Algorithms for Machine Learning and Inference Exam

Adam Olsson

June 2019

Question 1

1a)

The concept of MLE is: given a data set, we will select the model parameters that make our observations most likely (i.e maximises the likelihood). We do this by defining a loss function from joint density of the observings and then finding a optimum for the natural logarithm of the loss function. The natural logarithm often makes the loss function convex which creates a single (global) optimum. Maximising the log loss also maximises the likelihood.

The concept of Bayesian Estimation is to infere the posterior distribution from the prior, likelihood and marginal distribution. In simpler terms: if we know the distribution of our observings, can we find the parameters of our model. The reason is that our model parameters can be considered as continious random variables and can have multiple seemingly good values. So what we do is that we sample from the model parameter distribution and find the expected values.

1b)

Assuming i.i.d:

$$\mathrm{L} = p(\mathbf{t}|\mathbf{X}, \lambda) = \prod_{n=1}^{N} p(t_n|x_n, \lambda) = \prod_{n=1}^{N} \frac{\lambda^{x_n} e^{-\lambda}}{x_n!} = e^{-N\lambda} \frac{\lambda^{\sum_{n=1}^{N} x_n}}{\sum_{n=1}^{N} x_n!}$$

Note that the numerator is λ raised to the power of the sum. Formatting makes it hard to see.

1c)

Log L = log $(e^{-N\lambda})$ + log $(\frac{\lambda^{\sum_{n=1}^{N} x_n}}{\sum_{n=1}^{N} x_n!})$ =log$(e^{-N\lambda})$ + log$(\lambda^{\sum_{n=1}^{N} x_n})$ - log$(\sum_{n=1}^{N} x_n!)$ = $-N\lambda$ + log$(\lambda) \sum_{n=1}^{N} x_n$ - log$(\sum_{n=1}^{N} x_n!)$

Solving for optimum:

$\frac{\partial}{\partial \lambda}$ Log loss $= -N + \frac{1}{\lambda}\sum_{n=1}^{N} x_n = 0$
$N = \frac{1}{\lambda}\sum_{n=1}^{N} x_n$
$\lambda_{MLE} = \frac{1}{N}\sum_{n=1}^{N} x_n$

1d)

$\frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)}\lambda^{\alpha_0-1}e^{-\beta_0\lambda} * \frac{\lambda^{x_n}e^{-\lambda}}{x_n!} = \frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)}\frac{1}{x_n!}\lambda^{\alpha_0+x-1}e^{-\lambda(\beta_0+1)}$

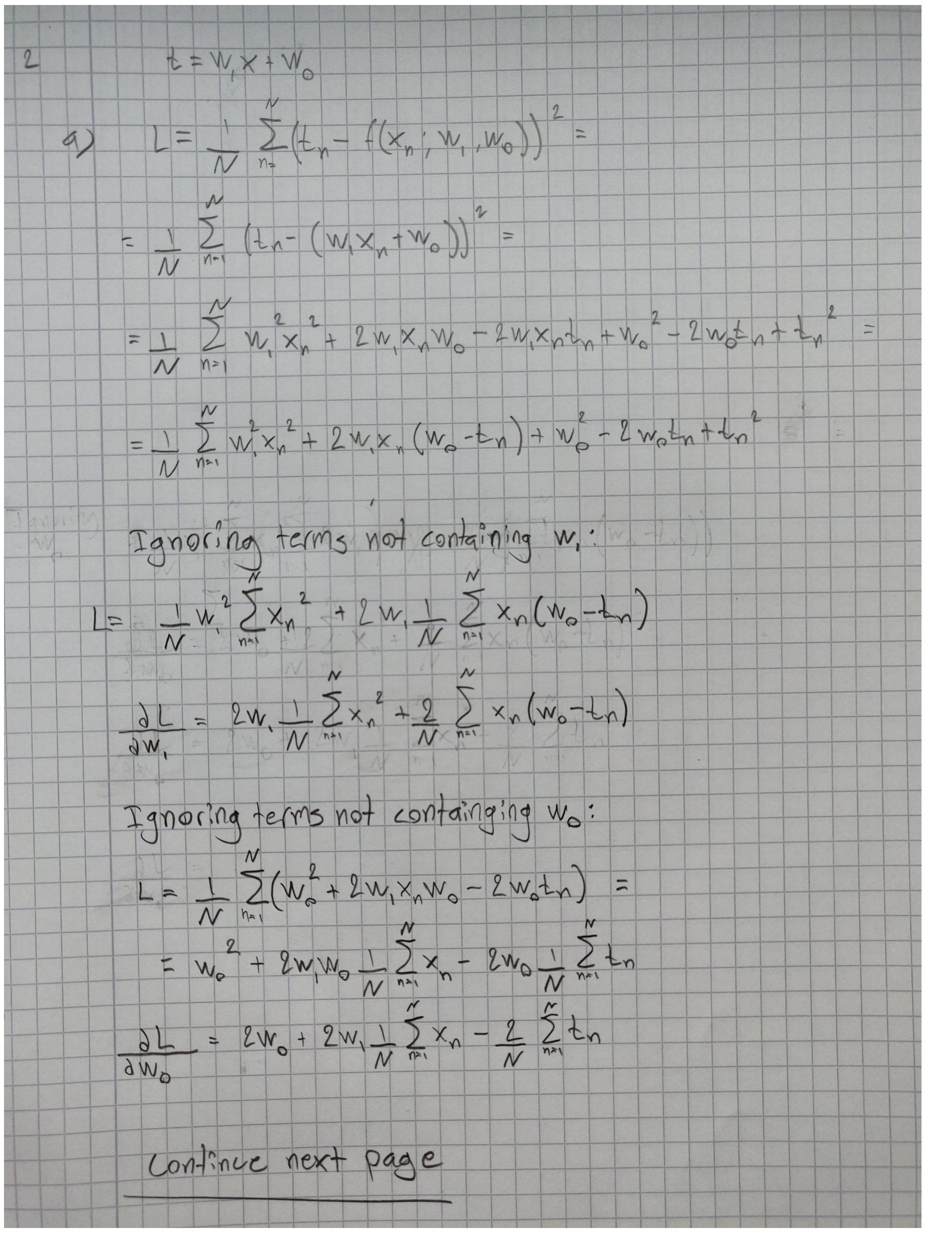
$\alpha = \alpha_0 + x$; $\beta = \beta_0 + 1$

$\frac{\beta^{\alpha}}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda} = \frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)}\lambda^{\alpha_0+x-1}e^{-\lambda(\beta_0+1)}$

Ignoring $\frac{1}{x_n!}$ since it is a constant.

Question 2

2a)



2 $t = w_1 x + w_0$

a) $$L = \frac{1}{N}\sum_{n=}^{N}\left(t_n - f(x_n; w_1, w_0)\right)^2 =$$

$$= \frac{1}{N}\sum_{n=1}^{N}\left(t_n - (w_1 x_n + w_0)\right)^2 =$$

$$= \frac{1}{N}\sum_{n=1}^{N} w_1^2 x_n^2 + 2w_1 x_n w_0 - 2w_1 x_n t_n + w_0^2 - 2w_0 t_n + t_n^2 =$$

$$= \frac{1}{N}\sum_{n=1}^{N} w_1^2 x_n^2 + 2w_1 x_n (w_0 - t_n) + w_0^2 - 2w_0 t_n + t_n^2$$

Ignoring terms not containing w_1:

$$L = \frac{1}{N} w_1^2 \sum_{n=1}^{N} x_n^2 + 2w_1 \frac{1}{N}\sum_{n=1}^{N} x_n (w_0 - t_n)$$

$$\frac{\partial L}{\partial w_1} = 2w_1 \frac{1}{N}\sum_{n=1}^{N} x_n^2 + \frac{2}{N}\sum_{n=1}^{N} x_n (w_0 - t_n)$$

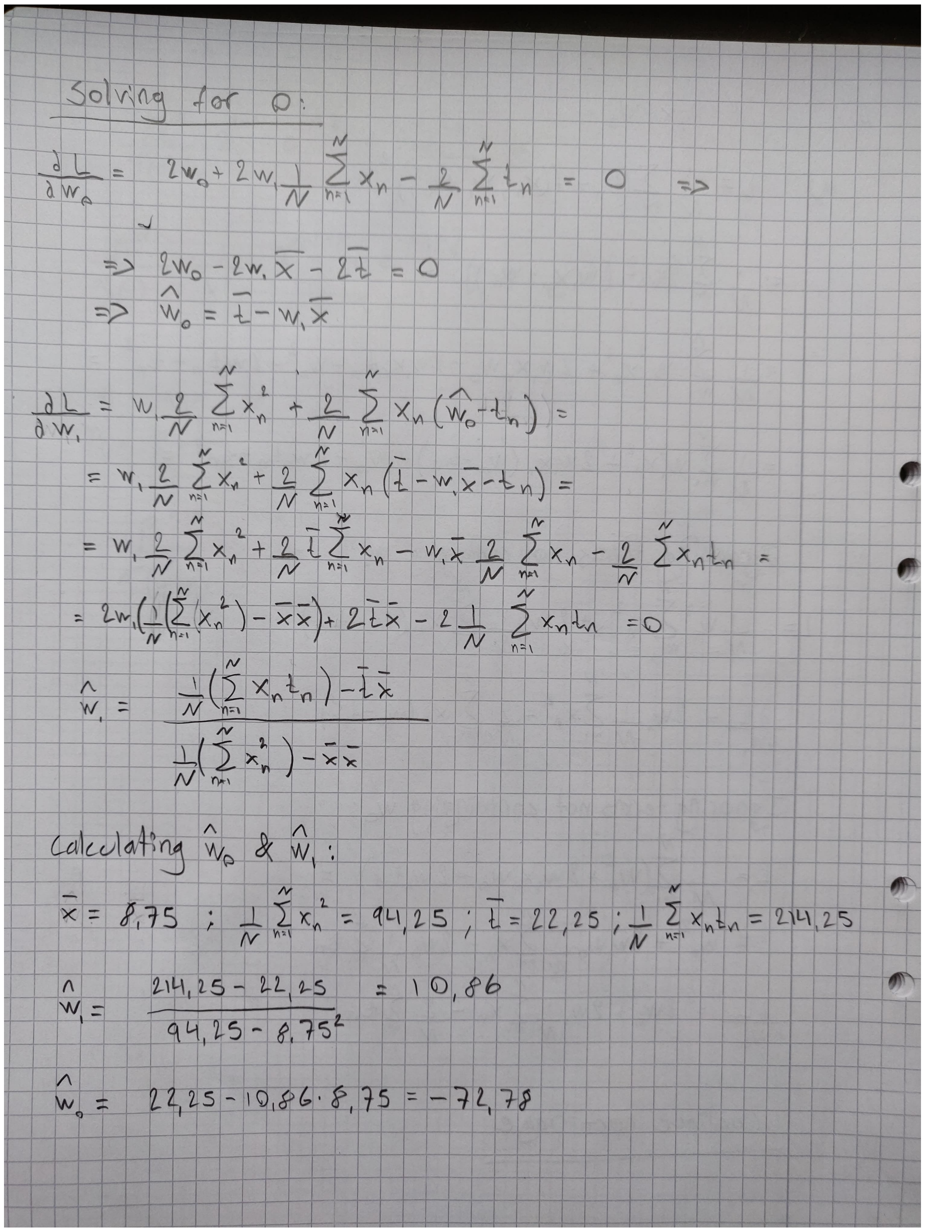
Ignoring terms not containging w_0:

$$L = \frac{1}{N}\sum_{n=1}^{N}\left(w_0^2 + 2w_1 x_n w_0 - 2w_0 t_n\right) =$$

$$= w_0^2 + 2w_1 w_0 \frac{1}{N}\sum_{n=1}^{N} x_n - 2w_0 \frac{1}{N}\sum_{n=1}^{N} t_n$$

$$\frac{\partial L}{\partial w_0} = 2w_0 + 2w_1 \frac{1}{N}\sum_{n=1}^{N} x_n - \frac{2}{N}\sum_{n=1}^{N} t_n$$

Continue next page



Solving for θ:

$$\frac{\partial L}{\partial w_0} = 2w_0 + 2w_1 \frac{1}{N}\sum_{n=1}^{N} x_n - \frac{2}{N}\sum_{n=1}^{N} t_n = 0 \quad \Rightarrow$$

$$\Rightarrow 2w_0 - 2w_1\bar{x} - 2\bar{t} = 0$$

$$\Rightarrow \hat{w}_0 = \bar{t} - w_1\bar{x}$$

$$\frac{\partial L}{\partial w_1} = w_1 \frac{2}{N}\sum_{n=1}^{N} x_n^2 + \frac{2}{N}\sum_{n=1}^{N} x_n(\hat{w}_0 - t_n) =$$

$$= w_1 \frac{2}{N}\sum_{n=1}^{N} x_n^2 + \frac{2}{N}\sum_{n=1}^{N} x_n(\bar{t} - w_1\bar{x} - t_n) =$$

$$= w_1 \frac{2}{N}\sum_{n=1}^{N} x_n^2 + \frac{2}{N}\bar{t}\sum_{n=1}^{N} x_n - w_1\bar{x}\frac{2}{N}\sum_{n=1}^{N} x_n - \frac{2}{N}\sum^{N} x_n t_n =$$

$$= 2w_1\left(\frac{1}{N}\left(\sum_{n=1}^{N} x_n^2\right) - \bar{x}\bar{x}\right) + 2\bar{t}\bar{x} - 2\frac{1}{N}\sum_{n=1}^{N} x_n t_n = 0$$

$$\hat{w}_1 = \frac{\frac{1}{N}\left(\sum_{n=1}^{N} x_n t_n\right) - \bar{t}\bar{x}}{\frac{1}{N}\left(\sum_{n=1}^{N} x_n^2\right) - \bar{x}\bar{x}}$$

Calculating $\hat{w}_0$ & $\hat{w}_1$:

$\bar{x} = 8,75$; $\frac{1}{N}\sum_{n=1}^{N} x_n^2 = 94,25$; $\bar{t} = 22,25$; $\frac{1}{N}\sum_{n=1}^{N} x_n t_n = 214,25$

$$\hat{w}_1 = \frac{214,25 - 22,25}{94,25 - 8,75^2} = 10,86$$

$$\hat{w}_0 = 22,25 - 10,86 \cdot 8,75 = -72,78$$

2b)

i) We only find a point estimate of the parameters which is very dependent on what data we have. Different observings would give other params.

ii) We use past beliefs to incorporate them into what we think will happen in the future. We don't look at one single point estimate of model params but instead sample the parameter space to find the most fitting.

2c)

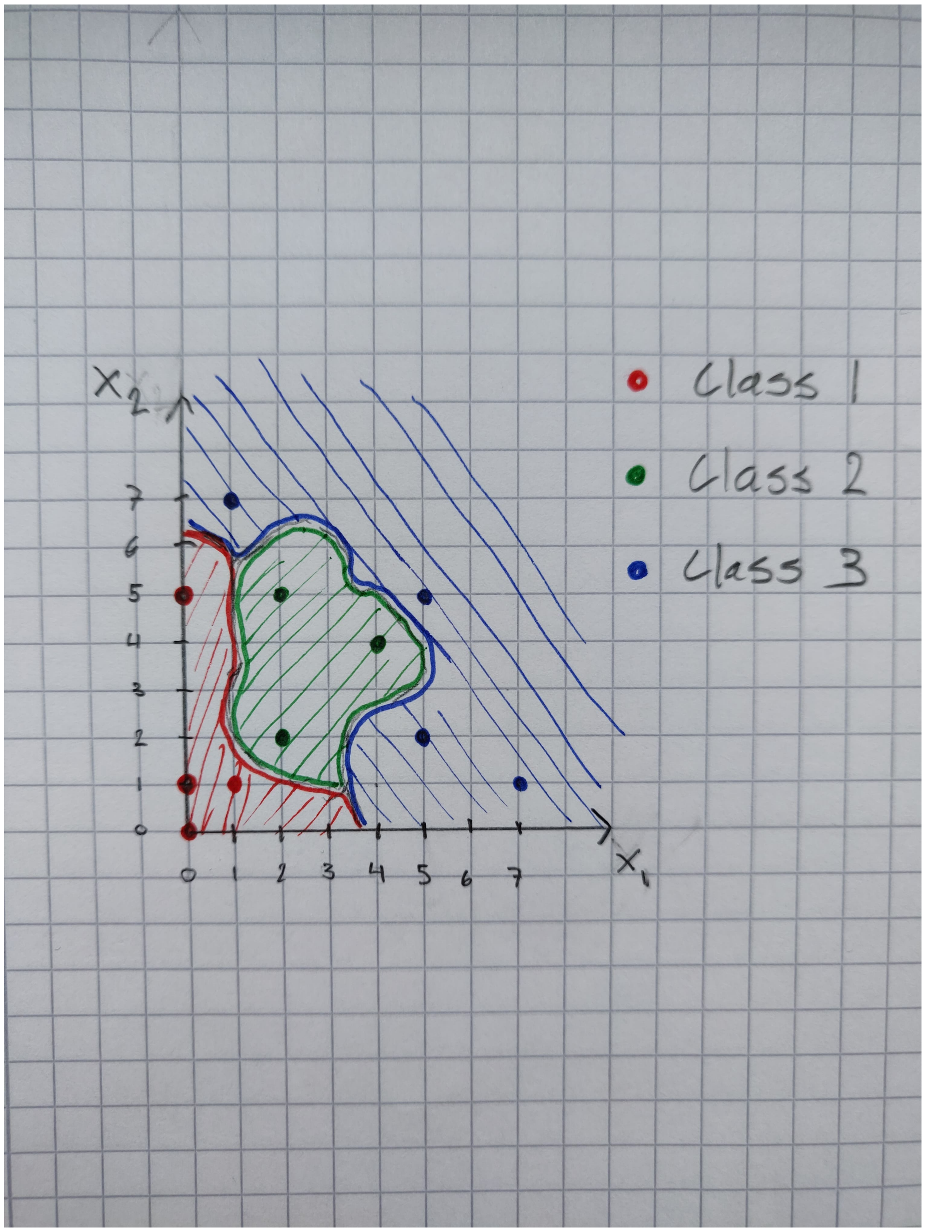
True because we have a more advanced model that can fit the data better but it is also more prone to overfitting.

2d)

We do CV to get an idea how well our model generalize on the training data. Evaluation techniques doesn't evaluate how well our model has generalized on a completely new data set. However, CV gives an indication of this.

Question 3

3a)



3b)

Let the K+1 closest point among the K nearest classes decide the label. It would be reasonable because it would choose the class that has highest density in the area of interest

3c)

Points in class1 that are compared to points from the same class gets a low value (<0.1) and points in class2 that are compared to points from the same class gets a high value (>0.8). However when x1 and x2 are compared to x6 and x5 respectively the resulting values are closer to 0.5 (0.34 and 0.56 to be exact). I do not think this is a suitable kernel because I would have like to have a clearer separation of values between points from different classes. For example: points that are not in the same class gets a negative score. Although it is also possible to argue that if we find the separation threshold the kernel will split the points fine. But overall, i do not find the kernel suitable.

3d)

No answer.

Question 4

4a)

$O = \frac{W-K+2P}{S} + 1 = W - K + 1$ where, W is width or height of input image, K kernel size, P padding and S is stride.

Output width: $w - 2h_f + 2$

Output height: $h - 2h_f + 2$

Output size: (8) x $(h - 2h_f + 2)$ x $(w - 2h_f + 2)$

Learnable parameters: $8(2h_f + 1)^2$

4b)

The weight matrix would be of size: $[8(h - 2h_f + 2)(w - 2h_f + 2)]$ x $[wh]$

It means that the fully connected layer need a lot more data to fit properly since it needs to learn all those parameters.

4c)

Unclear question, I don't understand what I am suppose to take the derivative with respect to so here are both x as an element and x as a vector:

$y_i = \sum_{j=1}^{N} W_{i,j} x_j => y = \mathbf{W}x$ (dot product)

$\frac{\partial E}{\partial x} = \frac{\partial y}{\partial x}\frac{\partial E}{\partial y} = \mathbf{W}\frac{\partial E}{\partial y}$

$\frac{\partial E}{\partial x_j} = \frac{\partial y}{\partial x_j}\frac{\partial E}{\partial y} = W_j \frac{\partial E}{\partial y}$

$\frac{\partial E}{\partial x_j} = \frac{\partial y_i}{\partial x_j}\frac{\partial E}{\partial y_i} = W_{i,j} \frac{\partial E}{\partial y_i}$

4d)

1. Initialize the weight matrix and learning rate
2. Repeat: $w_{i,j}^{t+1} = w_{i,j}^{t} - \gamma \frac{\partial E^t}{\partial w_{i,j}}$ for all (i, j)

Question 5

5a)

GMM assumes that the data is generated from multiple distributions where K-means assumes data is generated from a single distribution.

5b)

1. Select one of the K Gaussians where each distribution has a probability π_K of being chosen. $(\sum_K \pi_K) = 1$
2. Sample one point from the selected Gaussian.
3. Repeat from (1)

5c)

The number of learnable parameters are: $c_K = KD + (K - 1)$

5d)

We repeat calculating the BIC score for multiple Ks and select the K which gives the lowest score.

$BIC_K^* = \text{argmin}_K \; BIC_K$

5e)

True, increasing K will always result in smaller cost because you are decreasing the distance to each cluster center. The extreme case of this would be when K = N, every data point is its own cluster and the cost would thus be 0.

5f)

λ is a regularization parameter and by increasing it we punish more complex models. This happens because we want to minimize the cost function and increasing λ results in a higher cost.