Machine Learning 2020

Practice Exam A

WITH ANSWERS

March 19, 2020

This document provides a detailed explanation of the exam structure with some example questions for each category. It is *not* an example of what the actual exam will look like. For that, see Practice Exam B.

Some tips:

- Pay particular attention to **the application questions**. These are the most difficult, but they are also the most *predictable*. If you have a look at them early on, you should know what to pay attention to as the course progresses.
- The exam contains easy questions as well as tricky ones. Don't make the mistake of suspecting trick questions when something seems too good to be true. Some questions are just easy.
- Focus on revising for the recall questions and the application questions. If you practice these well, the exam should be easy to pass.
- Note that application questions follow a fixed pattern. If you run out of practice questions, you can easily create your own examples.
- The exam is **only two hours**. To get a good grade you should practice enough to answer questions quickly as well as accurately.

1 Recall questions

Approximately one third of the exam will be *recall questions*. These are questions that ask for a simple detail from a single slide without too much depth. If you have seen and understood every slide of every lecture once, you should be able to answer the majority of recall questions correctly. These questions are never phrased to trick you or to catch you out.

Examples of recall questions

1. What separates **offline learning** from **reinforcement learning**?

   A In reinforcement learning the training labels are reinforced through boosting.
   B Offline learning can be done without connection to the internet. Reinforcement learning requires reinforcement from a separate server.
   C In reinforcement learning, the learner takes actions and receives feedback from the environment. In offline learning we learn from a fixed dataset. ✓
   D Reinforcement learning uses backpropagation to approximate the gradient, whereas offline learning uses symbolic computation.

   See lecture 1. Note that the incorrect questions are phrased to sound reasonable if you don't know the specifics, but are mostly nonsense, if you know what these concepts mean.

2. The most important rule in machine learning is "never judge your performance on the training data." If we break this rule, what can happen as a consequence?

   A The loss surface no longer provides an informative gradient.
   B We get cost imbalance.
   C We end up choosing a model that overfits the training data.✓
   D We commit multiple testing.

   Introduced in lecture 1, also discussed in lecture 3.

3. We have a classifier c and a test set. Which is **true**?

   A To compute the *precision* for c on the test set, we must define how to turn it into a ranking classifier.
   B To compute the *false positive rate* for c on the test set, we must define how to turn it into a ranking classifier.
   C To compute the *confusion matrix* for c on the test set, we must define how to turn it into a ranking classifier.
   D To compute the ROC *area under the curve* for c on the test set, we must define how to turn it into a ranking classifier.✓

   The first three can be computed for any binary classifier. For the ROC AUC, however, we need a ranking of the instances in the test set from most negative to most positive. See lecture 3 and homework 3.

4. Testing too many times on the test set increases the chance of random effects influencing your choice of model. Nevertheless, we may need to test many different models and many different hyperparameters. What is the solution suggested in the lectures?

   A To withhold the test set, and use a train/validation split on the remainder to evaluate model choices and hyperparameters. ✓
   B To normalize the the data so that they appear normally distributed. Normalization will not help with this problem.
   C To use bootstrap sampling to gauge the variance of the model. Bootstrap sampling (lecture 3 and lecture 10) *will* help you gauge the variance. But that will not solve this problem.
   D To use a boosted ensemble, to reduce the variance of the model, and with it, the probability of random effects. An ensemble (lecture 10) can increase performance by reducing bias and variance, but this does not help with the problem of test-set reuse (lecture 3). It's just another way of building a model.

2 Combination questions

Approximately one third of the exam will be *combination questions*. These are slightly deeper than simple recall questions. They may, for instance. require you to

- combine pieces of information from different parts of the lecture,
- Answer a question posed in the negative, i.e. "Which is *not* one of the reasons that ...?",
- recognise that an answer that seems correct is actually not true. We won't write trick question for the sake of tricking you, but sometimes it's important to include common misconceptions.

The difference between recall and combination questions is a little fuzzy, but hopefully, you can infer the general idea from the examples below.

Examples of combination questions

5. Different features in our data may have wildly different scales: a person's age may fall in the range from 0 to 100, while their savings can fall in the range from 0 to 100 000. For many machine learning algorithms, we need to modify the data so that all features have roughly the same scale. Which is **not** a method to achieve this?

A Imputation ✓
B Standardization
C Normalization
D Principal Component Analysis

This question is phrased negatively. This usually makes it more difficult to guess. If you know that imputation is a way to deal with missing data (not normalization), you can get the answer that way. To get the answer by elimination, you'll have to know that B, C and D can all be used for normalization of the features. D is a tricky one, since PCA is more commonly used for dimensionality reduction, but it can also be used for normalization.

6. The *variational* autoencoder adapts the regular autoencoder in a number of ways. Which is **not** one of them?

   A It adds a sampling step in the middle.
   B It makes the outputs of the encoder and decoder parameters of probability distributions.
   C It adds a loss term to ensure that the latent space is laid out like a standard normal distribution.
   D It adds a discriminator that learns to separate generated examples from those in the dataset. ✓

   Again, a negative question. You'll need to know all the differences between an autoencoder and a VAE in order to get the answer, or you need to recognize that a discriminator is component of a GAN, not a VAE.

7. We have two discrete random variables: A with outcomes $1, 2, 3$ and B with outcomes a, b, c. We are given the joint probability $p(A, B)$ in a table, with the outcomes of A enumerated along the rows (vertically), and the outcomes of B enumerated along the columns (horizontally). How do we compute the probability $p(A = 1 \mid B = a)$?

   A We find the probability in the first column and the first row.
   B We find the probability in the first column and the first row, and divide it by the sum over the first column. ✓
   C We find the probability in the first column and the first row, and divide it by the sum over the first row.
   D We sum the probabilities over the first column and the first row.

   The answer to this question resides in a single slide (in lecture 5), but you'll have to have remembered the concept of conditional probability well enough to transform the information here to the correct picture.

8. Why is gradient descent difficult to apply in a reinforcement learning setting?

A The loss surface is flat in most places, so the gradient is zero almost everywhere.
B The backpropagation algorithm doesn't apply if the output of a model is a probability distribution.
C There is a non-differentiable step between the model input and the model output.
D There is a non-differentiable step between the model output and the reward.✓
This information is again contained in a single point in the lecture series, but it's at the end (in the 13th lecture) and not extensively discussed.

9. The squared error loss function looks like this: $\sum_i (y_i - t_i)^2$, where the sum is over all instances, y_i is the model output for instance i and t_i is the training label. Which is **not** a reason for squaring the difference between the two?
A It ensures that negative and positive differences don't cancel out in the sum.
B It ensures that large errors count very heavily towards the total loss.
C When used in classification, it ensures that points near the decision boundary weigh most heavily.✓
D It is a consequence of assuming normally distributed errors, and deriving the maximum likelihood solution.
Again, a negative question. Answers A and B were discussed in lecture 1 and repeated in lecture 2. Answer D was discussed in the second probability lecture. Answer C, however, is the opposite of what the least squares loss does. It is true for the log loss.

3 Application questions

The final third of the exam will be *application questions*. These are questions that ask you to apply an algorithm, perform some computation or follow some derivation. These are the questions which you'll need to actively practice for.

All application questions follow a predetermined pattern and are practiced in the homework exercises.

There are 10 types, with a sequence of about three questions for each type. For each exam we will select some types, and adapt the specifics of the question but not the structure. For instance, we may change the dataset or change which parameter we take a derivative for.

The following is **a complete list** of all types. If you master all 10 types given below, there will be no surprises on the exam.

1. **Find the gradient** For a simple (usually polynomial) model, work out the derivative with respect to one of the parameters.
2. **Find a ranking** Given a simple dataset, and a linear classifier work out a ranking of the instances, and identify the number of ranking errors and the coverage.
3. **Entropy** For a given set of probability distributions, compute the entropy and the cross-entropy.
4. **Scalar backpropagation** Apply the backpropagation algorithm to a complicated scalar function. Break the function up into modules and use the local derivatives to compute the derivative for a particular input.
5. **Decision trees** Given a dataset, work out which feature makes for the best split.
6. **Evidence lower bound** Work through the derivation of the *evidence lower bound* (as used in the EM and VAE algorithms) and identify the missing steps.
7. **Naive Bayes** Given a dataset with categorical variables, compute the probabilities that a naive Bayes classifier assigns to each class, with and without smoothing.
8. **The kernel trick** Work out the explicit feature space for a given kernel.
9. **Matrix backpropagation** For a given module in a matrix-based automatic differentiation system, work out the Jacobian/vector products required to implement the backward pass.
10. **Lagrange multipliers** Work out the optimum for a constrained optimization problem, using Lagrange multipliers.

In some cases there are questions included to check that you understand *what* you're doing and what it means, in addition to knowing *how* to do it. These will be entirely different in the exam.

Examples of application questions

type: find the gradient

We want to train the following model:

$$y_i = -v{x_i}^2 + wx_i + b$$

with parameters v, w and b, where x_i and y_i are scalars. the dataset provides target values t_i. We derive the gradient of the loss with respect to b as follows:

$$
\begin{aligned}
\frac{\partial \frac{1}{2}\sum_i (y_i - t_i)^2}{\partial b} &= \frac{1}{2}\frac{\partial \sum_i (y_i - t_i)^2}{\partial b} && (1)\\
&= \frac{1}{2}\sum_i \frac{\partial (y_i - t_i)^2}{\partial b} && (2)\\
&= \frac{1}{2}\sum_i \frac{\partial (y_i - t_i)^2}{\partial (y_i - t_i)}\frac{\partial (y_i - t_i)}{\partial b} && (3)\\
&= \frac{1}{2}\sum_i 2(y_i - t_i)\frac{\partial (y_i - t_i)}{\partial b} && (4)
\end{aligned}
$$

10. To get from line (2) to line (3), we use the
    A Chain rule ✓
    B Product rule
    C Constant factor rule
    D Sum rule
11. To get from line (1) to line (2), we use the
    A Chain rule
    B Product rule
    C Constant factor rule
    D Sum rule ✓
    The names of the rules are given on your cheat sheet so you don't have to memorize them. Just make sure that you understand what is happening in steps like these, and you can look up the answer.
12. Fill in the definition of y_i and work out the derivative with respect to b. Which is the correct result?
    A $\frac{1}{2}\sum_i \left(-v{x_i}^2 + wx_i + b - t_i\right)$
    B $\sum_i \left(-v{x_i}^2 + wx_i + b - t_i\right)$ ✓
    C $\frac{1}{2}\sum_i x_i \left(-v{x_i}^2 + wx_i + b - t_i\right)$
    D $\sum_i x_i \left(-v{x_i}^2 + wx_i + b - t_i\right)$
    Note that the $\frac{1}{2}$ in front of the loss is canceled out by the 2 factor we received in step 4. For more practice try taking the derivative wrt to w. You should get answer D.

type: find a ranking

We have the following training set:

	x_1	x_2	label
a	0	1	Neg
b	2	2	Neg
c	1	4	Neg
d	2	5	Neg
e	3	6	Pos
f	6	8	Pos
g	5	3	Pos
h	8	7	Pos

For the following questions, it helps to draw the data and the classification boundary in feature space.
We use a linear classifier defined by

$$c(x_1, x_2) = \begin{cases} \text{Pos} & \text{if } 0 \cdot x_1 + x_2 - 2 > 0 \\ \text{Neg} & \text{otherwise.} \end{cases}$$

Note that this classifier ignores x_1, so we can make things easy for ourselves by just looking at x_2. The bigger x_2, the more positive the class (careful, this may be the other way around in other questions), so we can just sort the instances by x_2.

It will never be more difficult than this on the exam, but make sure you know how you'd do this with multiple features, or with a decision tree classifier.

13. If we turn c into a *ranking* classifier, how does it rank the points, from most Negative to most Positive?

A a b c d e f g h
B a b g c d e h f✓
C a b g c d h e f
D a c b d e g f h
See lecture 3 and homework 3. For the linear classifier, we use the distance to the decision boundary to rank the points.

14. How many ranking errors does the classifier make?

A None
B 1
C 2 ✓
D 3
This question is a *very* common pitfall for students. Make sure you understand what a *ranking error* is. Here's a hint: on a dataset of 5 instances, a classifier can make as many as *10* ranking errors.

15. If we draw a coverage matrix (as done in the slides), what proportion of the cells will be red?

A $\frac{3}{15}$ **B** $\frac{6}{15}$ **C** $\frac{1}{8}$✓ **D** $\frac{6}{16}$

In the coverage matrix, each cell is a pair of instances, consisting of a positive and a negative pair (all pairs that might cause a ranking error). This question asks for the proportion of ranking errors made (see the previous question) to the total proportion of *possible* ranking errors. Note that in this case we do simplify the fraction from 2/16 to 1/8.

type: Entropy

Here are two distributions, p and q, on the members of a set $X = \{a, b, c, d\}$. We will use binary entropy (i.e. computed with base-2 logarithms). Note that in the computation of the entropy $0 \cdot \log_2(0) = 0$, but otherwise, $\log_2 0$ is undefined as usual.

	p	q
a	¼	0
b	¼	¼
c	¼	¼
d	¼	½

16. What are their entropies?

    A $H(p) = 2$, $H(q) = 1$
    B $H(p) = 2$, $H(q) = 1.5$✓
    C $H(p) = 1$, $H(q) = 1$
    D $H(p) = 1$, $H(q) = 1.5$

17. What is the cross-entropy $H(q, p)$?

    A $H(q, p) = 1$
    B $H(q, p) = 1.5$
    C $H(q, p) = 2$✓
    D $H(q, p) = 2.5$

    These two questions are basic application of the entropy formulas on your cheat sheet. However, to calculate this quickly, it can help to write out the sum as logarithms, and apply what you know about logarithms to simplify the sum, before you grab the calculator. For instance:

$$\begin{aligned}
-H(q) &= 0 \log 0 + \frac{1}{4}\log\frac{1}{4} + \frac{1}{4}\log\frac{1}{4} + \frac{1}{2}\log\frac{1}{2} \\
&= \frac{2}{4}\log\frac{1}{4} + \frac{2}{4}\log\frac{2}{4} \\
&= \frac{2}{4}(\log\frac{1}{4} + \log\frac{2}{4}) \\
&= \frac{2}{4}(-\log 4 + \log 2 - \log 4) \\
&= \frac{2}{4}(-2 + 1 - 2) = -1.5
\end{aligned}$$

It's extremely easy to make sign errors when dealing with entropy. Computing $-H$ makes the formula a positive sum, which makes things a little simpler.

With a bit of practice, this sort of thing is much easier (and more accurate) than punching the formula into the calculator explicitly.

18. If you try to compute $H(p, q)$, you'll notice that something goes wrong. What does this mean?
A As $q(a)$ goes to zero, a's codelength with code q goes to infinity. This makes the expected codelength under the uniform distribution p infinite as well.✓
B Because p is uniform, its expected codelength is always optimal under *any* distribution.
C A standard (graphical) calculator does not have the precision to compute the answer. In a python notebook, we would not have a problem.
D Because $q(a) = 0$ the expected codelength with code q, under distribution q is not defined. This means that the resulting cross-entropy also becomes undefined.
We will occasionally throw in questions like this to test your understanding of a formula. This is a particularly tough one: it requires you to understand the intuitive explanation of entropy in lecture 5, and how it leads to the formula for entropy.

Specifically, the correspondence between codelength and probability: the lower the probability, the higher the codelength. If the probability is 0, the codelength becomes infinite. This wasn't problem for $H(q)$, because $0 \log 0$ is defined to be 0 in the context of entropy, but for the cross-entropy, we end up with an infinity.

type: Backpropagation

NB: This question type is not just asking for a derivative (although the function may be simple enough for that). What we want is for you to apply the *backpropagation algorithm: that is, to work our the local derivatives symbolically, and then do the rest numerically.*

We will use the backpropagation algorithm to find the derivative of the function

$$f(x) = \sin\left(\sin(x)\cos(x)\right)$$

with respect to x.

First, we break the function up into modules:

$$\begin{aligned} f &= \sin(c) \\ c &= ab \\ b &= \cos(x) \\ a &= \sin(x) \end{aligned}$$

19. What should be in the place of the dots?

A $b = \cos(x)$, $c = ab$✓
B $b = \cos(x)$, $c = \sin(x)\cos(x)$
C $b = \cos(c)$, $c = ab$
D $b = \cos(c)$, $c = \sin(x)\cos(x)$

20. For the backpropagation algorithm, we need to work out the local derivatives symbolically. Which are the required local derivatives?

A $\partial f/\partial c$, $\partial c/\partial a$, $\partial c/\partial b$, $\partial a/\partial x$ and $\partial b/\partial x$ ✓
B $\partial c/\partial f$, $\partial a/\partial c$, $\partial b/\partial c$, $\partial x/\partial a$ and $\partial x/\partial b$
C $\partial f/\partial c$, $\partial f/\partial a$, $\partial f/\partial b$ and $\partial f/\partial x$
D $\partial c/\partial f$, $\partial a/\partial f$, $\partial b/\partial f$ and $\partial x/\partial f$

21. In terms of the local derivatives. Which is the correct expression for the gradient $\partial f/\partial x$?

A $\sin(c)\,[b\cos(x) + a\sin(x)]$
B $\sin(c)\,[b\cos(x) - a\sin(x)]$
C $\cos(c)\,[b\cos(x) + a\sin(x)]$
D $\cos(c)\,[b\cos(x) - a\sin(x)]$✓

The most difficult part of this question type is (usually) the application of the *multivariate chain rule* (see the Deep Learning 1 lecture): this rule tells us how the chain rule works if the variable for which we are taking the derivative (in this case x) affects the output (f) along multiple paths. It can help to draw a diagram of the computation graph over the modules:

$$\begin{tikzcd}
 & a \arrow[dr] & & \\
x \arrow[ur] \arrow[dr] & & c \arrow[r] & f \\
 & b \arrow[ur] & &
\end{tikzcd}$$

The multivariate chain rule says that because x influences f along two paths along the computation graph, we can take the derivatives along both paths and sum them. The variables that on the other paths are taken as constants. In this cases:

$$\begin{aligned}
\frac{\partial f}{\partial x} &= \frac{\partial f}{\partial c}\frac{\partial c}{\partial x} && \text{application of the basic chain rule}\\
&= \frac{\partial f}{\partial c}\left(\frac{\partial c}{\partial a}\frac{\partial a}{\partial x} + \frac{\partial c}{\partial b}\frac{\partial b}{\partial x}\right) && \text{mv chain rule to simplify } \frac{\partial c}{\partial x}\\
&= \frac{\partial f}{\partial c}\frac{\partial c}{\partial a}\frac{\partial a}{\partial x} + \frac{\partial f}{\partial c}\frac{\partial c}{\partial b}\frac{\partial b}{\partial x} && \text{simplify}
\end{aligned}$$

We can now work out the local derivatives for each module, fill them in, and simplify. See homework exercise 5 for the rest of the process.

type: Decision trees

Consider the following task. The aim is to predict the class y from the binary features x_1, x_2, x_3 and x_4.

x_1	x_2	x_3	x_4	y
B	A	A	A	Yes
A	A	B	A	No
B	A	A	A	Yes
A	A	B	A	No
B	B	A	A	Yes
B	B	A	A	Yes
B	A	A	B	Yes
A	B	B	B	No
A	A	B	B	No
A	B	A	B	Yes
B	B	B	B	No
A	B	B	B	No

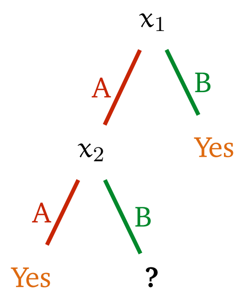
22. In standard decision tree learning (as explained in the lectures), without pruning. Which would be the first feature chosen for a split?
A x_1 **B** x_2 **C** x_3✓ **D** x_4

    You can work this question out by computing the information gain, but that's slow and error-prone. It's usually quicker to do it intuitively. The best split causes the most uneven distribution over the classes Yes and No on both sides of the split. In this case, x_3 causes only Yes on the A side of the split and only No on the B side (that is, after the split, all instances have the same class and the entropy is zero for both).

23. If we remove that feature from the data, which would be chosen instead?
A x_1✓ **B** x_2 **C** x_3 **D** x_4

    Note the symmetry in the data. Splitting on x_4 causes a 2/4 distribution on *both* sides of the split. Splitting on x_2 causes a 3/3 distribution and splitting on x_1 causes a 1/5 distribution. If you understand entropy, you won't need the calculator to know that x_1 has the highest information gain.

Consider the following (partial) decision tree:



We can place either x_3 or x_4 on the open node (indicated by the question mark).

24. At this point, what is (approximately) the information gain for x_3?
A 0 **B** 0.91✓ **C** 1.41 **D** 2.75

For both leaves after splitting on x_3, there is only one class left, so the entropy is 0 for both. Therefore, the information grain is $H(S) - 0$, where $H(s)$ is the entropy of $S = \{\text{No}, \text{No}, \text{Yes}\}$.

Tip: check your answer against your intuitive understanding of entropy. We know that the answer is equal to $H(S)$, which is the optimum way of encoding a draw from S, or equivalently, a draw from a distribution with $\left(\frac{1}{3}, \frac{2}{3}\right)$. 1 bit is the entropy for a uniform distribution, so it must be less than that. It also can't be zero, because that would mean we're sure of the outcome.

type: Evidence lower bound

NB: This question type involves filling in the blanks in a derivation. You are free to just memorize the derivation, but note that half the points come from understanding what the derivation means.

The EM and VAE algorithms are both based on the following decomposition.

$$\begin{aligned}
L(q,\theta) + KL(q,p) &= \mathbb{E}_q \ln \frac{p(x, z \mid \theta)}{q(z \mid x)} - \mathbb{E}_q \ln \frac{p(z \mid x, \theta)}{q(z \mid x)} \\
&= \mathbb{E}_q \ln p(x, z \mid \theta) - \mathbb{E}_q \ln q(z \mid x) - \mathbb{E}_q \ln p(z \mid x, \theta) + \mathbb{E}_q \ln q(z \mid x) \\
&= \mathbb{E}_q \ln p(x, z \mid \theta) - \mathbb{E}_q \ln p(z \mid x, \theta) \\
&= \mathbb{E}_q \ln \frac{p(x, z \mid \theta)}{p(z \mid x, \theta)} \\
&= \mathbb{E}_q \ln \frac{p(z \mid x, \theta) p(x \mid \theta)}{p(z \mid x, \theta)} \\
&= \mathbb{E}_q \ln p(x \mid \theta) \\
&= \ln p(x \mid \theta)
\end{aligned}$$

Note that while we start with $L(q, \theta) + KL(q, p)$ and end with $\ln p(x \mid \theta)$, the actual point of this derivation is to simplify $\ln p(x \mid \theta)$ (which we're actually interested in) into something computable.

25. What should be in place of $\langle a \rangle$ and $\langle b \rangle$?
A $\langle a \rangle : \mathbb{E}_q \ln p(z \mid x, \theta)$, $\langle b \rangle : \ln p(x \mid \theta)$ ✓
B $\langle a \rangle : \mathbb{E}_q \ln p(z \mid x, \theta)$, $\langle b \rangle : \mathbb{E}_q \ln p(x)$
C $\langle a \rangle : \mathbb{E}_q \ln p(z, x \mid \theta)$, $\langle b \rangle : \ln p(x \mid \theta)$
D $\langle a \rangle : \mathbb{E}_q \ln p(z, x \mid \theta)$, $\langle b \rangle : \mathbb{E}_q \ln p(x)$

26. How is this derivation used in the EM algorithm?

A To maximize $\ln p(\mathbf{x} \mid \theta)$, we iterate between choosing θ to maximize $L(q, \theta)$ and then choosing q to minimize $\text{KL}(q, p)$.✓
B To maximize $\ln p(\mathbf{x} \mid \theta)$, we treat $L(q, \theta)$ as a lower bound and optimize its parameters by backpropagation.
C To maximize $L(q, \theta) + \text{KL}(q, p)$ we rewrite it to $\ln p(\mathbf{x} \mid \theta)$ and use random search to find the optimal θ.
D To maximize $L(q, \theta) + \text{KL}(q, p)$ we rewrite it to $\ln p(\mathbf{x} \mid \theta)$ and apply the kernel trick to find the optimal θ.

The VAE requires further rewriting. To simplify notation, we will omit the parameters θ.

$$
\begin{aligned}
-\ln p_\theta(x) \geqslant -L(q, \theta) &= -\mathbb{E}_q \ln p(x, z) + \mathbb{E}_q \ln q(z \mid x) \\
&= -\mathbb{E}_q \ln p(x \mid z) - \mathbb{E}_q \ln p(z) + \mathbb{E}_q \ln q(z \mid x) \\
&= -\mathbb{E}_q \ln p(x \mid z) + \mathbb{E}_q \ln \frac{q(z \mid x)}{p(z)} \\
&= -\mathbb{E}_q \ln p(x \mid z) - \mathbb{E}_q \ln \frac{p(z)}{q(z \mid x)} \\
&= -\mathbb{E}_q \ln p(x \mid z) + \text{KL}\left(q(z \mid x), p(z)\right) \\
&= -\mathbb{E}_q \ln p(x \mid z) + \text{KL}(q(z \mid x), N(0, \mathbf{I}))
\end{aligned}
$$

27. What should be in place of $\langle a \rangle$ and $\langle b \rangle$?

A $\langle a \rangle : \quad \ln \mathbb{E}_q p(x, z) - \ln \mathbb{E}_q q(z \mid x), \quad \langle b \rangle : N(0, \mathbf{I})$
B $\langle a \rangle : -\ln \mathbb{E}_q p(x, z) + \ln \mathbb{E}_q q(z \mid x), \quad \langle b \rangle : N(0, \mathbf{I})$✓
C $\langle a \rangle : \quad \ln \mathbb{E}_q p(x \mid z) - \ln \mathbb{E}_q q(z \mid x), \quad \langle b \rangle : N(z, \text{var}(z))$
D $\langle a \rangle : -\ln \mathbb{E}_q p(x \mid z) + \ln \mathbb{E}_q q(z \mid x), \quad \langle b \rangle : N(z, \text{var}(z))$

28. Why do we use $-\ln p(x)$ instead of $\ln p(x)$?

A To give us a reward function (where higher is better) which is the convention in reinforcement learning.
B To give us a loss function (where lower is better), which is the convention in deep learning systems.✓
C To ensure that negative and positive errors don't cancel out against one another.
D To ensure that negative and positive errors do cancel out against one another.

Maximizing $\ln p_\theta(x)$ is a common way of framing the maximum likelihood objective (we want to choose the parameters that maximize the probability (density) of the data we take the logarithm to make the analysis easier or the search more numerically stable). Since deep learning frameworks require a loss to minimize, we minimize $-\ln p_\theta(x)$.

type: Naive Bayes

The following dataset represents a spam classification problem: we observe 8 emails and measure two binary features. The first is T if the word ”pill” occurs in the e-mail (F otherwise) and the second is T if the word “meeting” occurs.

“pill”	“meeting”	label
T	F	Spam
T	F	Spam
F	T	Spam
T	F	Spam
F	F	Ham
F	F	Ham
F	T	Ham
T	T	Ham

We build a naive Bayes classifier on this data, as described in the lecture. We estimate the class priors (p(Spam) and p(Ham)) from the data.

29. We observe one email that contains both words and one that contains neither. Which class does the classifier assign to each?

 A both words: Ham, neither word: Ham
 B both words: Ham, neither word: Spam
 C both words: Spam, neither word: Ham✓
 D both words: Spam, neither word: Spam

 It is extremely easy to make a mistake in these questions, and in this case all possible answers are included. Make sure to check your work carefully.

30. We observe an email e that contains the word “pill”, but not the word “meeting”. What probabilities does the classifier assign?

 A $p(\text{Ham} \mid e) = 9/11$, $p(\text{Spam} \mid e) = 2/11$
 B $p(\text{Ham} \mid e) = 2/11$, $p(\text{Spam} \mid e) = 9/11$✓
 C $p(\text{Ham} \mid e) = 9/32$, $p(\text{Spam} \mid e) = 2/32$
 D $p(\text{Ham} \mid e) = 2/32$, $p(\text{Spam} \mid e) = 9/32$

We add pseudo-observations to the data to deal with unseen emails. The pseudo-observations have the same weight as the normal ones. Compute how the judgments change.
NB: We only need to add enough observations to ensure that every value has been seen once per class per feature. So for Spam we add one observation with all F’s, one observation with all T’s and the same for Ham: four pseudo-observations in total.

“pill”	“meeting”	label
T	F	Spam
T	F	Spam
F	T	Spam
T	F	Spam
F	F	Ham
F	F	Ham
F	T	Ham
T	T	Ham
T	T	Spam
F	F	Spam
T	T	Ham
F	F	Ham

31. We observe one email that contains both words and one that contains neither. Which class does the *smoothed* classifier assign to each?
   A both words: Ham, neither word: Ham
   B both words: Ham, neither word: Spam
   C both words: Spam, neither word: Ham ✓
   D both words: Spam, neither word: Spam
   In a homework exercise, we would make sure that smoothing has an effect, to illustrate its purpose. In an exam, the results are not guaranteed to be so intuitive. Trust your calculations.

32. We observe an email that contains the word “pill”, but not the word “meeting”. What probabilities does the *smoothed* classifier assign?
   A $p(\text{Ham} \mid e) = 16/22$, $p(\text{Spam} \mid e) = 6/22$
   B $p(\text{Ham} \mid e) = 6/22$, $p(\text{Spam} \mid e) = 16/22$✓
   C $p(\text{Ham} \mid e) = 16/72$, $p(\text{Spam} \mid e) = 6/72$
   D $p(\text{Ham} \mid e) = 6/72$, $p(\text{Spam} \mid e) = 16/72$

type: the kernel trick

Consider the following kernel:

$$k(\mathbf{x}, \mathbf{y}) = (2 \cdot x^T y + 3)^2$$

We wouldn’t normally put constants in a kernel like this (since it doesn’t really affect the resulting model). They’re just there so we can check that you know how to work out the feature space.
We apply this kernel to two-dimensional vectors

$$\mathbf{x} = \begin{pmatrix}1\\2\end{pmatrix}, \mathbf{y} = \begin{pmatrix}3\\1\end{pmatrix} .$$

33. Which is correct?

A $k(x, y) = \begin{pmatrix} 81 \\ 49 \end{pmatrix}$

B $k(x, y) = \begin{pmatrix} 49 \\ 81 \end{pmatrix}$

C $k(x, y) = 169$ ✓

D $k(x, y) = 144$

The output of a kernel is *always* a scalar.

34. Assuming two-dimensional inputs, what is the explicit feature space x' for which k computes the dot product?

A $x' = \begin{pmatrix} 2\sqrt{2} & \cdot x_1{}^2 \\ 2\sqrt{2} & \cdot x_2{}^2 \\ 2 & \cdot x_1 x_2 \\ \sqrt{3} & \cdot x_1 \\ \sqrt{3} & \cdot x_2 \\ 3 & \end{pmatrix}$ **B** $x' = \begin{pmatrix} 8 & \cdot x_1{}^2 \\ 8 & \cdot x_2{}^2 \\ 4 & \cdot x_1 x_2 \\ 9 & \cdot x_1 \\ 9 & \cdot x_2 \\ 9 & \end{pmatrix}$

C ✓$x' = \begin{pmatrix} 2 & \cdot x_1{}^2 \\ 2 & \cdot x_2{}^2 \\ 2\sqrt{2} & \cdot x_1 x_2 \\ 2\sqrt{3} & \cdot x_1 \\ 2\sqrt{3} & \cdot x_2 \\ 3 & \end{pmatrix}$ **D** $x' = \begin{pmatrix} 4 & \cdot x_1{}^2 \\ 4 & \cdot x_2{}^2 \\ 8 & \cdot x_1 x_2 \\ 12 & \cdot x_1 \\ 12 & \cdot x_2 \\ 9 & \end{pmatrix}$

This question requires you to work out a three-term square $(a + b + c)^2$. If you can't remember the formula, just work it out from the two-term one:

$$\begin{aligned}(a + b + c)^2 &= ((a + b) + c)^2 \\ &= (a + b)^2 + 2(a + b)c + c^2 \\ &= (a + b)^2 + 2ac + 2bc + c^2 \\ &= a^2 + 2ab + b^2 + 2ac + 2bc + c^2 \\ &= a^2 + b^2 + c^2 + 2ab + 2ac + 2bc\ .\end{aligned}$$

We would like a kernel for the feature space

$$x = \begin{pmatrix} c_1 & \cdot & x_1{}^2 \\ c_2 & \cdot & x_2{}^2 \\ c_3 & \cdot & x_1 x_2 \\ c_4 & \cdot & x_1 \\ c_5 & \cdot & x_2 \\ c_6 & & \end{pmatrix},$$

where $c_1, \ldots, c_6$ are constants.

35. Which kernel does the job?

    A $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y})^2$
    B $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)$
    C $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^2$✓
    D $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^3$

    The basic strategy here is just to work out all four feature spaces. However, you can make an educated guess where to start. Remember that the $+1$ helps us to get a constant term in the feature space and to retain the original features (so it can't be A). Furthermore, raising to the power of n gives us all n-way dot products. This answer includes the 2-way dot products, but not the 3-way dot-products, so its likely answer C.

type: Matrix backpropagation

We have a module f in an automatic differentiation (AD) system (as discussed in the lectures) which computes the following function (its *forward pass*):

$$f_{w}(x) = x^2 w$$

where x is a scalar and w is a vector. Note that this makes f a function from a scalar to a vector.

We will assume that the AD engine will work out the downstream derivatives for us. Given these, we will need to compute the derivatives over the argument x and over the parameters w. Let the vector $\mathbf{f}$ represent the output of our function.

36. What is the local scalar derivative $\partial f_k / \partial x$?

    A $\partial f_k / \partial x = 2x\, w_k$ ✓
    B $\partial f_k / \partial x = x\, w_k$
    C $\partial f_k / \partial x = 2x^2\, w_k$
    D $\partial f_k / \partial x = x^2\, w_k$

37. Our AD engine provides a vector $\mathbf{d}$ such that $d_i = \partial L / \partial f_i$. What should the module return as the gradients of x?

    A $x^2\mathbf{d}$
    B $2x\mathbf{d}$
    C $2x\mathbf{d}^T w$
    D $x^2\mathbf{d}^T w$

Remember, we're looking to compute $\partial L/\partial x$.

$$\begin{aligned}
\frac{\partial L}{\partial x} &= \sum_k \frac{\partial L}{\partial f_k}\frac{f_k}{\partial x} && \text{apply mv chain rule}\\
&= \sum_k \frac{\partial L}{\partial f_k} 2x w_k && \text{from the previous exercise}\\
&= 2x \sum_k \frac{\partial L}{\partial f_k} w_k \\
&= 2x \sum_k d_k w_k && \text{note that the sum is a dot product, so...}\\
&= 2x\mathbf{d}^T \boldsymbol{w}
\end{aligned}$$

38. What is the local scalar derivative $\partial f_k/\partial w_i$?

A $\partial f_k/\partial w_i = x^2$
B $\partial f_k/\partial w_i = 0$
C $\partial f_k/\partial w_i = x^2$ if $k = i$, 0 otherwise✓
D $\partial f_k/\partial w_i = 0$ if $k = i$, x^2 otherwise

39. Given $\mathbf{d}$ with $d_i = \partial L/\partial d_i$, what should the module return as the gradients of $\boldsymbol{w}$?

A $x^2\mathbf{d}$
B $2x\mathbf{d}$
C $2x\mathbf{d}^T\boldsymbol{w}$
D $x^2\mathbf{d}^T\boldsymbol{w}$

Remember, we're looking to compute the vector $\boldsymbol{w}'$ where $w'_i = \partial L/\partial w_i$.

$$\begin{aligned}
\frac{\partial L}{\partial w_i} &= \sum_k \frac{\partial L}{\partial f_k}\frac{f_k}{\partial w_i} && \text{apply mv chain rule}\\
&= \frac{\partial L}{\partial f_i}\frac{f_i}{\partial w_i} = \frac{\partial L}{\partial f_i} x^2 && \text{from the previous exercise}\\
&= x^2 d_i
\end{aligned}$$

Which means that $\boldsymbol{w}' = x^2\mathbf{d}$.

type: Lagrange multipliers

Consider the following optimization problem:

$$\begin{gathered}
\arg\min_{x,y} ax + by \\
\text{such that } \; x^2 + y^2 = 1,
\end{gathered}$$

where a, and b are non-zero constants.

40. Which is the correct Lagrangian for this problem?

A $L(x, y) = ax + by$
B $L(x, y) = 2x + 2y$
C $L(x, y, \alpha) = ax + by + \alpha x^2 + \alpha y^2 - \alpha$ ✓
D $L(x, y, \alpha) = a + b + 2\alpha x + 2\alpha y - \alpha$

41. Which is correct?

A $\partial L/\partial x = b + 2\alpha y$
B $\partial L/\partial x = b + 2\alpha y^2$
C $\partial L/\partial y = b + 2\alpha y$ ✓
D $\partial L/\partial y = b + 2\alpha y^2$

42. Let $n = \sqrt{a^2 + b^2}$ What are (all) the solutions?

A $x = a/n, y = -b/n$
B $x = a/n, y = -b/n$ and $x = -a/n, y = b/n$
C $x = \sqrt{a}/n, y = \sqrt{b}/n$ and $x = a/n, y = b/n$
D $x = a/n, y = b/n$ and $x = -a/n, y = -b/n$ ✓

Note that this is a different way of proving what we learned in lecture 2: the unit vector (x, y) that maximizes the dot product with vector (a, b) is the one that points in the same direction (that is, the solution is (a, b) normalized to a unit vector).

The second solution is the unit vector that points in the opposite direction; the one that minimizes the dot product (remember, we find all optima, not just the maxima).

If you want extra practice, try the same thing in three dimensions and then in n dimensions.

43. Could we also find these solutions by using standard gradient descent with the gradient of the Lagrangian?

A Yes, this is how SVMs with kernels are commonly solved.
B Yes, this is how SVMs are commonly solved, but in that case the kernel trick cannot be applied.
C No, the solutions are saddle-points, not optima. ✓
D No, the gradient is not zero at the solutions.

NB: There are some ways to apply gradient descent if the roots of the derivative of L can't be found analytically, but standard gradient descent won't work.