Machine Learning 2020

Practice Exam A

February 21, 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.
- The version of this document with answers contains many explanations and additional tips, so be sure to read that one too.

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.

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.

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.

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.
C To use bootstrap sampling to gauge the variance of the model.
D To use a boosted ensemble, to reduce the variance of the model, and with it, the probability of random effects.

This is discussed in lecture 3. It is a crucial part of any machine learning project.

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

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.

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.

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.

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.

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:

$$\frac{\partial \frac{1}{2}\sum_i (y_i - t_i)^2}{\partial b} = \frac{1}{2}\frac{\partial \sum_i (y_i - t_i)^2}{\partial b} \tag{1}$$

$$= \frac{1}{2}\sum_i \frac{\partial (y_i - t_i)^2}{\partial b} \tag{2}$$

$$= \frac{1}{2}\sum_i \frac{\partial (y_i - t_i)^2}{\partial (y_i - t_i)}\frac{\partial (y_i - t_i)}{\partial b} \tag{3}$$

$$= \frac{1}{2}\sum_i 2(y_i - t_i)\frac{\partial (y_i - t_i)}{\partial b} \tag{4}$$

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

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** $\sum_i x_i \left(-v{x_i}^2 + wx_i + b - t_i\right)$

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

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

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

14. How many ranking errors does the classifier make?
    A None
    B 1
    C 2
    D 3

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** $\frac{6}{16}$

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	1/4	0
b	1/4	1/4
c	1/4	1/4
d	1/4	1/2

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 $H(q, p) = 2.5$

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.

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:

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

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 $\cos(c)\,[b\cos(x) - a\sin(x)]$

type: Decision trees

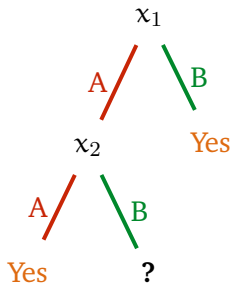
Consider the following task. The aim is to predict the class y from the binary features x_1, x_2, x_3 and x_4.

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

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

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

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.

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

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

    A $\langle a \rangle : \mathbb{E}_q \ln p(\mathbf{z} \mid \mathbf{x},\theta)$, $\langle b \rangle : \ln p(\mathbf{x} \mid \theta)$
    B $\langle a \rangle : \mathbb{E}_q \ln p(\mathbf{z} \mid \mathbf{x},\theta)$, $\langle b \rangle : \mathbb{E}_q \ln p(\mathbf{x})$
    C $\langle a \rangle : \mathbb{E}_q \ln p(\mathbf{z},\mathbf{x} \mid \theta)$, $\langle b \rangle : \ln p(\mathbf{x} \mid \theta)$
    D $\langle a \rangle : \mathbb{E}_q \ln p(\mathbf{z},\mathbf{x} \mid \theta)$, $\langle b \rangle : \mathbb{E}_q \ln p(\mathbf{x})$

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 $KL(q,p)$.
    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 $L(q,\theta) + KL(q,p)$ we rewrite it to $\ln p(\mathbf{x} \mid \theta)$ and use random search to find the optimal θ.
    D To maximize $L(q,\theta) + KL(q,p)$ 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.

$$
\begin{aligned}
-\ln p_\theta(x) \geqslant -L(q,\theta) &= \langle a \rangle \\
&= -\mathbb{E}_q \ln p(x \mid z) - \mathbb{E}_q \ln p(z) + \mathbb{E}_q \ln q(z \mid x) \\
&= -\mathbb{E}_q \ln p(x \mid z) + \mathbb{E}_q \ln \frac{q(z \mid x)}{p(z)} \\
&= -\mathbb{E}_q \ln p(x \mid z) - \mathbb{E}_q \ln \frac{p(z)}{q(z \mid x)} \\
&= -\mathbb{E}_q \ln p(x \mid z) + \mathrm{KL}\left(q(z \mid x), p(z)\right) \\
&= -\mathbb{E}_q \ln p(x \mid z) + \mathrm{KL}(q(z \mid x), \langle b \rangle)
\end{aligned}
$$

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 $\langle a \rangle : \quad \ln \mathbb{E}_q p(x \mid z) - \ln \mathbb{E}_q q(z \mid x), \quad \langle b \rangle : N(z, \mathrm{var}(z))$
D $\langle a \rangle : -\ln \mathbb{E}_q p(x \mid z) + \ln \mathbb{E}_q q(z \mid x), \quad \langle b \rangle : N(z, \mathrm{var}(z))$

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.

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 ($\mathrm{p}(\text{Spam})$ and $\mathrm{p}(\text{Ham})$) 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

30. We observe an email e that contains the word "pill", but not the word "meeting". What probabilities does the classifier assign?
    A $\mathrm{p}(\text{Ham} \mid e) = 9/11$, $\mathrm{p}(\text{Spam} \mid e) = 2/11$
    B $\mathrm{p}(\text{Ham} \mid e) = 2/11$, $\mathrm{p}(\text{Spam} \mid e) = 9/11$
    C $\mathrm{p}(\text{Ham} \mid e) = 9/32$, $\mathrm{p}(\text{Spam} \mid e) = 2/32$
    D $\mathrm{p}(\text{Ham} \mid e) = 2/32$, $\mathrm{p}(\text{Spam} \mid e) = 9/32$

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.

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

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

type: the kernel trick

Consider the following kernel:

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

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 $\mathrm{k}(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} 81 \\ 49 \end{pmatrix}$
B $\mathrm{k}(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} 49 \\ 81 \end{pmatrix}$
C $\mathrm{k}(\mathbf{x}, \mathbf{y}) = 169$
D $\mathrm{k}(\mathbf{x}, \mathbf{y}) = 144$

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

A $\mathbf{x}' = \begin{pmatrix} 2\sqrt{2} & \cdot \mathrm{x}_1{}^2 \\ 2\sqrt{2} & \cdot \mathrm{x}_2{}^2 \\ 2 & \cdot \mathrm{x}_1\mathrm{x}_2 \\ \sqrt{3} & \cdot \mathrm{x}_1 \\ \sqrt{3} & \cdot \mathrm{x}_2 \\ 3 & \end{pmatrix}$ **B** $\mathbf{x}' = \begin{pmatrix} 8 & \cdot \mathrm{x}_1{}^2 \\ 8 & \cdot \mathrm{x}_2{}^2 \\ 4 & \cdot \mathrm{x}_1\mathrm{x}_2 \\ 9 & \cdot \mathrm{x}_1 \\ 9 & \cdot \mathrm{x}_2 \\ 9 & \end{pmatrix}$

C $\mathbf{x}' = \begin{pmatrix} 2 & \cdot \mathrm{x}_1{}^2 \\ 2 & \cdot \mathrm{x}_2{}^2 \\ 2\sqrt{2} & \cdot \mathrm{x}_1\mathrm{x}_2 \\ 2\sqrt{3} & \cdot \mathrm{x}_1 \\ 2\sqrt{3} & \cdot \mathrm{x}_2 \\ 3 & \end{pmatrix}$ **D** $\mathbf{x}' = \begin{pmatrix} 4 & \cdot \mathrm{x}_1{}^2 \\ 4 & \cdot \mathrm{x}_2{}^2 \\ 8 & \cdot \mathrm{x}_1\mathrm{x}_2 \\ 12 & \cdot \mathrm{x}_1 \\ 12 & \cdot \mathrm{x}_2 \\ 9 & \end{pmatrix}$

We would like a kernel for the feature space

$$\mathbf{x} = \begin{pmatrix} \mathrm{c}_1 & \cdot & \mathrm{x}_1{}^2 \\ \mathrm{c}_2 & \cdot & \mathrm{x}_2{}^2 \\ \mathrm{c}_3 & \cdot & \mathrm{x}_1\mathrm{x}_2 \\ \mathrm{c}_4 & \cdot & \mathrm{x}_1 \\ \mathrm{c}_5 & \cdot & \mathrm{x}_2 \\ \mathrm{c}_6 & & \end{pmatrix},$$

where $\mathrm{c}_1, \ldots, \mathrm{c}_6$ are constants.

35. Which kernel does the job?

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

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** $x^2 \mathbf{d}^T w$

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

type: Lagrange multipliers

Consider the following optimization problem:

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

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 $x = -a/n, y = -b/n$

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.