Capsule Networks

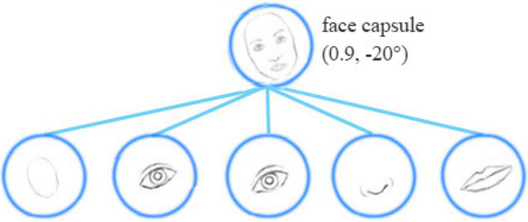
Samuel Cheng

School of ECE
University of Oklahoma

Spring, 2018
(Slides credit to Aurélien Géron)

Main innovation of capsule networks

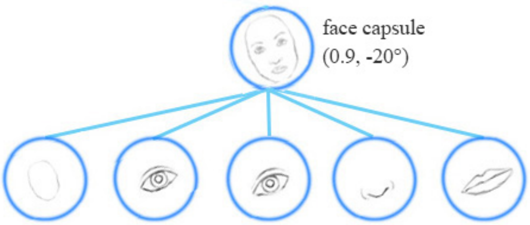
Image credit: J. Hui



- Group neurons into "capsules"
  - A capsule processes a group of information as vector rather than scalar
- "Data-driven routing"
  - Routing weights are neither fixed nor predefined
  - Routing by agreement: later layer can indirectly controlled what is sent to it

Main innovation of capsule networks

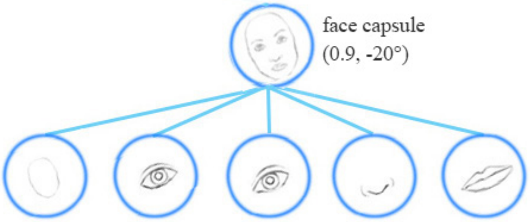
Image credit: J. Hui



- Group neurons into "capsules"
  - A capsule processes a group of information as vector rather than scalar
- "Data-driven routing"
  - Routing weights are neither fixed nor predefined
  - Routing by agreement: later layer can indirectly controlled what is sent to it

Main innovation of capsule networks

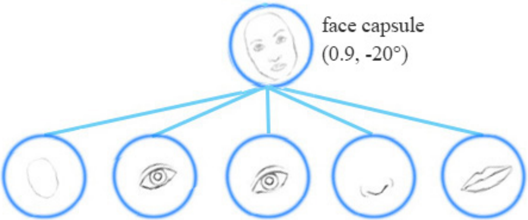
Image credit: J. Hui



- Group neurons into "capsules"
  - A capsule processes a group of information as vector rather than scalar
- "Data-driven routing"
  - Routing weights are neither fixed nor predefined
  - Routing by agreement: later layer can indirectly controlled what is sent to it

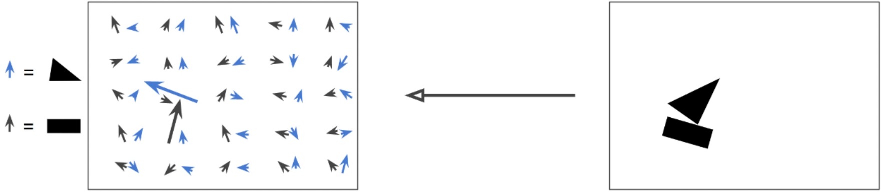
Main innovation of capsule networks

Image credit: J. Hui



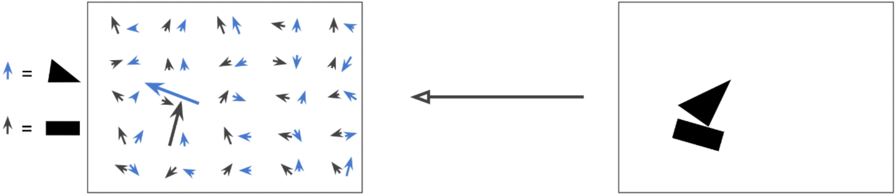
- Group neurons into "capsules"
  - A capsule processes a group of information as vector rather than scalar
- "Data-driven routing"
  - Routing weights are neither fixed nor predefined
  - Routing by agreement: later layer can indirectly controlled what is sent to it

Primary capsules

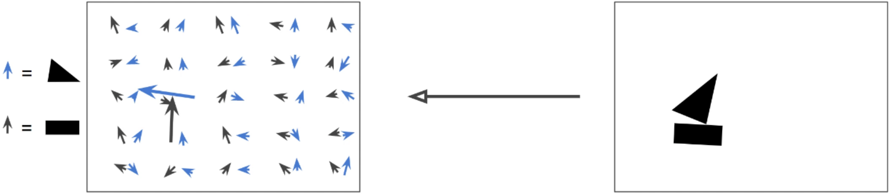


Activation vector: **Length** = estimated probability of presence
Orientation = object's estimated pose parameters

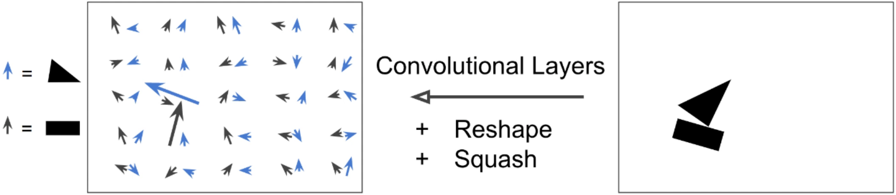
Equivarance



Equivarance



Primary capsules

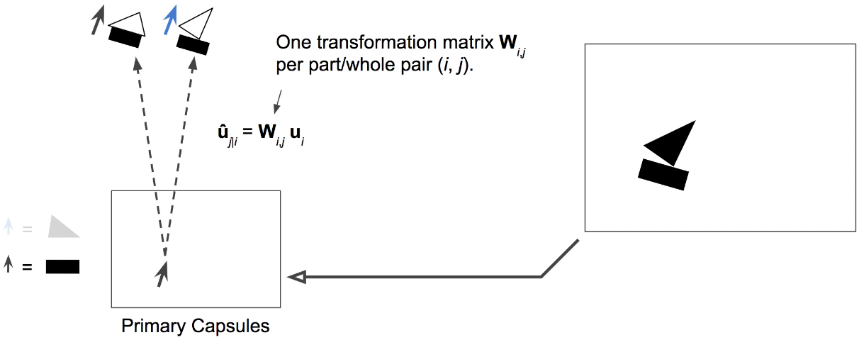


$$\text{Squash}(\mathbf{u}) = \frac{\|\mathbf{u}\|^2}{1+\|\mathbf{u}\|^2}\frac{\mathbf{u}}{\|\mathbf{u}\|}$$

Key idea

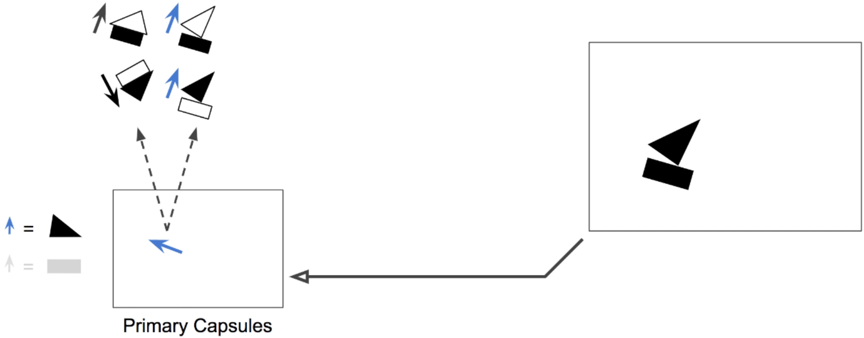
Every capsule in the first layer trying to predict every capsule in the next layer

Predict next layer capsules

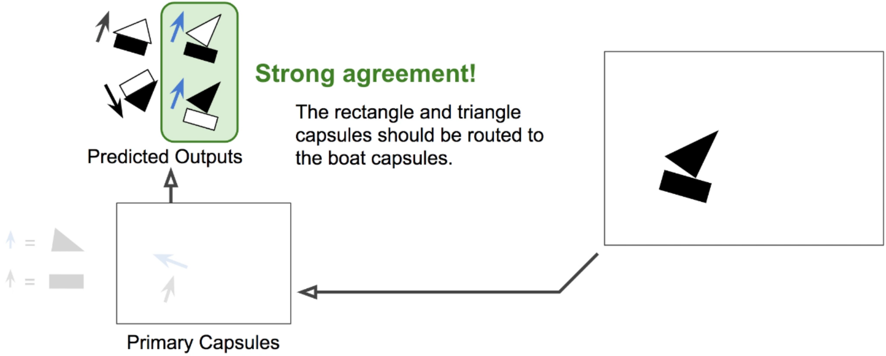


- The transformation matrix $\mathbf{W}_{i,j}$ is supposed to be learned through training

Predict next layer capsules



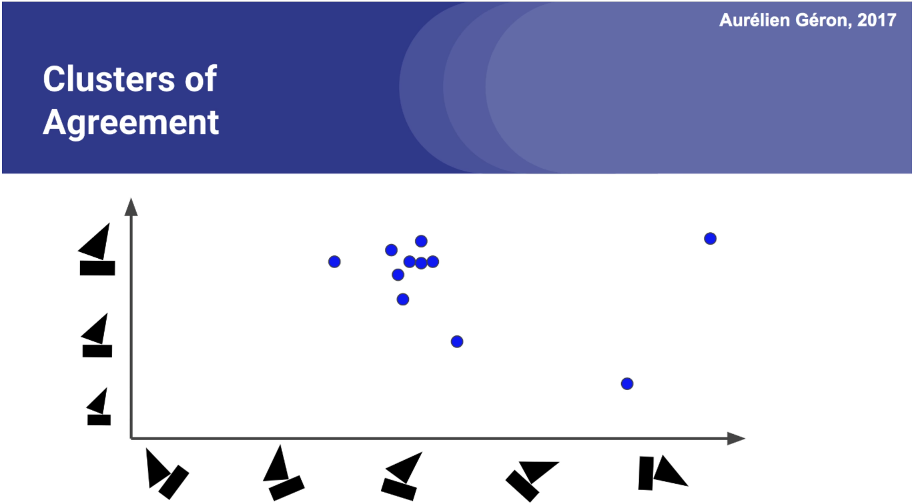
Routing by agreement



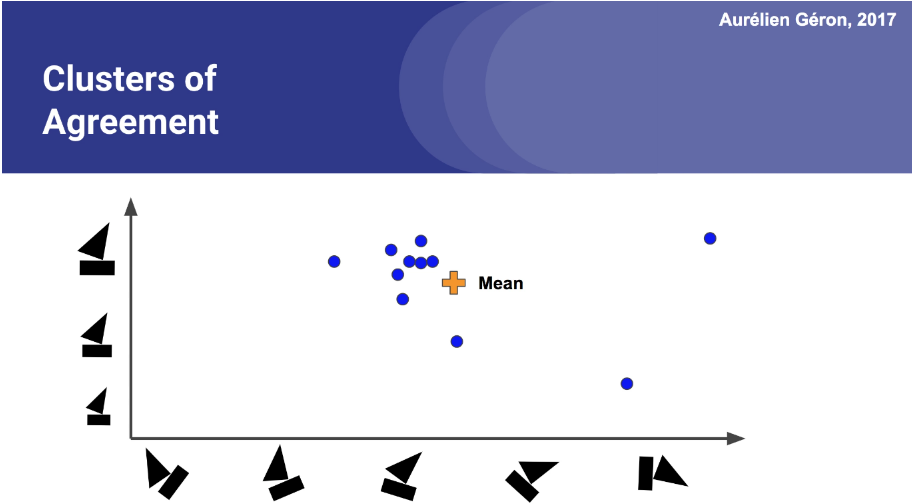
Routing weight adjustment

- Routing weight is adjusted dynamically based on matches between input capsules and output capsules
- The key idea is similar to k-mean

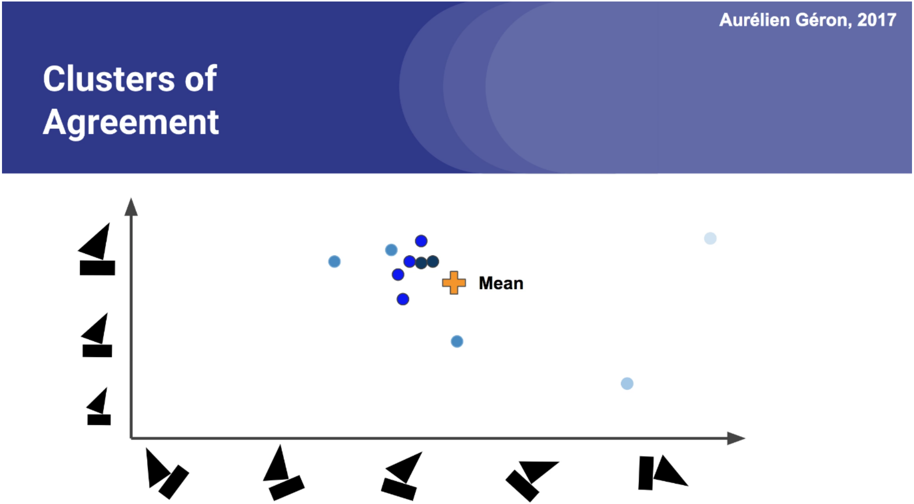
Routing weight adjustment



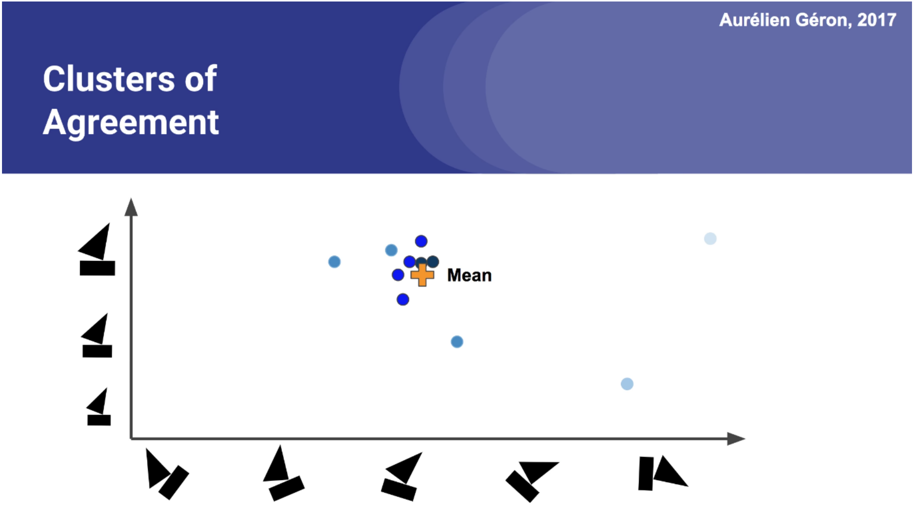
Routing weight adjustment



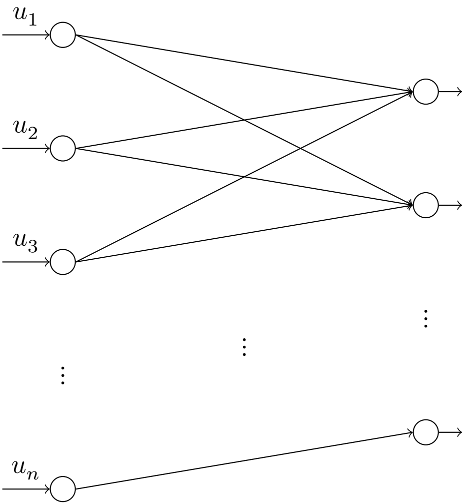
Routing weight adjustment



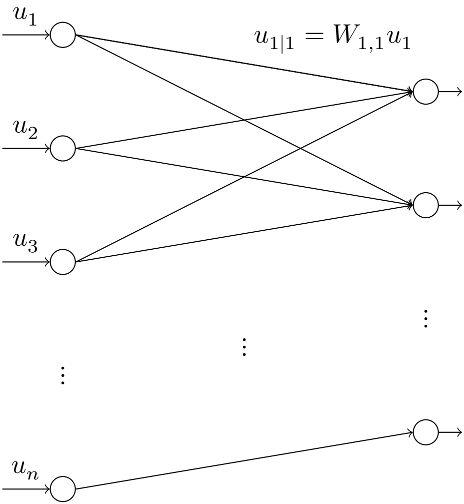
Routing weight adjustment



Capsule update

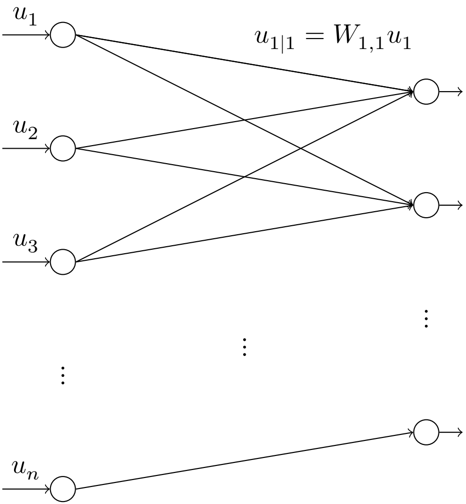


Capsule update



Capsule update

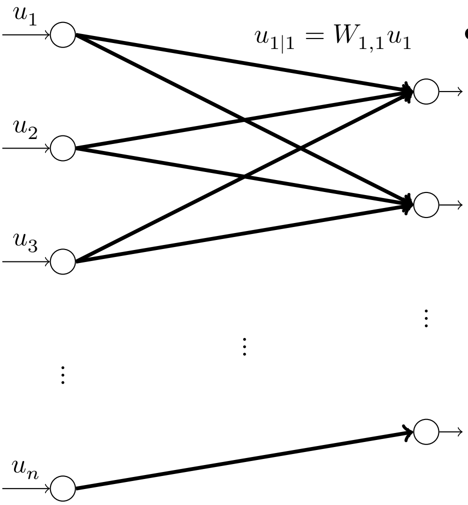
$$b_{1,1} = b_{2,1} = \cdots = b_{n,m} = 0$$



Capsule update

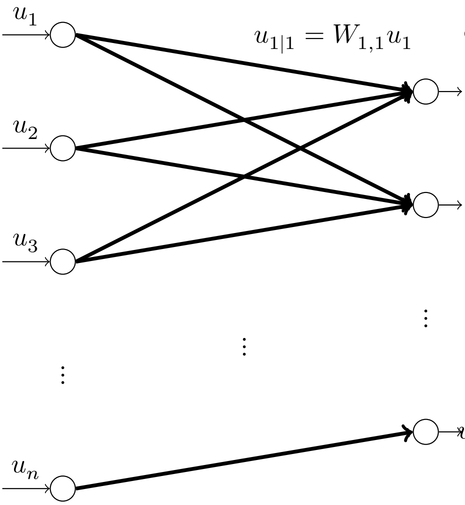
$$b_{1,1} = b_{2,1} = \cdots = b_{n,m} = 0$$

$$\mathbf{c}_{i,:} = \text{softmax}(\mathbf{b}_{i,:})$$



Capsule update

$$b_{1,1} = b_{2,1} = \cdots = b_{n,m} = 0$$



$$\mathbf{c}_{i,:} = \text{softmax}(\mathbf{b}_{i,:})$$

$$v_1 = \text{squash}\left(\sum_{i=1}^{n} c_{i,1} u_{1|i}\right)$$

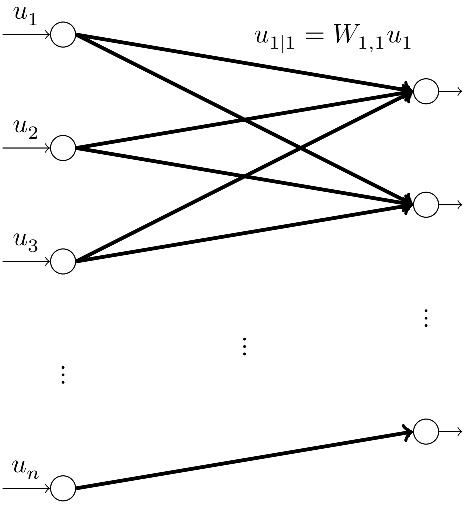
$$v_2 = \text{squash}\left(\sum_{i=1}^{n} c_{i,2} u_{2|i}\right)$$

$$\vdots$$

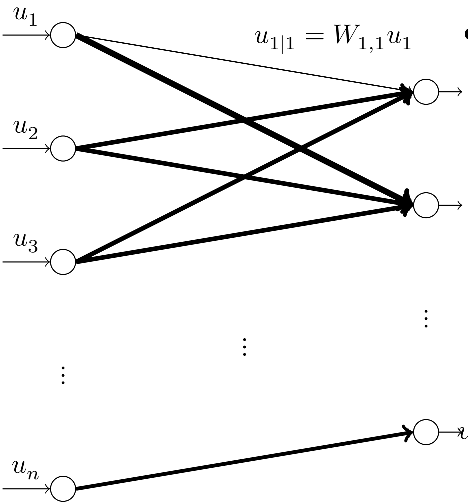
$$v_m = \text{squash}\left(\sum_{i=1}^{n} c_{i,m} u_{m|i}\right)$$

Capsule update

$$b_{i,j} = b_{i,j} + \langle v_j, u_{j|i} \rangle$$



Capsule update



$$b_{i,j} = b_{i,j} + \langle v_j, u_{j|i} \rangle$$

$$\mathbf{c}_{i,:} = \text{softmax}(\mathbf{b}_{i,:})$$

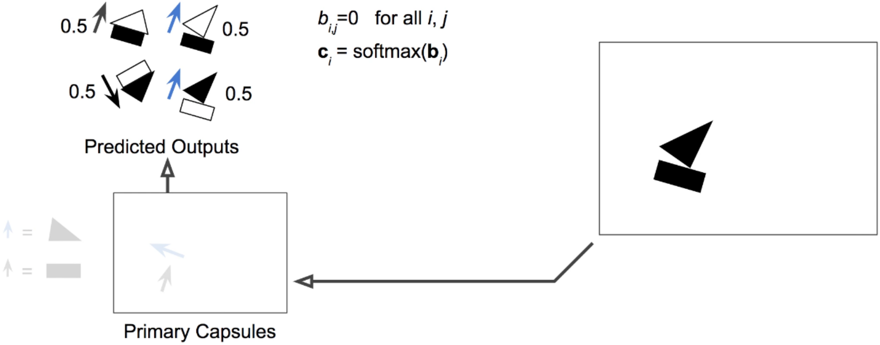
$$v_1 = \text{squash}\left(\sum_{i=1}^n c_{i,1} u_{1|i}\right)$$

$$v_2 = \text{squash}\left(\sum_{i=1}^n c_{i,2} u_{2|i}\right)$$

$$\vdots$$

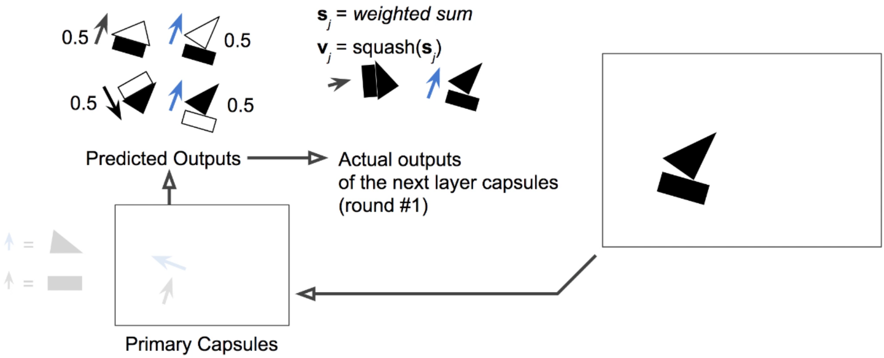
$$v_m = \text{squash}\left(\sum_{i=1}^n c_{i,m} u_{m|i}\right)$$

Toy example revisit



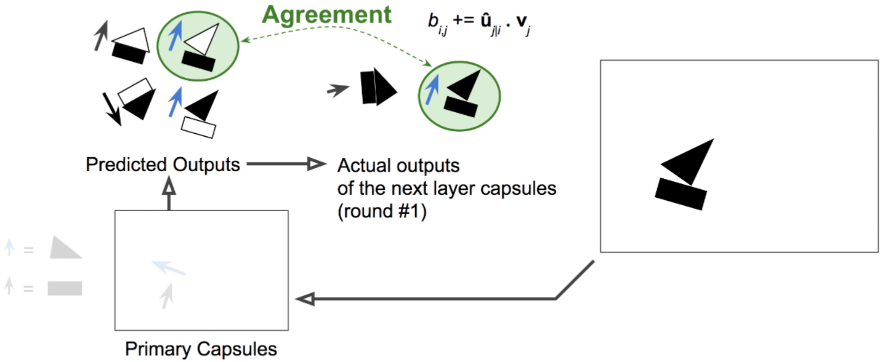
- "Transition" weights $\mathbf{c}$ are normalized with softmax
- The weights are set uniformly at the beginning

Toy example revisit



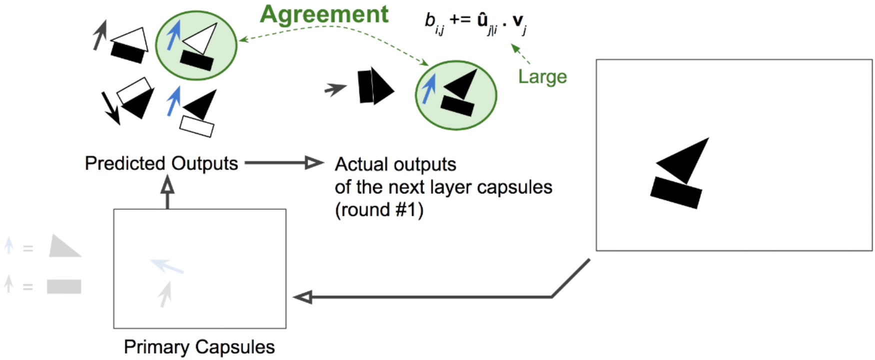
- The prediction of a capsule from all previous level capsules are weighted sum together with the current weight
- A squash function is used to normalized the output predictions

Toy example revisit



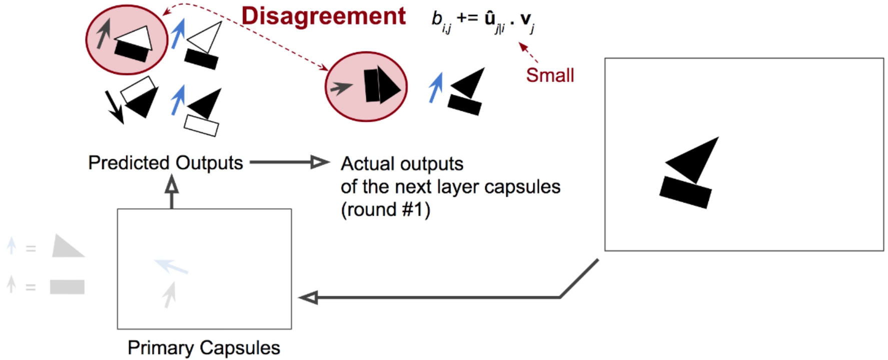
- The level of agreement is estimated by simply computing the dot production between a predicted output $\hat{\mathbf{u}}_{i,j}$ and the actual output $\mathbf{v}_j$ (after weighted sum)
- The transition weight (before softmax) is updated by simply adding the dot product

Toy example revisit



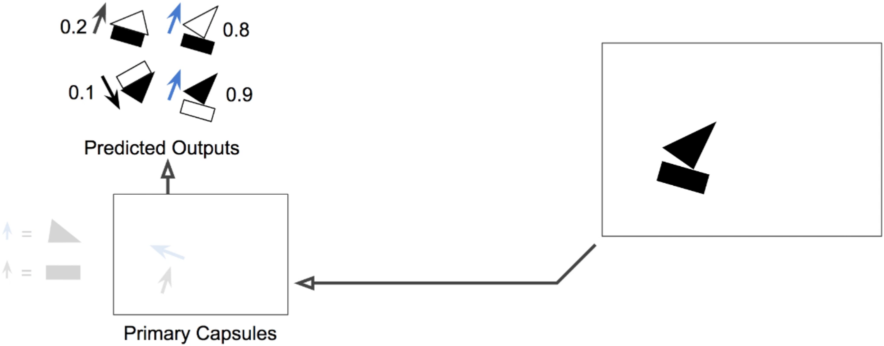
- The level of agreement is estimated by simply computing the dot production between a predicted output $\hat{\mathbf{u}}_{i,j}$ and the actual output $\mathbf{v}_j$ (after weighted sum)
- The transition weight (before softmax) is updated by simply adding the dot product

Toy example revisit



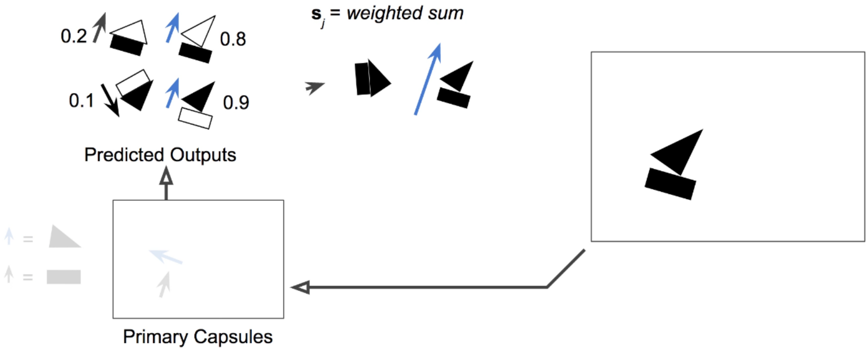
- The level of agreement is estimated by simply computing the dot production between a predicted output $\hat{\mathbf{u}}_{i,j}$ and the actual output $\mathbf{v}_j$ (after weighted sum)
- The transition weight (before softmax) is updated by simply adding the dot product

Toy example revisit



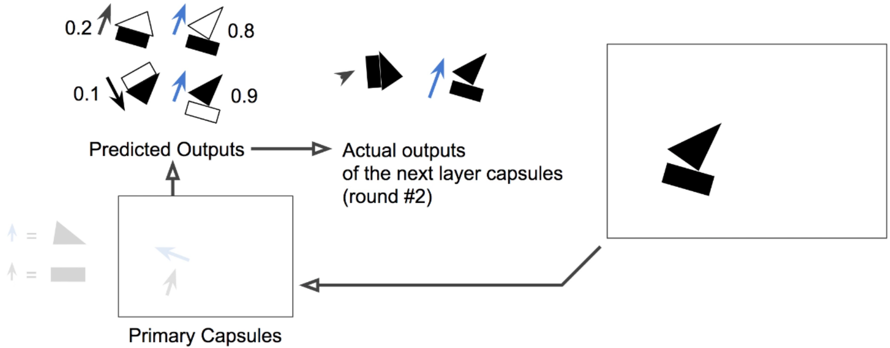
- The whole process is repeated using the updated transition weights

Toy example revisit



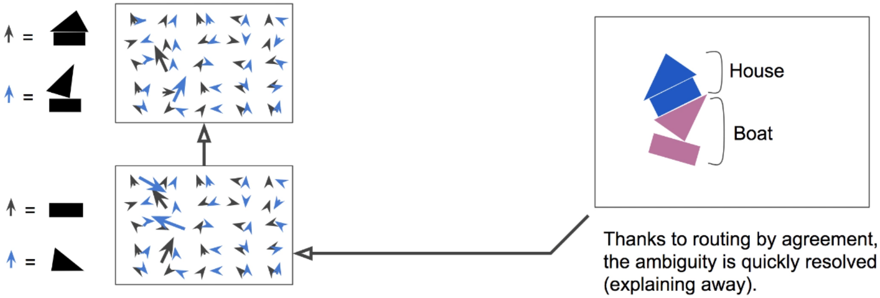
- The whole process is repeated using the updated transition weights

Toy example revisit



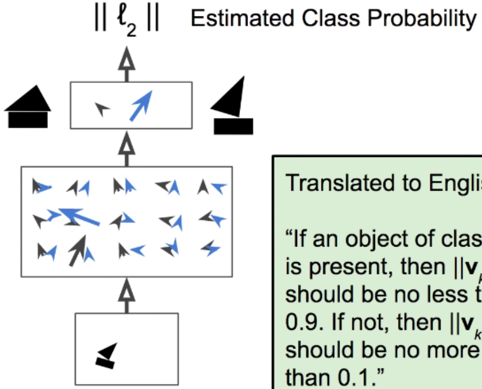
- The whole process is repeated using the updated transition weights

Toy example revisit



- An upside down house exists in the figure
  - But this interpretation is not encouraged since the lower rectangle and the upper triangle cannot be explained this way

Multi-class margin loss



Translated to English:

"If an object of class k is present, then $\|\mathbf{v}_k\|$ should be no less than 0.9. If not, then $\|\mathbf{v}_k\|$ should be no more than 0.1."

To allow multiple classes, minimize margin loss:

$$L_k = T_k \max(0, m^+ - \|\mathbf{v}_k\|)^2 + \lambda (1 - T_k) \max(0, \|\mathbf{v}_k\| - m^-)^2$$

$T_k = 1$ iff class k is present

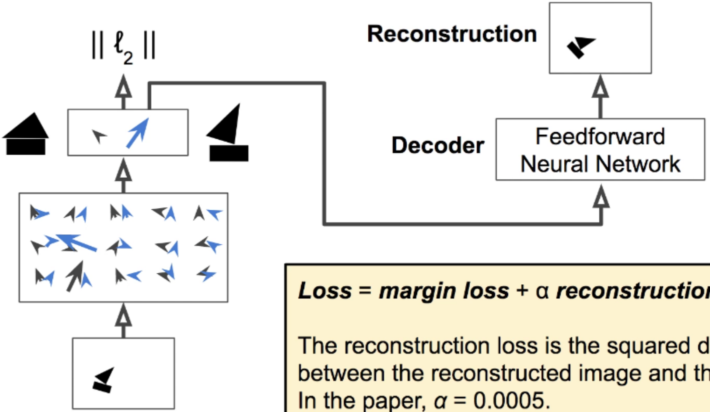
In the paper:

$m^- = 0.1$

$m^+ = 0.9$

$\lambda = 0.5$

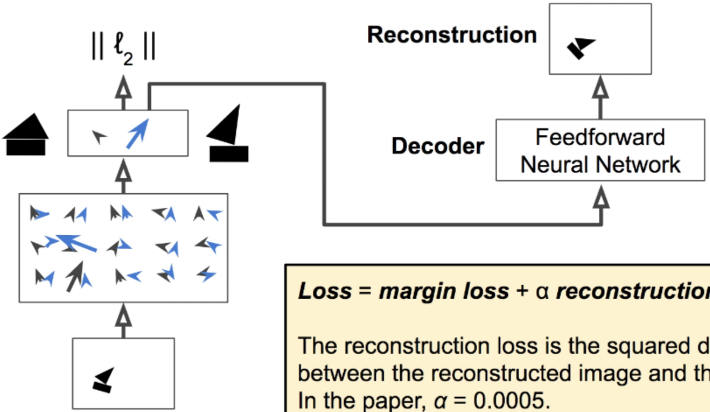
Net loss



Loss **=** ***margin loss*** **+ α** ***reconstruction loss***

The reconstruction loss is the squared difference between the reconstructed image and the input image. In the paper, $\alpha = 0.0005$.

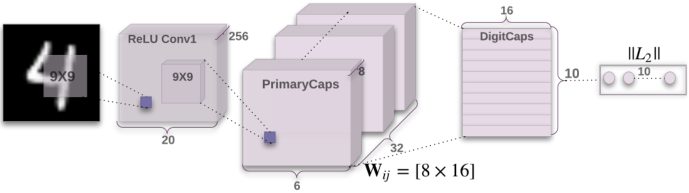
Net loss



Loss* = *margin loss* + α *reconstruction loss

The reconstruction loss is the squared difference between the reconstructed image and the input image. In the paper, $\alpha = 0.0005$.

MNIST example



- ReLU Conv1
  - 9×9 kernels, stride 1, no padding, $\rightarrow 28 - 9 + 1 = 20$
  - 256 channels
- PrimaryCaps
  - 9×9 kernels, stride 2, no padding $\rightarrow \lfloor \frac{20-9}{2} \rfloor + 1 = 6$
  - 8 neurons per capsule
  - $32 \times 6 \times 6$ capsules
- DigiCaps
  - 16 neurons per capsule
  - 10 capsules (classes)

Hinton's second capsule paper

- It may not be an efficient representation to have both pose and probability embedded in one vector
- Hinton *et al.* suggest to split this representation into a scalar value (with the probability of activation) and a pose matrix in their followup paper

Routing by agreement with EM

- Instead of computing agreement through similarity (inner product) between capsules across two layers, compute a dedicated pose information for each capsule of lower layer
  - Use 4×4 matrix M_i for pose of capsule i
  - The pose to the next layer is adjusted by multiplying a transform matrix. The effective pose of capsule i (lower layer) in capsule j's perspective (next layer) is $v_{i,j} = W_{i,j} M_i$
- For each pose h, the agreement of the pose can be gauged by

$$cost_{i,j}^h = -\ln p_{j|i}^h,$$

where the probability $p_{j|i}^h$ is approximated as $\mathcal{N}((W_{i,j}M_i)^h; \mu_j^h, \sigma_j^h)$ and μ_j^h and σ_j^h are approximated using EM (explained later)

- The assignment probability of capsule i to capsule j, $r_{i,j}$, will be scaled by the activation of capsule i, a_i. $r_{i,j} \leftarrow a_i r_{i,j}$
- Output of capsule j: $a_j = sigmoid(\lambda(b_j - \sum_h \sum_i r_{i,j} cost_{i,j}^h))$

Routing by agreement with EM

- Instead of computing agreement through similarity (inner product) between capsules across two layers, compute a dedicated pose information for each capsule of lower layer
  - Use 4×4 matrix M_i for pose of capsule i
  - The pose to the next layer is adjusted by multiplying a transform matrix. The effective pose of capsule i (lower layer) in capsule j's perspective (next layer) is $v_{i,j} = W_{i,j} M_i$
- For each pose h, the agreement of the pose can be gauged by

$$cost_{i,j}^h = -\ln p_{j|i}^h,$$

where the probability $p_{j|i}^h$ is approximated as $\mathcal{N}((W_{i,j}M_i)^h; \mu_j^h, \sigma_j^h)$ and μ_j^h and σ_j^h are approximated using EM (explained later)

- The assignment probability of capsule i to capsule j, $r_{i,j}$, will be scaled by the activation of capsule i, a_i. $r_{i,j} \leftarrow a_i r_{i,j}$
- Output of capsule j: $a_j = sigmoid(\lambda(b_j - \sum_h \sum_i r_{i,j} cost_{i,j}^h))$

Routing by agreement with EM

- Instead of computing agreement through similarity (inner product) between capsules across two layers, compute a dedicated pose information for each capsule of lower layer
    - Use 4×4 matrix M_i for pose of capsule i
    - The pose to the next layer is adjusted by multiplying a transform matrix. The effective pose of capsule i (lower layer) in capsule j's perspective (next layer) is $v_{i,j} = W_{i,j} M_i$
- For each pose h, the agreement of the pose can be gauged by

$$cost_{i,j}^h = -\ln p_{j|i}^h,$$

where the probability $p_{j|i}^h$ is approximated as $\mathcal{N}((W_{i,j}M_i)^h; \mu_j^h, \sigma_j^h)$ and μ_j^h and σ_j^h are approximated using EM (explained later)

- The assignment probability of capsule i to capsule j, $r_{i,j}$, will be scaled by the activation of capsule i, a_i. $r_{i,j} \leftarrow a_i r_{i,j}$
- Output of capsule j: $a_j = sigmoid(\lambda(b_j - \sum_h \sum_i r_{i,j} cost_{i,j}^h))$

Routing by agreement with EM

- Instead of computing agreement through similarity (inner product) between capsules across two layers, compute a dedicated pose information for each capsule of lower layer
  - Use 4×4 matrix M_i for pose of capsule i
  - The pose to the next layer is adjusted by multiplying a transform matrix. The effective pose of capsule i (lower layer) in capsule j's perspective (next layer) is $v_{i,j} = W_{i,j} M_i$
- For each pose h, the agreement of the pose can be gauged by

$$cost_{i,j}^h = -\ln p_{j|i}^h,$$

where the probability $p_{j|i}^h$ is approximated as $\mathcal{N}((W_{i,j}M_i)^h; \mu_j^h, \sigma_j^h)$ and μ_j^h and σ_j^h are approximated using EM (explained later)

- The assignment probability of capsule i to capsule j, $r_{i,j}$, will be scaled by the activation of capsule i, a_i. $r_{i,j} \leftarrow a_i r_{i,j}$
- Output of capsule j: $a_j = sigmoid(\lambda(b_j - \sum_h \sum_i r_{i,j} cost_{i,j}^h))$

Routing by agreement with EM

- Instead of computing agreement through similarity (inner product) between capsules across two layers, compute a dedicated pose information for each capsule of lower layer
  - Use 4×4 matrix M_i for pose of capsule i
  - The pose to the next layer is adjusted by multiplying a transform matrix. The effective pose of capsule i (lower layer) in capsule j's perspective (next layer) is $v_{i,j} = W_{i,j} M_i$
- For each pose h, the agreement of the pose can be gauged by

$$cost^h_{i,j} = -\ln p^h_{j|i},$$

  where the probability $p^h_{j|i}$ is approximated as $\mathcal{N}((W_{i,j}M_i)^h; \mu^h_j, \sigma^h_j)$ and μ^h_j and σ^h_j are approximated using EM (explained later)
- The assignment probability of capsule i to capsule j, $r_{i,j}$, will be scaled by the activation of capsule i, a_i. $r_{i,j} \leftarrow a_i r_{i,j}$
- Output of capsule j: $a_j = sigmoid(\lambda(b_j - \sum_h \sum_i r_{i,j} cost^h_{i,j}))$

Routing by agreement with EM

- Instead of computing agreement through similarity (inner product) between capsules across two layers, compute a dedicated pose information for each capsule of lower layer
  - Use 4×4 matrix M_i for pose of capsule i
  - The pose to the next layer is adjusted by multiplying a transform matrix. The effective pose of capsule i (lower layer) in capsule j's perspective (next layer) is $v_{i,j} = W_{i,j} M_i$
- For each pose h, the agreement of the pose can be gauged by

$$cost_{i,j}^h = -\ln p_{j|i}^h,$$

where the probability $p_{j|i}^h$ is approximated as $\mathcal{N}((W_{i,j} M_i)^h; \mu_j^h, \sigma_j^h)$ and μ_j^h and σ_j^h are approximated using EM (explained later)

- The assignment probability of capsule i to capsule j, $r_{i,j}$, will be scaled by the activation of capsule i, a_i. $r_{i,j} \leftarrow a_i r_{i,j}$
- Output of capsule j: $a_j = sigmoid(\lambda(b_j - \sum_h \sum_i r_{i,j} cost_{i,j}^h))$

Simplify cost

Ω_L : set of capsule indices at layer L

$$cost_j^h \triangleq \sum_{i \in \Omega_L} r_{i,j} cost_{i,j}^h$$

Simplify cost

Ω_L : set of capsule indices at layer L

$$cost_j^h \triangleq \sum_{i \in \Omega_L} r_{i,j} cost_{i,j}^h = \sum_{i \in \Omega_L} -r_{i,j} \ln p_{j|i}^h$$

Simplify cost

Ω_L : set of capsule indices at layer L

$$
\begin{aligned}
cost_j^h &\triangleq \sum_{i\in\Omega_L} r_{i,j} cost_{i,j}^h = \sum_{i\in\Omega_L} -r_{i,j} \ln p_{j|i}^h \\
&= \sum_{i\in\Omega_L} -r_{i,j} \ln\left(\frac{1}{\sqrt{2\pi(\sigma_j^h)^2}} \exp\left(-\frac{(v_{i,j}^h-\mu_j^h)^2}{2(\sigma_j^h)^2}\right)\right)
\end{aligned}
$$

Simplify cost

Ω_L : set of capsule indices at layer L

$$
\begin{aligned}
cost_j^h &\triangleq \sum_{i\in\Omega_L} r_{i,j} cost_{i,j}^h = \sum_{i\in\Omega_L} -r_{i,j} \ln p_{j|i}^h \\
&= \sum_{i\in\Omega_L} -r_{i,j} \ln\left(\frac{1}{\sqrt{2\pi(\sigma_j^h)^2}} \exp\left(-\frac{(v_{i,j}^h-\mu_j^h)^2}{2(\sigma_j^h)^2}\right)\right) \\
&= \sum_{i\in\Omega_L} r_{i,j}\left[\ln\sqrt{2\pi} + \ln\sigma_j^h + \frac{(v_{i,j}^h-\mu_j^h)^2}{2(\sigma_j^h)^2}\right]
\end{aligned}
$$

Simplify cost

Ω_L : set of capsule indices at layer L

$$
\begin{aligned}
cost_j^h &\triangleq \sum_{i\in\Omega_L} r_{i,j} cost_{i,j}^h = \sum_{i\in\Omega_L} -r_{i,j}\ln p_{j|i}^h \\
&= \sum_{i\in\Omega_L} -r_{i,j}\ln\left(\frac{1}{\sqrt{2\pi(\sigma_j^h)^2}}\exp\left(-\frac{(v_{i,j}^h-\mu_j^h)^2}{2(\sigma_j^h)^2}\right)\right) \\
&= \sum_{i\in\Omega_L} r_{i,j}\left[\ln\sqrt{2\pi}+\ln\sigma_j^h+\frac{(v_{i,j}^h-\mu_j^h)^2}{2(\sigma_j^h)^2}\right]
\end{aligned}
$$

Note that in EM update $(\sigma_j^h)^2 \leftarrow \dfrac{\sum_i r_{i,j}(v_{i,j}^h-\mu_j^h)^2}{\sum_i r_{i,j}}$

Simplify cost

Ω_L : set of capsule indices at layer L

$$
\begin{aligned}
cost_j^h &\triangleq \sum_{i\in\Omega_L} r_{i,j} cost_{i,j}^h = \sum_{i\in\Omega_L} -r_{i,j}\ln p_{j|i}^h \\
&= \sum_{i\in\Omega_L} -r_{i,j}\ln\left(\frac{1}{\sqrt{2\pi(\sigma_j^h)^2}}\exp\left(-\frac{(v_{i,j}^h-\mu_j^h)^2}{2(\sigma_j^h)^2}\right)\right) \\
&= \sum_{i\in\Omega_L} r_{i,j}\left[\ln\sqrt{2\pi}+\ln\sigma_j^h+\frac{(v_{i,j}^h-\mu_j^h)^2}{2(\sigma_j^h)^2}\right]
\end{aligned}
$$

Note that in EM update $(\sigma_j^h)^2 \leftarrow \dfrac{\sum_i r_{i,j}(v_{i,j}^h-\mu_j^h)^2}{\sum_i r_{i,j}}$

$$
= \sum_{i\in\Omega_L} r_{i,j}\left[\underbrace{\ln\sqrt{2\pi}+\frac{1}{2}}_{k_j}+\ln\sigma_j^h\right]
$$

k_j is obtained through training in Hinton's paper

EM Update

M-step: Estimating statistics (means and variances)

$$\forall i \in \Omega_L, \forall j \in \Omega_{L+1} : r_{i,j} \leftarrow a_i r_{i,j}$$

E-step: Estimating assigning probabilities

EM Update

M-step: Estimating statistics (means and variances)

$$\forall i \in \Omega_L, \forall j \in \Omega_{L+1} : r_{i,j} \leftarrow a_i r_{i,j}$$

$$\forall h, \forall j \in \Omega_{L+1} : \mu_j^h \leftarrow \frac{\sum_{i \in \Omega_L} r_{i,j} v_{i,j}^h}{\sum_{i \in \Omega_L} r_{i,j}}$$

E-step: Estimating assigning probabilities

EM Update

M-step: Estimating statistics (means and variances)

$$\forall i \in \Omega_L, \forall j \in \Omega_{L+1} : r_{i,j} \leftarrow a_i r_{i,j}$$

$$\forall h, \forall j \in \Omega_{L+1} : \mu_j^h \leftarrow \frac{\sum_{i \in \Omega_L} r_{i,j} v_{i,j}^h}{\sum_{i \in \Omega_L} r_{i,j}}$$

$$\forall h, \forall j \in \Omega_{L+1} : (\sigma_j^h)^2 \leftarrow \frac{\sum_{i \in \Omega_L} r_{i,j} (v_{i,j}^h - \mu_j)^2}{\sum_{i \in \Omega_L} r_{i,j}}$$

E-step: Estimating assigning probabilities

EM Update

M-step: Estimating statistics (means and variances)

$$\forall i \in \Omega_L, \forall j \in \Omega_{L+1} : r_{i,j} \leftarrow a_i r_{i,j}$$

$$\forall h, \forall j \in \Omega_{L+1} : \mu_j^h \leftarrow \frac{\sum_{i \in \Omega_L} r_{i,j} v_{i,j}^h}{\sum_{i \in \Omega_L} r_{i,j}}$$

$$\forall h, \forall j \in \Omega_{L+1} : (\sigma_j^h)^2 \leftarrow \frac{\sum_{i \in \Omega_L} r_{i,j} (v_{i,j}^h - \mu_j)^2}{\sum_{i \in \Omega_L} r_{i,j}}$$

$$\forall h, \forall j \in \Omega_{L+1} : cost_j^h \leftarrow (k_j + \ln \sigma_j^h) \sum_{i \in \Omega_L} r_{i,j}$$

E-step: Estimating assigning probabilities

EM Update

M-step: Estimating statistics (means and variances)

$$\forall i \in \Omega_L, \forall j \in \Omega_{L+1} : r_{i,j} \leftarrow a_i r_{i,j}$$

$$\forall h, \forall j \in \Omega_{L+1} : \mu_j^h \leftarrow \frac{\sum_{i \in \Omega_L} r_{i,j} v_{i,j}^h}{\sum_{i \in \Omega_L} r_{i,j}}$$

$$\forall h, \forall j \in \Omega_{L+1} : (\sigma_j^h)^2 \leftarrow \frac{\sum_{i \in \Omega_L} r_{i,j} (v_{i,j}^h - \mu_j)^2}{\sum_{i \in \Omega_L} r_{i,j}}$$

$$\forall h, \forall j \in \Omega_{L+1} : cost_j^h \leftarrow (k_j + \ln \sigma_j^h) \sum_{i \in \Omega_L} r_{i,j}$$

$$\forall j \in \Omega_{L+1} : a_j \leftarrow sigmoid(\lambda(b_j - \sum_h cost_j^h))$$

E-step: Estimating assigning probabilities

EM Update

M-step: Estimating statistics (means and variances)

$$\forall i \in \Omega_L, \forall j \in \Omega_{L+1} : r_{i,j} \leftarrow a_i r_{i,j}$$

$$\forall h, \forall j \in \Omega_{L+1} : \mu_j^h \leftarrow \frac{\sum_{i\in\Omega_L} r_{i,j} v_{i,j}^h}{\sum_{i\in\Omega_L} r_{i,j}}$$

$$\forall h, \forall j \in \Omega_{L+1} : (\sigma_j^h)^2 \leftarrow \frac{\sum_{i\in\Omega_L} r_{i,j} (v_{i,j}^h - \mu_j)^2}{\sum_{i\in\Omega_L} r_{i,j}}$$

$$\forall h, \forall j \in \Omega_{L+1} : cost_j^h \leftarrow (k_j + \ln \sigma_j^h) \sum_{i\in\Omega_L} r_{i,j}$$

$$\forall j \in \Omega_{L+1} : a_j \leftarrow sigmoid(\lambda(b_j - \sum_h cost_j^h))$$

E-step: Estimating assigning probabilities

EM Update

M-step: Estimating statistics (means and variances)

$$\forall i \in \Omega_L, \forall j \in \Omega_{L+1} : r_{i,j} \leftarrow a_i r_{i,j}$$

$$\forall h, \forall j \in \Omega_{L+1} : \mu_j^h \leftarrow \frac{\sum_{i \in \Omega_L} r_{i,j} v_{i,j}^h}{\sum_{i \in \Omega_L} r_{i,j}}$$

$$\forall h, \forall j \in \Omega_{L+1} : (\sigma_j^h)^2 \leftarrow \frac{\sum_{i \in \Omega_L} r_{i,j} (v_{i,j}^h - \mu_j)^2}{\sum_{i \in \Omega_L} r_{i,j}}$$

$$\forall h, \forall j \in \Omega_{L+1} : cost_j^h \leftarrow (k_j + \ln \sigma_j^h) \sum_{i \in \Omega_L} r_{i,j}$$

$$\forall j \in \Omega_{L+1} : a_j \leftarrow sigmoid(\lambda(b_j - \sum_h cost_j^h))$$

E-step: Estimating assigning probabilities

$$\forall i \in \Omega_L, \forall j \in \Omega_{L+1} : p_{i,j} \leftarrow \prod_h \mathcal{N}(v_{i,j}^h; \mu_j^h, \sigma_j^h)$$

EM Update

M-step: Estimating statistics (means and variances)

$$\forall i \in \Omega_L, \forall j \in \Omega_{L+1} : r_{i,j} \leftarrow a_i r_{i,j}$$

$$\forall h, \forall j \in \Omega_{L+1} : \mu_j^h \leftarrow \frac{\sum_{i \in \Omega_L} r_{i,j} v_{i,j}^h}{\sum_{i \in \Omega_L} r_{i,j}}$$

$$\forall h, \forall j \in \Omega_{L+1} : (\sigma_j^h)^2 \leftarrow \frac{\sum_{i \in \Omega_L} r_{i,j} (v_{i,j}^h - \mu_j)^2}{\sum_{i \in \Omega_L} r_{i,j}}$$

$$\forall h, \forall j \in \Omega_{L+1} : cost_j^h \leftarrow (k_j + \ln \sigma_j^h) \sum_{i \in \Omega_L} r_{i,j}$$

$$\forall j \in \Omega_{L+1} : a_j \leftarrow sigmoid(\lambda(b_j - \sum_h cost_j^h))$$

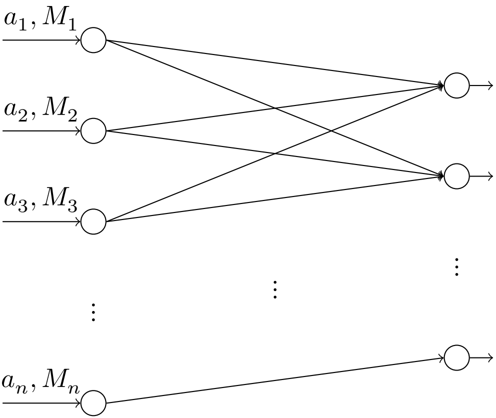
E-step: Estimating assigning probabilities

$$\forall i \in \Omega_L, \forall j \in \Omega_{L+1} : p_{i,j} \leftarrow \prod_h \mathcal{N}(v_{i,j}^h; \mu_j^h, \sigma_j^h)$$

$$\forall i \in \Omega_L, \forall j \in \Omega_{L+1} : r_{i,j} \leftarrow \frac{a_j p_{i,j}}{\sum_{j' \in \Omega_{L+1}} a_{j'} p_{i,j'}}$$

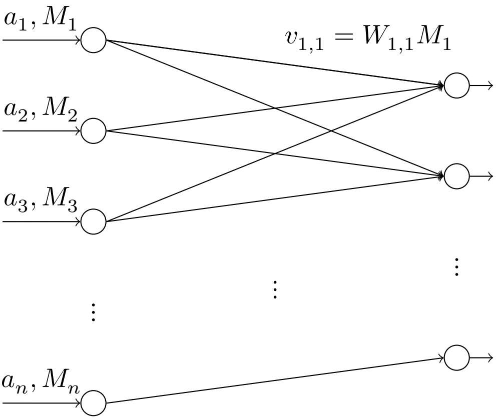
Capsule update with EM

Initialize: $r_{i,j} \leftarrow 1/|\mathcal{N}_i|$



Capsule update with EM

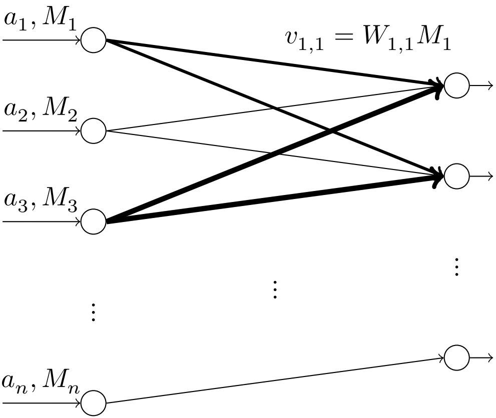
Initialize: $r_{i,j} \leftarrow 1/|\mathcal{N}_i|$



Capsule update with EM

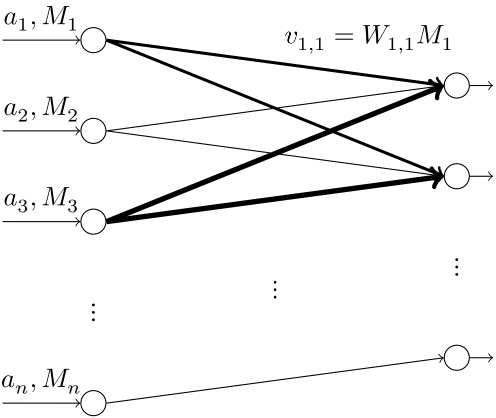
M-step:

$$r_{i,j} \leftarrow a_i r_{i,j}$$



Capsule update with EM

M-step:



$r_{i,j} \leftarrow a_i r_{i,j}$

Estimate μ_j^h, σ_j^h

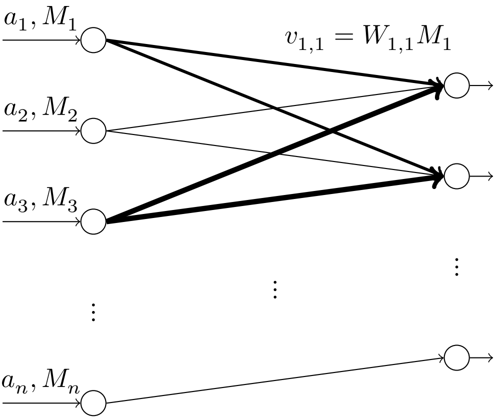
Capsule update with EM

M-step: $cost_j^h = -\sum_i r_{i,j} \ln p_{j|i}^h = -\sum_i r_{i,j} \ln \mathcal{N}(v_{i,j}^h; \mu_j^h, \sigma_j^h)$

$r_{i,j} \leftarrow a_i r_{i,j}$

Estimate μ_j^h, σ_j^h

$cost_j^h \leftarrow (k_j + \ln \sigma_j^h) \sum_i r_{i,j}$



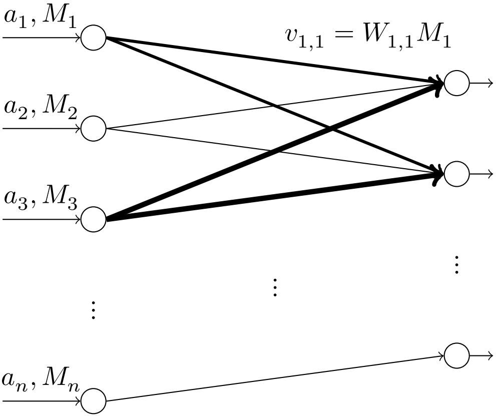
Capsule update with EM

M-step: $$cost_j^h = -\sum_i r_{i,j} \ln p_{j|i}^h = -\sum_i r_{i,j} \ln \mathcal{N}(v_{i,j}^h; \mu_j^h, \sigma_j^h)$$

$$r_{i,j} \leftarrow a_i r_{i,j}$$

Estimate μ_j^h, σ_j^h

$$cost_j^h \leftarrow (k_j + \ln \sigma_j^h) \sum_i r_{i,j}$$



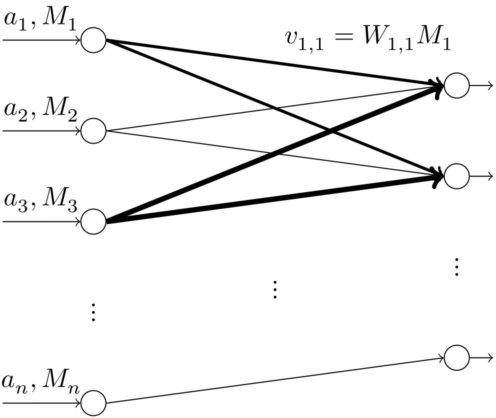
$$a_1 \leftarrow \text{sigmoid}\left(\lambda(b_1 - \sum_h cost_1^h)\right)$$

$$a_2 \leftarrow \text{sigmoid}\left(\lambda(b_2 - \sum_h cost_2^h)\right)$$

$$a_m \leftarrow \text{sigmoid}\left(\lambda(b_m - \sum_h cost_m^h)\right)$$

Capsule update with EM

E-step: $p_{i,j} \leftarrow \prod_h \mathcal{N}(v_{i,j}^h; \mu_j^h, \sigma_j^h)$

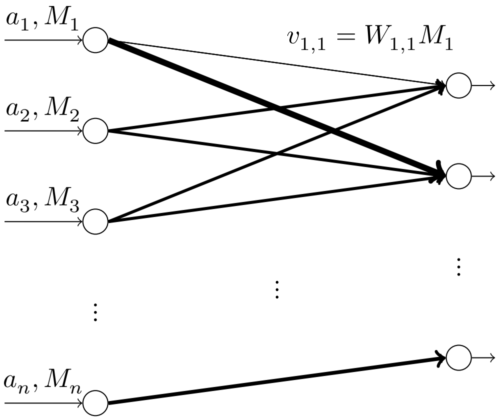


Capsule update with EM

E-step:

$$p_{i,j} \leftarrow \prod_h \mathcal{N}(v_{i,j}^h; \mu_j^h, \sigma_j^h)$$

$$r_{i,j} \leftarrow \frac{a_j p_{i,j}}{\sum_j a_j p_{i,j}}$$



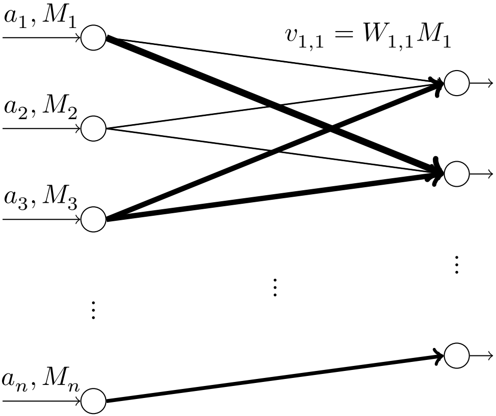
Capsule update with EM

M-step:

$$r_{i,j} \leftarrow a_i r_{i,j}$$

Estimate μ_j^h, σ_j^h

$$cost_j^h \leftarrow (k_j + \ln \sigma_j^h) \sum_i r_{i,j}$$



$$a_1 \leftarrow \text{sigmoid}\left(\lambda(b_1 - \sum_h cost_1^h)\right)$$

$$a_2 \leftarrow \text{sigmoid}\left(\lambda(b_2 - \sum_h cost_2^h)\right)$$

$$\vdots$$

$$a_m \leftarrow \text{sigmoid}\left(\lambda(b_m - \sum_h cost_m^h)\right)$$

Temperature and λ

- In $a_j \leftarrow sigmoid(\lambda(b_j - \sum_h cost_j^h))$, λ is set to the inverse of a "temperature" parameter
- It is similar to simulated annealing that temperature decreases as we get better approximate of the assigning probability $r_{i,j}$
- The paper does not specify the detail control of λ
  - A blog by J Hui tried setting λ to 1 initially and then incrementing it by 1 after each routing iteration. The result seems to work fine

Spread loss

- To make training less sensitive to initialization and hyper-parameter, the authors used spread loss, which is given by

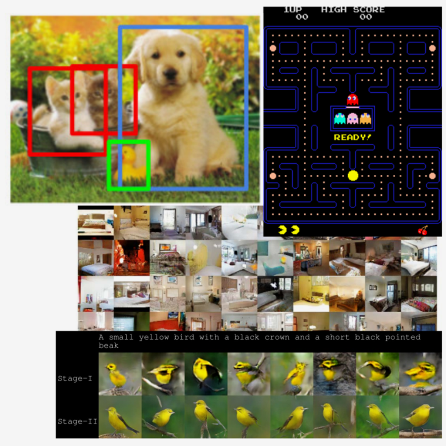
$$L = \sum_{i \neq t} \max(0, m - (a_t - a_i))^2$$

- m is a margin of error, which starts with 0.2 and increase by 0.1 after each epoch of training. It stops at the maximum of 0.9

Reference

- Understanding dynamic routing between capsules
- Understanding matrix capsule with EM routing

We went through a lot...



- Backprop
- Regularization, weight initialization
- CNN
  - R-CNN, faster R-CNN
  - deep dream
- RNN
- Generative models
  - GANs
  - Variational autoencoders
  - Boltzmann machine
- Neural Turing machines
- Deep Q-learning

Many ideas were not new...

- Mainly two things happened
  - Inexpensive computational power
    - GPUs
    - TPUs...
  - Large dataset available
    - ImageNet
    - MS COCO
    - Kaggle...
- And persistent efforts of many AI researchers
  - Hinton (Toronto, Google)
  - Yann Lecun (NYU, Facebook)
  - Bengio (Montreal)

Many ideas were not new...

- Mainly two things happened
  - Inexpensive computational power
    - GPUs
    - TPUs...
  - Large dataset available
    - ImageNet
    - MS COCO
    - Kaggle...
- And persistent efforts of many AI researchers
  - Hinton (Toronto, Google)
  - Yann Lecun (NYU, Facebook)
  - Bengio (Montreal)

Many ideas were not new...

- Mainly two things happened
  - Inexpensive computational power
    - GPUs
    - TPUs...
  - Large dataset available
    - ImageNet
    - MS COCO
    - Kaggle...
- And persistent efforts of many AI researchers
  - Hinton (Toronto, Google)
  - Yann Lecun (NYU, Facebook)
  - Bengio (Montreal)

Many ideas were not new...

IMAGENET

- Mainly two things happened
  - Inexpensive computational power
    - GPUs
    - TPUs...
  - Large dataset available
    - ImageNet
    - MS COCO
    - Kaggle...
- And persistent efforts of many AI researchers
  - Hinton (Toronto, Google)
  - Yann Lecun (NYU, Facebook)
  - Bengio (Montreal)

Four missing pieces of AI (by Lecun)

- Theoretical Understanding for Deep Learning
  - What is the geometry of the objective function in deep networks?
  - Why the ConvNet architecture works so well? [Mallat, Bruna, Tygert...]
- Integrating Representation/Deep Learning with Reasoning, Attention, Planning and Memory
  - A lot of recent work on reasoning/planning, attention, memory, learning “algorithms”
  - Memory-augmented neural nets
  - “Differentiable” algorithm
- Integrating supervised, unsupervised and reinforcement learning into a single “algorithm”
  - Boltzmann machines would be nice if they worked
  - Stacked What-Where Auto-Encoders, Ladder Networks...
- Effective ways to do unsupervised learning
  - Discovering the structure and regularities of the world by observing it and living in it like animals and human do

Information Theory and "Probabilistic Programming"

A shameless advertisement of my fall course

- Will look into (shallow) machine learning models not discussed in this class
  - SVM
  - Decision trees
  - Graphical models...
- Why relevant?
  - They are still very useful when you do not have enough data and do not need to have state-of-the-art accuracy
  - New ideas almost never came from scratch. They all are just some modification of old ideas
    - Standing on the shoulders of giants!

Information Theory and "Probabilistic Programming"

A shameless advertisement of my fall course

- Will look into (shallow) machine learning models not discussed in this class
  - SVM
  - Decision trees
  - Graphical models...
- Why relevant?
  - They are still very useful when you do not have enough data and do not need to have state-of-the-art accuracy
  - New ideas almost never came from scratch. They all are just some modification of old ideas
    - Standing on the shoulders of giants!

Information Theory and "Probabilistic Programming"

A shameless advertisement of my fall course

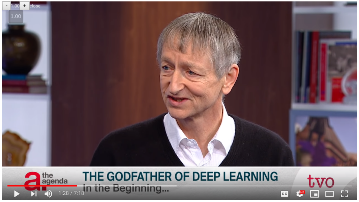
- Will look into (shallow) machine learning models not discussed in this class
    - SVM
    - Decision trees
    - Graphical models...
- Why relevant?
    - They are still very useful when you do not have enough data and do not need to have state-of-the-art accuracy
    - New ideas almost never came from scratch. They all are just some modification of old ideas
        - Standing on the shoulders of giants!

Information Theory and "Probabilistic Programming"

A shameless advertisement of my fall course

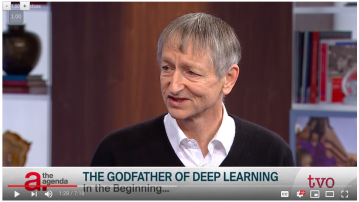
- Will look into (shallow) machine learning models not discussed in this class
  - SVM
  - Decision trees
  - Graphical models...
- Why relevant?
  - They are still very useful when you do not have enough data and do not need to have state-of-the-art accuracy
  - New ideas almost never came from scratch. They all are just some modification of old ideas
    - Standing on the shoulders of giants!

Final words



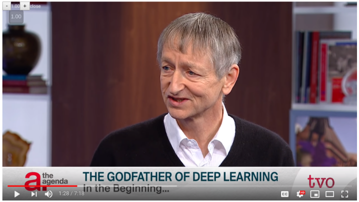
- 'When I was doing my Ph.D., my advisor would tell me that (I was wasting my time) every week. And I would say, "give me six months and I will prove you that it works." And every six months, I'd say that again'
- Don't easily believe something wouldn't work just because someone told you so
  - Try it yourself!
- If you really believe in it, be persistent and enjoy your last laugh

Final words



- 'When I was doing my Ph.D., my advisor would tell me that (I was wasting my time) every week. And I would say, "give me six months and I will prove you that it works." And every six months, I'd say that again'
- Don't easily believe something wouldn't work just because someone told you so
  - Try it yourself!
- If you really believe in it, be persistent and enjoy your last laugh

Final words



- 'When I was doing my Ph.D., my advisor would tell me that (I was wasting my time) every week. And I would say, "give me six months and I will prove you that it works." And every six months, I'd say that again'
- Don't easily believe something wouldn't work just because someone told you so
    - Try it yourself!
- If you really believe in it, be persistent and enjoy your last laugh

Wish you all good luck with your finals and presentations!

Don't forget project submission!

And have a fruitful sem-break!

Please fill in evaluation!