin{aligned}
\mathbb{E}_q(\log w_i) &= a(R)' \\
&= \left[\sum_{i=1}^{k} \log\Gamma\left(R_i+1\right) - \log\Gamma\left(\sum_{i=1}^{k} R_i + 1\right) \right]' \\
&= \psi\left(R_i+1\right) - \psi\left(\sum_{i=1}^{k} R_i + 1\right) \\
&\overset{*}{=} \psi\left(R_i+1\right) - \psi\left(n+k\right)
\end{aligned}$$

Similarly:

$$\mathbb{E}_q(\log \alpha_{il}) \overset{\Delta}{=} \psi\left(S_{il}(v)+1\right) - \psi\left(\sum_{v=1}^{V_l} S_{il}(v) + 1\right)$$

$$\begin{aligned}
&= \sum_{j=1}^{n} \sum_{i=1}^{k} \mathbb{1}(z_j = i) \left[\psi\left(R_i+1\right) - \psi\left(n+k\right) + \sum_{l=1}^{L} \sum_{x=1}^{J} \sum_{v=1}^{V_l} \mathbb{1}(x_{jl} = v) \left[\psi\left(S_{il}(v)+1\right) - \psi\left(\sum_{v=1}^{V_l} S_{il}(v)+1\right) \right] \right] + \text{const} \\
&= \sum_{j=1}^{n} \sum_{i=1}^{k} \mathbb{1}(z_j = i) \underbrace{\left[\psi\left(R_i+1\right) - \psi\left(n+k\right) + \sum_{l=1}^{L} \sum_{x=1}^{J} \left[\psi\left(S_{il}(x_{jl})+1\right) - \psi\left(\sum_{v=1}^{V_l} S_{il}(x_{jl})+1\right) \right] \right]}_{m_j(i)} + \text{const} \\
&= \sum_{j=1}^{n} m_j(z_j)
\end{aligned}$$

$$\begin{aligned}
q(z) &\propto \prod_{j=1}^{n} e^{m_j(z_j)} \\
&\propto \prod_{j=1}^{n} \frac{e^{m_j(z_j)}}{\sum_{i=1}^{k} e^{m_j(z_j)}} \\
&= \prod_{j=1}^{n} r_j(z_j) \\
&= \prod_{j=1}^{n} \text{Cat}(r_j)
\end{aligned}$$



Run	Iterations until Convergence
1	78
2	122
3	45
4	45
5	44

3. The results of 5 runs of variational inference to determine the class of genotyped impalas, assuming 4 classes, are shown above. Each bar corresponds to probability mass of a particular impala belonging to a population corresponding to the color. Impalas with the same color are similar in genotype and thus likely to be part of the same population. From these plots, there appears to be 2 or 3 distinct clusters of impalas, as there appears to consistently be 2 or 3 distinct blocks of the same color across the different algorithm runs. Because most of the impalas appear to have their bars almost entirely solid, the algorithm appears to be very confident in its population classifying, despite the fact that the results are not perfectly consistent across runs.

4. With the additional information given by Table 2 in the assignment, the results make a great deal of sense. According to Table 2, impalas 1-137 are black-faced impala while 138-216 are common

impala, and indeed, there is a very clear classification boundary in the above plot between 137 and 138. Additionally, knowing that the impala have been grouped by region, these results also suggest that there is a detectable difference in genotype between black-faced impala of certain groups of regions, which based on how these regions are geographically located, could make sense.

The variational inference method seems to have trouble correctly classifying a few impalas, which do not appear to belong to their particular subspecies or region. Impalas 133, 159, 164, and 191 in particular seem to be consistently misclassified, as evidenced by having a different color and sticking out in the bar plots above. This could be due to several reasons. The impala could have traveled far from its home and settled in a different region, or its parents could be from different populations, leading to a mixed genotype. The impala could also have been misclassified or mis-recorded by the scientists conducting the study, in which case the algorithm is right and the differences are due to human error.

Matlab Code:

```
%% STA 531 Problem Set #9
%  Kevin Liang
%
%  Variational Inference

%% Data and probabilities
%  Set random number seed
rng(1337)

% Import data
Homework9_ImportX;

% Data Parameters
% see variationalImpalas.m for descriptions
k = 4;
c = 2;
V = [15,13,6,6,9,14,16,9];
convergence = 10^-10;

% Instantiate object
varImps = variationalImpalas(X,k,c,V,convergence);

% Perform inference 5 times
iterations = NaN(5,1);
figure(1)
clf
colormap(jet)
suptitle('Variational Inference Classing')
for i = 1:5
   iterations(i) = varImps.infer();
   subplot(5,1,i)
   bar(varImps.r,1,'stacked')
   ylabel('Population class')
   xlabel('Impala')
   axis([0 size(X,1) 0 1])
end
```

```
classdef variationalImpalas < handle
   %UNTITLED3 Summary of this class goes here
   %   Detailed explanation goes here

   properties
      % Data
      X
      n          % n: number of animals
      indicator   % ind(X==v)

      % Data Parameters
      k          % k: number of populations
      C          % C: number of alleles/gene
      L          % L: number of locii analyzed
      V          % V: variants of alleles at each locus

      % Algorithm parameters
      convergence    % Convergence Criterion

      % Posterior hyperparameters
      r          % r: Pr(z_j=i|w,alpha)
      R          % R: sum(r_j(i)) over j
      S          % S: sum(r_j(i)*1(x_jlc=v)) over j
      mu         % mu: helper value for r_j
   end

   methods
      %% Constructor
      %  Fill in model parameters and instantiate object
      function obj = variationalImpalas(X,k,C,V,convergence)
         obj.X = X;
         [obj.n,L] = size(X);

         obj.k = k;
         obj.C = 2;
         obj.L = L/C;
         obj.V = V;

         obj.convergence = convergence;

         obj.r = NaN(obj.n,obj.k);
         obj.R = NaN(1,obj.k);
         obj.S = cell(obj.k,obj.L);
         for i = 1:obj.k
            for l = 1:obj.L
               obj.S{i,l} = NaN(obj.C,obj.V(l));
            end
         end
         obj.mu = NaN(obj.n,obj.k);

         obj.indicator = cell(1,obj.L);
         for l = 1:obj.L
            for c = 1:obj.C
               for v = 1:obj.V(l)
                  if X(:,2*(l-1)+c)==v
                     obj.indicator{l}(c,v) = 1;
                  else
```

```
%% Update w and alpha
%  Hyperparameters R and S
function update_w_alpha(obj)
   obj.R = sum(obj.r);
   for i = 1:obj.k
      for l = 1:obj.L
         for c = 1:obj.C
            for v = 1:obj.V(l)
               obj.S{i,l}(c,v) = 0;
               for j = 1:obj.n
                  if obj.X(j,2*(l-1)+c)==v
                     obj.S{i,l}(c,v) = obj.S{i,l}(c,v) + obj.r(j,i);
                  end
               end
            end
         end
      end
   end
end

%% Update z
%  Hyperparameter r
function update_z(obj)
   for j = 1:obj.n
      for i = 1:obj.k
         psiSum = 0;
         for l = 1:obj.L
            for c = 1:obj.C
               for v = 1:obj.V(l)
                  if obj.X(j,2*(l-1)+c)==v
                     psiSum = psiSum + psi(obj.S{i,l}(c,v)+1) - ...
                        psi(sum(obj.S{i,l}(c,:)+1));
                  end
               end
            end
         end
         obj.mu(j,i) = psi(obj.R(i)+1) - psi(obj.n+obj.k) + psiSum;

      end
   end

   obj.r = bsxfun(@rdivide,exp(obj.mu),sum(exp(obj.mu),2));
end

%% Initialize r randomly, clear R and S
function initialize(obj)
   obj.r = rand(obj.n,obj.k);
```

```
      for c = 1:obj.C
        for v = 1:obj.V(l)
          if X(:,2*(l-1)+c)==v
            obj.indicator{l}(c,v) = 1;
          else
            obj.indicator{l}(c,v) = 0;
          end
        end
      end
    end
  end

  %% Infer Parameters using Variational Inference
  function iterations = infer(obj)
    iterations = 0;

    obj.initialize();

    while(true)
      iterations = iterations+1;
      r_old = obj.r;

      obj.update_w_alpha();
      obj.update_z();

      if(obj.converged(r_old))
        break;
      end
    end
  end

    %% Initialize r randomly, clear R and S
    function initialize(obj)
      obj.r = rand(obj.n,obj.k);
      obj.r = bsxfun(@rdivide,obj.r,sum(obj.r,2));
      obj.R = NaN(1,obj.k);
      obj.S = cell(obj.k,obj.L);
      for i = 1:obj.k
        for l = 1:obj.L
          obj.S{i,l} = NaN(obj.C,obj.V(l));
        end
      end
    end

    %% Convergence criterion
    function boolean = converged(obj,r_old)
      rmsDiff = (1/(obj.n*obj.k)*sum(sum((obj.r-r_old).^2))).^(1/2);

      if rmsDiff<obj.convergence
        boolean = true;
      else
        boolean = false;
      end
    end
  end

end
```