EX 1 Show that $\frac{\partial C}{\partial q_i} = -\frac{p_i}{q_i}$

a)

$$\frac{\partial C}{\partial q_i} = \frac{\partial}{\partial q_i} \sum_j p_j \log\left(\frac{p_j}{q_j}\right) = \frac{\partial}{\partial q_i} p_i \log\left(\frac{p_i}{q_i}\right) =$$

$= [$ from properties of derivative of a logarithm:

$$\frac{\partial}{\partial x} \ln(x^n) = n\left(\frac{1}{x}\right)$$

from the properties of logarithm: $\ln(xy) = \ln(x) + \ln(y)$ $]$

$$= \frac{\partial}{\partial q_i} p_i \left[\log(p_i) + \log(q_i^{-1})\right] = \frac{\partial}{\partial q_i} p_i \log(q_i^{-1}) =$$

$$= p_i \cdot (-1) \cdot \frac{1}{q_i} = -\frac{p_i}{q_i}$$

b) Show that $\frac{\partial C}{\partial x_i} = -p_i + q_i$, we know $q_i = \frac{\exp(x_i)}{\sum_k \exp(x_k)}$

$$\frac{\partial C}{\partial x_i} = \frac{\partial}{\partial x_i} \sum_j p_j \left[\log(p_j) - \log(q_j)\right]$$

$$= \frac{\partial}{\partial x_i} \sum_j p_j \left(-\log(q_j)\right) = -\frac{\partial}{\partial x_i} \sum_j p_j \log\left(\frac{\exp(x_j)}{\sum_k \exp(x_k)}\right) =$$

$$= -\frac{\partial}{\partial x_i} \sum_j p_j \left[\log(\exp(x_j)) - \log(\textstyle\sum_k \exp(x_k))\right] =$$

$$= -\frac{\partial}{\partial x_i} \sum_j p_j \left[x_j - \log(\textstyle\sum_k \exp(x_k))\right] =$$

$$= -\frac{\partial}{\partial x_i} \sum_j p_j x_j + \frac{\partial}{\partial x_i} \sum_j p_j \log(\textstyle\sum_k (\exp(x_k)) =$$

knowing "chain rule" of derivatives $(f \circ g)' = (f' \circ g) \cdot g'$

$$= -p_i + \sum_j p_j \cdot \frac{1}{\sum_k (\exp(x_k))} \cdot \frac{\partial \sum_k (\exp(x_k))}{\partial x_i} =$$

$$= -p_i + \underbrace{\textstyle\sum_j p_j}_{=1} \cdot \underbrace{\frac{\exp(x_i)}{\sum_k (\exp(x_k))}}_{=q_i} = -p + q_i \qquad = \exp(x_i)$$

1 c) Explain which of these two gradients is the most appropriate for practical use in a learning algorithm.

1) stability or boundedness of the gradient

2) ability to produce a valid probability distribution

The values of the first gradient $\frac{\partial C}{\partial q_i} = -\frac{p_i}{q_i}$ can be in range $(-\infty; 0\rangle$. For the second gradient $\frac{\partial C}{\partial x_i} = -p_i + q_i$ the range would be $\langle -1; 1\rangle$.

Therefore the 2nd gradient $\frac{\partial C}{\partial x_i}$ is more appropriate for practical use in a learning algorithm, since the values are bounded and can produce a valid prob. distribution (since the magnitude of the gradient doesn't exceed 1, and the sign indicates direction of the gradient).

The problems of the first gradient are:

- the values are not stable (can explode quickly to minus infinity)
- the gradient can't be applied if $q_i = 0$
- it will have difficulties to produce valid prob. distribution (since the absolute values exceed one)

EX 2

t-SNE solves the problem of crowding by using a more heavy-tailed neighborhood distribution (esp. t-Student distrib. with d.o.f. = 1 => Cauchy distrib) in the low-dimensional output space than in the input space. Since then, the neigborhood probability falls off less rapidly i.e. there is less need to push some points far-off and crowd remaining points close together in the center.

In the crowding problem, pairs of points that are only slightly similar have to be modeled too far apart in the map. Since there is a relatively large number of pairs of points that are slightly similar, these points would all like to be closer together in the maps (i.e. they crush the low-dimensional map together).

By using a heavy-tailed distribution to measure similarities in the low-dimensional map, t-SNE allows points that are only slightly similar to be visualized much further appart in the map.

Therefore the idea of „Reverse t-SNE" would lead to the fact that the moderate distances in high-dimensional space would be modeled by shorter distane in low-dimensional space, what would amplify the problem of crowding (assuming that it exists).

Since the extent to which the crowding problem occurs depends on the ratio between the intrinsic data dimensionality and the embedding dimensionality. So, if we embed in e.g. 30 dim., the crowding problem is less severe than when we embed in 2 dim. (Or if we try to embed intrinsically very low-dimensional data e.g. Swiss roll) then we don't need to worry about „crowding". (e.g. Swiss roll data)

Therefore, if we don't experience severe crowding problem „Reverse t-SNE" would help to see the structure / cluster data which intrinsically have bigger in the distribution tails (e.g. financial data) and that will allow to better separate clusters in the low-dim. map.