You MUST answer this question.

1. (a) You have a dataset $D = (\mathbf{x}^n, y^n; n = 1, 2, \ldots, N)$, and a conditional machine learning model of the form $P(y^n|\mathbf{x}^n, \mu_1, \mu_2, s_1, s_2)$, where μ_1, μ_2, s_1 and s_2 are all parameters of the model. The parameters μ_1 and μ_2 are *mean* parameters used within the model, and the s_1 and s_2 are *variance* parameters.

   i. Write down the log-likelihood maximisation (for the whole dataset D) in the form of a constrained minimisation problem for the parameters μ_1, μ_2, s_1 and s_2. A constrained minimisation is written in the form

$$\theta^* = \arg\min_{\theta} f(\theta) \text{ subject to } \begin{array}{c} g_1(\theta) > 0 \\ g_2(\theta) > 0 \\ \ldots \\ g_k(\theta) > 0 \end{array} \quad (1)$$

   for some k and $f, g_1, g_2, \ldots, g_k$. [*4 marks*]

   ii. Rewrite the log-likelihood maximisation as an equivalent unconstrained minimisation problem. [*3 marks*]

   iii. Very briefly explain why using line minimisation within a high dimensional optimisation procedure can be beneficial in terms of speed? [*2 marks*]

   (b) Suppose we have a posterior density $P(\theta|D)$ for parameters θ and for data denoted by D.

   i. Describe the Monte-Carlo approach for approximating the posterior mean

$$\int \theta P(\theta|D) d\theta, \quad (2)$$

   where the integral is a definite integral over the whole parameter space. [*3 marks*]

   ii. Describe the process for rejection sampling using a distribution $Q(\theta)$ that we are able to sample from, and where $Q(\theta) > \alpha P(D|\theta)P(\theta)$. [*4 marks*]

   iii. In importance sampling using a proposal distribution $Q(\theta)$ for a target distribution $P(\theta|D)$, what condition do we need on Q for the importance sampling procedure to be valid? [*1 mark*]

   (c) Write out the Bernoulli likelihood for a binary dataset $x^1, x^2, \ldots, x^N$ with Bernoulli probability p. From this, derive the log-likelihood and hence show that the maximum likelihood value for p given data $\mathcal{D}$ corresponds to the proportion of 1s in the dataset. [*5 marks*]

   (d) Naive Bayes assumes conditional independence. By considering the worst case of two attributes being identical given the class label, explain what effect a positive correlation between attributes (given the class) has on the inferred posterior probabilities. [*3 marks*]

Pretty sure 1. a) isn't examinable in 2013?

1. a) i) Has anyone done this one? Alex Yepifanov says constraints are just variance >= 0. Not Equal.

Alex Yepifanov here: Likelihood is product(P(xi|theta))
log likelihood is sum log(P(xi|theta). We usually want to maximise this. Since we want a minimization this is equivalent to

min -log(P(xi|theta))
and the only constraints mentioned in the slide
s is variances > 0 (let me check for equality quickly)l

so final form would be

min -log(P(xi|mu1,mu2,var1,var2))
subject to
var1 > 0
var2 > 0
STRICTLY greater than. Can not equal to 0.

ii. To make unconstrained, set Lolololovariable = log(var1) ce cacat? lolololol variable? lolololo and Yolovariable = log(var2)

min - log(P(xi|mu1,mu2,Lolololovariable,Yolovariable)

I regret nothing. (lol)

I forgot to write in indice limits and such. Also not 100% sure about correctness of answer but it should mostly be there.

iii) We do not need to compute H or inverse of H which is O(d^2) and O(d^3) respectively.
What is H here?

b) i) When the posterior distribution is difficult to integrate analytically, if is often convenient to use Monte-Carlo integration, this methodology involves selecting points at random **from the target distribution** and evaluating the integral at those points. We can estimate the posterior mean by using the formula:

$$E \approx \tilde{E} = \frac{1}{(number\ of\ samples)} * \sum_{i=1}^{number\ of\ samples} \theta_i$$

where θ_i are samples **from the distribution** $P(\theta|D)$.

When the number of samples tends to infinity our approximation of expectation tends to the actual expectation.

ii) In Rejection Sampling we want to sample from an upper bound to the distribution we want. We throw away samples to get the correct shape of the target distribution.

1. Choose a distribution $Q(\theta)$ that we can sample from such that $Q(\theta) > \alpha P(D|\theta)P(\theta)$
2. Sample θ_i from $Q(\theta)$. Sample u from uniform $U(0, 1)$.
3. if $u < P(\theta_i)/wQ(\theta_i)$ then accept sample θ_i and move on to the next i.
4. Otherwise, throw away θ_i and try again.

In the step 3, the $u < P(\theta_i)/wQ(\theta_i)$ maybe not relevant. (I mean he did not focus on the topic). It should be u< aP(D|/theta)P(/theta) / Q(\Theta) ------ Qi Hu
Actually I believe it should be u < P(D | theta)P(theta), since that is the curve we try to observe.

iii) We must have the following: Q(x)=0 when P(x)=0.
Support(Q) superset of Support(P) is the correct answer - Riaz
In summary, a good importance sampling function h(x) should have the following properties:
1. h(x) > 0 whenever g(x) != 0
2. h(x) should be close to being proportional to |g(x)|
3. it should be easy to simulate values from h(x)
4. it should be easy to compute the density h(x) for any value x that you might realize
**http://ib.berkeley.edu/labs/slatkin/eriq/classes/guest_lect/mc_lecture_notes.pdf

c) Bernoulli P(D) = p^x*(1-p)^(n-x) where x is the number of ones and n is the number of datapoints

Then L(P(D)) = x*log(p) + (n-x)*log(1-p)
Set the derivative to 0 to find maximum likelihood solution
d/dp L(P(D)) = x/p - (n-x)/(1-p)
x/p - (n-x)/(1-p) = 0
x/p = (n-x)/(1-p)
x*(1-p) = (n-x)*p
x - x*p = n*p - x*p
x = n*p
p = x/n

d) Anyone? If two attributes are positively correlated given the class label, then the assumption

of conditional independence is least applicable, as each of these attributes will be given equal weight in the model and double the weighting with respect to one of the other attributes. Consider the example of having three attributes for classifying gender where data items are defined as:

2. (a) In standard degree courses, students can get one of a number of final marks (Fail, 3rd, Lower 2nd, Upper 2nd, 1st). You plan to use a neural network to predict the class of degree that a student will get dependent on the marks they obtained on coursework for their courses. Note that different people do different courses, each with different numbers of coursework. You could choose to represent the data by having one input attribute for each piece of coursework, and substituting the mean coursework value (computed across all those who did the course) in cases where individuals did not do that course.

   Alternatively, you could represent the data for an individual by having one input attribute for each range (0% to 10%,11% to 20%, ..., 99% to 100%) and making the input attribute value to be the proportion of the coursework the person did, that was given a mark in that range. For example the inputs for one individual might take the form of a vector $(0\%, 0\%, 0\%, 0\%, 10\%, 40\%, 40\%, 10\%, 0\%, 0\%)^T$.

   i. Give one brief argument against each of the alternative representations. [*6 marks*]

   ii. Suppose that you knew you were going to use a Naive Bayes model. Describe a representation that you could then use that is similar to, but arguably more elegant than, the first alternative above in that it explictly represents missing data, and briefly explain why that representation is suitable for Naive Bayes? [*2 marks*]

   (b) This question relates to neural networks in practice:

   i. Explain why it is important to standardise the data and start with small weights in neural networks. [*3 marks*]

   ii. Why should each of the initial weights for the different units be different from one another? [*2 marks*]

   iii. If we stop training early, what is the effective bias this induces on our learnt networks? [*1 mark*]

   (c) Describe a simple gradient ascent procedure for optimising the weights $\mathbf{w}$ (which includes the biases) of a neural network, given the network log-likelihood, denoted $L(D|\mathbf{w})$ for data D. Describe the problems associated with setting the learning rate. You do not need to say how to compute any derivatives you might need. [*5 marks*]

QUESTION CONTINUES ON NEXT PAGE

x = (height in cm, height in inches, weight in kgs)
In this model, the posterior distribution would be affected twice as much by the variation in height

in comparison to changes in the weight attribute.
Posterior will tend to be more averagely distributed across all possible input combinations.This has nothing to do with if it is positively correlated or negatively correlated. Correlation violates the conditional independence and could give rise to weird parameters in learning.

2 a)
i) The variance in the first representation will probably be fairly low -- students will likely not take most of the courses in school of informatics, therefore the feature vectors will consist of mostly average scores. Even when the student took a course, the mark is likely to be close to average anyway, further reducing the variance.

The second representation is doomed to have a lot of zeros -- it is probably fair to assume that most students will get quite similar scores for all their coursework on average, i.e. between 60-80%, therefore 0-10% slot is likely not to get filled at all. Also it is not clear whether the 10% split is the correct one, maybe the data is best modelled when it is split in 5% intervals. Finally this does not encode in any way the fact that the score of 80% is better than the score of 40%.
It doesn't? Those who have a higher percentage of coursework in the upper ranges will likely get higher final marks against those who have a higher percentage of coursework in the middle ranges → encoding the fact that a score of 80% is better than a score of 40%.
This encoding also suffers the dummy variable trap. The sum of the entries is 100% so they are collinear.

ii) We could use question marks instead of average values in the first representation, i.e. have our feature vectors similar to this:

$$x_i = (15,\ ?,\ ?,\ 32,\ ?,\ \ldots.,\ 1,\ ?,\ 21)$$

Then when computing Naive Bayes likelihood, we can just ignore the missing values, from the calculations, because they marginalise nicely.

Formal proof for above:

Under naive bayes, the likelihood $P(x_1, x_2, \ldots, x_n|D) = \prod_i P(x_i|D)$ where $(x_1, \ldots, x_n)$is some data point (i.e. this is for one data point, not for whole dataset).
now assume that $x_j$is a missing value. w.l.o.g we can assume that it is the only one, then the likelihood of all other points, except the missing one can be computed by marginalising it out:

$$P(x_1, \ldots., x_{j-1}, x_{j+1}, \ldots, x_n) = \int P(x_1, x_2, \ldots, x_j, \ldots, x_n)\, dx_j = \int \prod_i P(x_i|D)\, dx_j = \prod_{i \neq j} P(x_i|D) \int P(x_j|D)\, dx_j$$

and thus

$P(x_1,\ldots,\ x_{j-1}, x_{j+1},\ldots, x_n) = \prod_{i \neq j} P(x_i|D)$

So all missing values can be simply ignored in naive bayes calculations.

I think he wants a mixture model representation since the EM algorithm explicitly deals with missing data.

The above answer with NB is definitely correct. - Riaz

2b)

i)
not sure about standardising input, this question kind of covers it, but I dont find it convincing: http://stackoverflow.com/questions/4674623/why-do-we-have-to-normalize-the-input-for-an-artificial-neural-network
the book says “good practice to standardize the inputs to zero mean and unit variance, so that the spherical Gaussian prior makes sense”

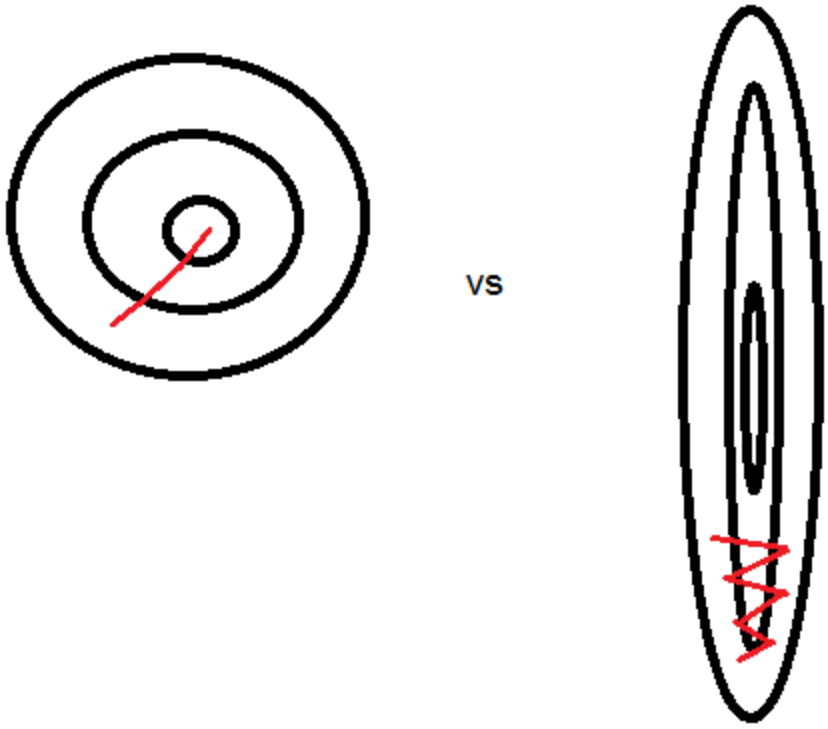
Additionally to above:

It is essential to rescale the inputs so that their variability reflects their importance, or at least is not in inverse relation to their importance. For lack of better prior information, it is common to standardize each input to the same range or the same standard deviation. If you know that `some inputs are more important than others, it may help to scale the inputs such that the more important ones have larger variances and/or ranges. An example of bad standardisation: If one input has a range of 0 to 1, while another input has a range of 0 to 1,000,000, then the contribution of the first input to the distance will be swamped by the second input.

regarding small weights - well, thats in order to reduce overfitting - large weights mean value of function changes faster with change of x values, causing wiggly functions, which often indicate overfitting, smaller weights make models simpler.

Additionally to above:
We want to pick values of the weights at random following a distribution which helps the optimization process to converge to a meaningful solution.
The main emphasis in the NN literature on initial values has been on the avoidance of saturation, hence the desire to use small random values. How small these random values should be depends on the scale of the inputs as well as the number of inputs and their correlations. Standardizing inputs removes the problem of scale dependence of the initial weights.

Standardising inputs also changes the shape of the error surface towards more circular parabolic bowls rather than elongated elliptical parabolic bowls - this helps learning by making the strongest direction of the gradient point more towards the centre of the bowl rather than having steep canyon like bowls, see below:



ii) ? - Having the same weights for all of them might be bad because when you're running backpropagation you end up with the same rates of change for each node and they all end up with identical weights (+1 to here, but not about the single node) which means that you're adding nothing new to the model and you might as well be representing it with just a single node.

Not sure if this is right though.

Turns out single node equivalence is correct (https://class.coursera.org/ml-003/lecture/55)

This means that all attributes have the same power to discriminate the classes. This is rarely the case. I don't agree that it is the same as running with a single node, the attributes are different, we just think they are all equally important.

Agree that if hidden units are assigned the same initial weights, all of their weights are going to be the same after training.
Since each hidden unit is supposed to learn a (linear) function of its inputs, if they have the same weights then they will learn the same function. In this case, the neural network loses its flexibility of learning complex functions.

No matter what was the input - if all weights are the same, all units in hidden layer will be the same too.

iii) It is relatively simple to the weight decay bias. The models will tend to become complicated as time progresses, stopping early effectively forces the model be simpler and thus overfit less.

Could this also have to do with bias on the subset of data used during early training?

2c)

Define error function E = -L(D|w), which we want to decrease.
Initialise w to some (random) vector
While E is unacceptably high,
    compute the gradient of E, call it g
    set the weight vector to be $w^{new} = w^{current} - \eta g$
    continue from w^{new}
return w
The problem of setting the learning rate, $\eta$is the tradeoff between setting the rate to a low value, meaning slow convergence, and setting the rate to large value, meaning oscillation around the lowest value.

I don't think this is related to neural network training; It is just general gradient descent algorithm?. Doesn't he want the back propagation algorithm outlined here? Maybe, but I think in this case the above answer is correct, the backpropagation algorithm comes into play when calculating the gradient of E, therefore the above is right in both cases (e.g. general and backprop)?

Isn't the gradient ascent used in backpropagation? I think in this case we need to explain the stochastic gradient ascent (one of the online methods), that updates one weight at a time rather than all of them together.

QUESTION CONTINUED FROM PREVIOUS PAGE

(d) Let $\sigma(x)$ be the logistic function, and so $\frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$. Consider the single datum likelihood for a logistic regression model:

$$P(y_i = 1|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x}) \quad (3)$$

(where the bias is included in $\mathbf{w}$ by augmenting the data $\mathbf{x}$ with a unit attribute). Show the derivative of the negative log-likelihood with respect to w_i is

$$-(1 - \sigma(\mathbf{w}^T\mathbf{x}))x_i \quad (4)$$

and hence derive the form for an element of the Hessian matrix (the matrix of second derivatives) for the logistic regression model. Using the fact that a convex function has no *local* minima, and the fact that the matrix $\mathbf{x}\mathbf{x}^T$ has all eigenvalues greater than or equal to zero, show that parameter estimation for logistic regression has at most a single maximum. [*6 marks*]

Wibi: This is for 2(d), on proving f = negative log likelihood of logistic regression has no local minimum.

Some facts (from the question, from derivation of the Hessian, and from wikipedia):
1. X = xx'
2. All eigenvalues of X is non-negative
3. If all eigenvalues of M is non-negative --> M is positive semi-definite
4. If M is the Hessian of f and M is positive semi-definite --> f is convex
5. If f is convex --> f has no local minimum
6. If M is positive semi-definite and r > 0 --> rM is positive semi-definite
7. d = σ(w'x)(1 - σ(w'x)) > 0
8. dX is the Hessian of f ... from derivation of the Hessian

From these,
9. X is positive semi-definite ... from 2 & 3
10. dX is positive semi-definite ... from 9 & 6
11. f is convex ... from 8, 10 & 4
12. f has no local minimum ... from 11 & 5

3. (a) For real $\mathbf{y}$ and binary c, let $P(\mathbf{y}|c, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$ be a class conditional Gaussian distribution with mean $\boldsymbol{\mu}_c$ and covariance matrix $\boldsymbol{\Sigma}_c$:

$$P(\mathbf{y}|c, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) = \frac{1}{|2\pi\boldsymbol{\Sigma}_c|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu}_c)^T\boldsymbol{\Sigma}_c^{-1}(\mathbf{y}-\boldsymbol{\mu}_c)\right) \quad (5)$$

Consider using a class-conditional model for a problem with two classes $c \in \{0, 1\}$. You are given learnt $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$, where the $\boldsymbol{\Sigma}_c$ are constrained to be identical covariance matrices for both classes $c = 0, 1$

i. How would you estimate $P(c)$ from a dataset of size N? What assumptions are you making? [*2 marks*]

ii. Write down how to obtain the probability of the class label $P(c|\mathbf{y})$ in terms of $P(\mathbf{y}|c)$ and $P(c)$. [*1 mark*]

iii. If $P(c = 1) = P(c = 0)$ in this situation, show that the decision boundary for the classification $P(c|\mathbf{y})$ is linear (assume we classify using the most probable class). [*6 marks*]

(b) It is possible to combine class-conditional modelling and Gaussian processes to make a class conditional Gaussian process model. In this model $P(\mathbf{y}|c, \mathbf{x}) = \prod_i P(y_i|c, \mathbf{x})$ ($\mathbf{y} \in \mathbb{R}$, $c \in \{0, 1\}$) where, for each c, $P(y_i|c, \mathbf{x})$ is a Gaussian process model. A prior $P(c|\mathbf{x}) = P(c)$ is also given. The aim is to predict $P(c|\mathbf{x}, \mathbf{y})$. Alternatively, $P(c|\mathbf{x}, \mathbf{y})$ can be modelled directly with a Gaussian process classifier.

i. Discuss the computational advantages of the class conditional Gaussian process model over the Gaussian process classifier (give at least two important advantages for full marks). [*4 marks*]

ii. What is one modelling disadvantage of the class conditional Gaussian process model over the Gaussian process classifier (give a problematic assumption of a class conditional Gaussian process described above). [*2 marks*]

(c) Define the variational approximation to the posterior distribution $P(\theta|\mathcal{D}, h)$, where h is a set of hyper-parameters. Show, using the fact that the $KL(Q||P) \geq 0$ for two distributions Q and P, that the marginal likelihood $P(\mathcal{D}|h)$ can be lower bounded through the use of the KL divergence. [*5 marks*]

(d) Define detailed balance for a Markov chain, and show that if detailed balance holds for a distribution $Q(X)$ and transition $P(X_t|X_{t-1}) = P(X_1|X_0)$, then that distribution must be a fixed point of the Markov Chain. In other words, if X_t is distributed as $Q(X_t)$ then taking one step of the Markov Chain $X_t \rightarrow X_{t+1}$, leaves X_{t+1} distributed as $Q(X_{t+1})$. [*5 marks*]

3a)

i. $p(c) = \frac{N_c}{N}$ where N_c is the number of items in a class. I am making an assumption that the dataset is a representative sample of population, and using MLE estimate for prior probability of class.

ii. Bayes rule $p(c|y) = \frac{P(c,y)}{P(y)} = \frac{P(c,y)}{\sum_c p(c,y)} = \frac{P(y|c)P(c)}{\sum_c p(y|c)p(c)}$

iii. http://www.inf.ed.ac.uk/teaching/courses/pmr/scans/PMR_Gaussian_classifier.pdf

3b) Gaussian processes are not in the course this year
3c) Start with a definition of KL(Q||P):

$KL(Q||P) = \int Q(\theta|D,h)\, log\, \frac{Q(\theta|D,h)}{P(\theta|D,h)}\, d\theta = E_q[log\, Q(\theta|D,h)] - E_q[log\, P(\theta|D,h)]$ (1)

Here, I define $E_q[F]$to be expectation of F under probability distribution Q, i.e.

$E_q[F] = \int Q(\theta|D,h)\, F(\theta)\, d\theta$

now

$P(\theta|D,h) = \frac{P(\theta,D|h)}{P(D|h)}$(2)

I assume we can compute $P(\theta, D|h)$

then putting (2) into (1) we get

$KL(Q||P) = E_q[log\, Q(\theta|D,h)] - E_q[logP(\theta,D|h)] + E_q[logP(D|h)]$

Now note that $E_q[logP(D|h)] = \int Q(\theta|D,h)P(D|h)\, d\theta = P(D|h) \int Q(\theta|D,h)d\theta = P(D|h)$

(because $P(D|h)$ does not depend on $\theta$and assuming Q is normalised (which it should be))

Therefore $KL(Q||P) = E_q[log\, Q(\theta|D,h)] - E_q[logP(\theta,D|h)] + logP(D|h)$

From $KL(Q||P) \geq 0$

$logP(D|h) \geq -(E_q[logQ(\theta|D,h)] - E_q[logP(\theta,D|h)]) \equiv -J$

Therefore, $-J$provides a lower bound on $log\, P(D|h)$

3d)

Detailed balance for markov chains is defined as:

$P(X_{t+1} = A|X_t = B)Q(B) = P(X_{t+1} = B|X_t = A)Q(A)$

Not 100% sure what the next part asks us to show, but I think that is the formal representation of what we need to prove:

$\sum_{x_t} Q(x_t)P(x_{t+1}|x_t) = Q(x_{t+1})$

checked slides, this seems to be correct:

Equilibrium Distribution: an ergodic Markov chain has a unique equilibrium distribution $P_\infty(\theta)$ such that

$$P_\infty(\boldsymbol{\theta}) = \int d\boldsymbol{\theta}' \; P_T(\boldsymbol{\theta}|\boldsymbol{\theta}')P_\infty(\boldsymbol{\theta}')$$

This follows directly from detailed balance, noting that $Q(x_t)P(x_{t+1}|x_t) = Q(x_{t+1})P(x_t|x_{t+1})$

Then,

$$\sum_{x_t} Q(x_t)P(x_{t+1}|x_t) = \sum_{x_t} Q(x_{t+1})P(x_t|x_{t+1}) = Q(x_{t+1}) \sum_{x_t} P(x_t|x_{t+1}) = Q(x_{t+1})$$

If x_t are not discrete, replace sums by integrals, the result should still hold.