Module 4F12: Computer Vision and Robotics

Examples Paper 4: Solutions

Straightforward questions are marked †
Tripos standard (but not necessarily Tripos length) questions are marked *

1. † Entropy and information

“Compute the information gained about which is the odd ball.”

let \( y \) represent the possible weighing outcomes (left pan heavier or right pan heavier) and \( x \) represent one of the 12 hypotheses about the identity of the odd ball. \( H(Y) = 1 \) bit and \( H(Y|X) = 1 \) bit so the mutual information is zero.

“Compute the information gained about which is the odd ball and whether it is heavy or light.”

let \( x \) now denote one of the 24 hypotheses (each ball may be heavier or lighter). \( H(Y) = 1 \) bit as before and \( H(Y|X) = 0 \) bits since knowing \( x \) renders the weighing outcome deterministic. The mutual information is 1 bit which accords with the intuition that half of the hypotheses have been ruled out by the weighing outcome.

2. * Decision Trees

“Compute the entropy of the class values, \( H(X) \), in each case. Explain your results.”

For all cases \( p_i(x = 1) = \frac{1}{2} \). (Since the class label distribution is independent from the query, \( p_i(x) \) does not vary.) The entropy of the class labels is therefore \( H(X) = 1 \) bit.

“Compute the entropy of the query values, \( H(Y) \), in each case. Comment on whether this entropy is a sensible criterion for selecting the best query.”

\[
p(y_1 = 1) = \frac{1}{2}. \text{ So } H(Y_1) = 1 \text{ bit.}
\]

\[
p(y_2 = 1) = \frac{1}{8}. \text{ Therefore } H(Y_2) = \frac{8}{8} \log_2 \frac{8}{7} + \frac{1}{8} \log_2 8 = 0.54 \text{ bits.}
\]

\[
p(y_2 = 1) = \frac{1}{2}. \text{ Therefore } H(Y_3) = 1 \text{ bit.}
\]
“Compute the conditional entropy, \( H(X|Y) \), and argue which query is the most informative.”

\[
\begin{array}{c|cc}
  y_1 & x = 0 & x = 1 \\
  \hline
  y_1 = 0 & 1/2 & 1/2 \\
  y_1 = 1 & 1/2 & 1/2 \\
\end{array}
\quad
\begin{array}{c|cc}
  y_2 & x = 0 & x = 1 \\
  \hline
  y_2 = 0 & 3/7 & 4/7 \\
  y_2 = 1 & 1 & 0 \\
\end{array}
\quad
\begin{array}{c|cc}
  y_3 & x = 0 & x = 1 \\
  \hline
  y_3 = 0 & 3/4 & 1/4 \\
  y_3 = 1 & 1/4 & 3/4 \\
\end{array}
\]

\[
H(Y_1|X) = 4 \times \frac{1}{4} \log_2 2 = 1 \text{ bit. Therefore } I(X;Y_1) = 0 \text{ bits}
\]

\[
H(Y_2|X) = \frac{3}{4} \log_2 \frac{7}{9} + \frac{1}{8} \log_2 1 + \frac{1}{2} \log_2 \frac{7}{4} = 0.86 \text{ bits. Therefore } I(X;Y_2) = 1 - 0.86 = 0.14 \text{ bits}
\]

\[
H(Y_3|X) = 2 \times \frac{3}{4} \log_2 \frac{4}{3} + 2 \times \frac{1}{8} \log_2 4 = 0.81 \text{ bits. Therefore } I(X;Y_3) = 1 - 0.81 = 0.19 \text{ bits}
\]

3. *Neural networks*

“Show that the derivatives of the cost function are :\[
\frac{d}{dw}G(w) = - \sum_n (t^{(n)} - x^{(n)})z^{(n)}
\]"

First note that application of the chain rule yields,

\[
\frac{d}{dw}G(w) = - \sum_n \frac{t^{(n)} - x^{(n)}}{x^{(n)}(1 - x^{(n)})} \frac{dx^{(n)}}{dw_k}.
\]

Then we find,

\[
\frac{dx^{(n)}}{dw_k} = x^{(n)}(1 - x^{(n)})z_k^{(n)}.
\]

Combining the above recovers the desired result.

“Interpreting the output of the network as \( x_i = p(t_i = 1|w, z) = \frac{\exp(w^T_i z)}{\sum_j \exp(w^T_j z)} \) write down a cost-function for training this network based on the log-probability of the training data given the weights \( w \) and inputs \( \{z^{(n)}\}_{n=1}^N \).”

A sensible cost function is given by the log-probability of the class labels given the inputs:

\[
G(w) = - \sum_n \sum_{i=1}^I t_i^{(n)} \log x_i(z^{(n)}; w).
\]

“What is the relationship between this network and the one described in the first part of this question?”
let the number of classes $I = 2$ we note that

$$x_1 = \frac{\exp(w_1^Tz)}{\exp(w_1^Tz) + \exp(w_2^Tz)} = \frac{1}{1 + \exp(-(w_1 - w_2)^Tz)} = 1 - x_2$$

which is identical to the above when $w = w_1 - w_2$

now check that the cost function we proposed is identical too

$$G(w) = -\sum_n \left[ t_1^{(n)} \log x_1(z^{(n)}; w) + t_2^{(n)} \log x_2(z^{(n)}; w) \right]$$

$$= -\sum_n \left[ t_1^{(n)} \log x_1(z^{(n)}; w) + (1 - t_1^{(n)}) \log(1 - x_1(z^{(n)}; w)) \right]$$

4. *Convolutional neural networks*

“Show that the derivatives of the objective function with respect to the convolutional weights, $w_k$, can themselves be computed efficiently using convolutions.”

First apply the chain and product rules:

$$\frac{d}{dw_k}G(w) = \sum_n \frac{dG(w)}{dx^{(n)}} \sum_j \frac{dx^{(n)}}{dy_j^{(n)}} \frac{dy_j^{(n)}}{da_j^{(n)}} \frac{da_j^{(n)}}{dw_k}$$

then compute the component parts:

$$\frac{dG(w)}{dx^{(n)}} = -\frac{t^{(n)} - x^{(n)}}{x^{(n)}(1 - x^{(n)})}, \quad \frac{dx^{(n)}}{da_j^{(n)}} = x^{(n)}(1 - x^{(n)})v_j.$$  

$$\frac{dy_j^{(n)}}{da_j^{(n)}} = \frac{df(a_j^{(n)})}{da_j^{(n)}} = f'(a_j^{(n)}), \quad \frac{da_j^{(n)}}{dw_k} = z_{j-k}^{(n)}$$

finally combine everything together

$$\frac{dG(w)}{dx^{(n)}} = -\sum_n (t^{(n)} - x^{(n)}) \sum_j v_j f'(a_j^{(n)}) z_{j-k}^{(n)}$$

which involves convolutions between $v_j f'(a_j^{(n)})$ and a reversed version of $z_j^{(n)}$