For these exercises, it will be helpful to review the notes on Linear Classifiers and the Perceptron. You may also find it helpful to write some test code with a local python installation or in a google colab notebook.

1) Classification

Consider a linear classifier through the origin in 4 dimensions, specified by

$$\theta = (1, -1, 2, -3)$$

Which of the following points x are classified as positive, i.e. $h(x; \theta) = +1$?

1. $(1, -1, 2, -3)$
2. $(1, 2, 3, 4)$
3. $(-1, -1, -1, -1)$
4. $(1, 1, 1, 1)$

Enter a Python list with a subset of the numbers 1, 2, 3, 4: [1, 3]

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

2) Classifier vs Hyperplane

Consider another parameter vector

$$\theta' = (-1, 1, -2, 3)$$

Ex2a

Does θ' represent the same hyperplane as θ does? yes

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

Ex2b

Does θ' represent the same classifier as θ does? no

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

3) Linearly Separable Training

As discussed in lecture and in the lecture notes, note that $\mathcal{E}_n(\theta, \theta_0)$ refers to the training error of the linear classifier specified by θ, θ_0, and $\mathcal{E}(\theta, \theta_0)$ refers to its test error. What does the fact that the training data are *linearly separable* imply?

Select "yes" or "no" for each of the following statements:

Ex3a

There must exist θ, θ_0 such that $\mathcal{E}(\theta, \theta_0) = 0$ no

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

Ex3b

There must exist θ, θ_0 such that $\mathcal{E}_n(\theta, \theta_0) = 0$ yes

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

Ex3c

A separator with 0 training error exists yes

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

Ex3d

A separator with 0 testing error exists, for all possible test sets no

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

Ex3e

The perceptron algorithm will find θ, θ_0 such that $\mathcal{E}_n(\theta, \theta_0) = 0$ yes

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

4) Separable Through Origin?

Provide two points, (x_0, x_1) and (y_0, y_1) in two dimensions that are linearly separable but not linearly separable through the origin. If you get stuck try drawing a picture and review the notes on offsets.

Enter a Python list with two entries of the form `[[ x0, x1], label]` where label is `1` or `-1`. (So each entry represents a point with 2 dimensions and its label) `[ [ [0,0], -1], [ [1, 1], 1] ]`

100.00%

You have infinitely many submissions remaining.

Solution: `[[[1, 1], 1], [[2, 2], -1]]`

Explanation:

There are many possible answers for this question. In the provided solution, [[[1, 1], 1], [[2, 2], -1]], the points are linearly separable, but not through the origin -- try drawing the points.

For these exercises, it will be helpful to review the notes on Linear Classifiers

1) Feature representation

For the following feature, pick what might be the best encoding for linear classification. The assumption is that there are other features in the data set.

The point of this question is to think about alternatives; there are many options, many not mentioned here.

Car make, e.g. Chevy, Ford, Toyota, VW, for predicting gas mileage (lo, hi).

4 unary features (one-hot): 1000, 0100, 0010, 0001

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

2) Feature mapping

Consider the following, one-dimensional, data set. It is not linearly separable in its original form.

1. $x^{(1)} = -1,\ y^{(1)} = +1$
2. $x^{(2)} = 0,\ y^{(2)} = -1$
3. $x^{(3)} = 1,\ y^{(3)} = +1$

Ex2.a.: Which of these feature transformations leads to a separable problem?

1. $\phi(x) = 0.5 * x$
2. $\phi(x) = |x|$
3. $\phi(x) = x^3$
4. $\phi(x) = x^4$
5. $\phi(x) = x^{2k}$ for any positive integer k

Enter a Python list with a subset of the numbers 1, 2, 3, 4, 5. [2, 4, 5]

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

Ex2.b.: Your friend Kernelius uses feature transformation $\phi(x) = (x, x^2)$ on the data above. In the new space, the linear classifier with $\theta = (0, 1)$ and $\theta_0 = -0.25$ achieves perfect accuracy. What points from the original space $\mathbb{R}$ map to this linear classifier in $\mathbb{R}^2$? (It may be helpful to find the equation of the separator.)

Enter a Python list with all values of x which constitute this separator.

[-0.5, 0.5]

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

These exercises will prepare you for understanding how to maximize margins, as discussed in the lecture notes. You may want to review the definition of the margin γ.

1) Margin definition

Recall that the signed distance to a point x from a hyperplane θ, θ_0 is $sd(x, \theta, \theta_0) = \frac{\theta^T x + \theta_0}{\|\theta\|}$.

Ex1a:

You start with a hyperplane θ, θ_0 and a point x. Suppose a new separator is given, where $\hat{\theta} = -\theta$ and $\hat{\theta}_0 = -\theta_0$.

Which of the following is true? the signed distance changes sign but not magnitude

Submit View Answer **100.00%**

You have 1 submission remaining.

Ex1b:

You start with a hyperplane θ, θ_0 and a point x. Suppose a new separator is given, where $\hat{\theta} = \theta$ and $\hat{\theta}_0 = -\theta_0$.

Which of the following is true: both the sign and the magnitude may change

Submit View Answer **100.00%**

You have 2 submissions remaining.

Ex1c:

The margin of labeled point x, y with respect to separator θ, θ_0 is:

$$\gamma(x, y, \theta, \theta_0) = \frac{y(\theta^T x + \theta_0)}{\|\theta\|}$$

Let `sd` stand for $sd(x, \theta, \theta_0)$, the signed distance from the separator to x. Define the margin in terms of `sd` and `y`, the label of x. Note that both of these are scalars. Provide an expression in Python syntax.

$\gamma(x, y, \theta, \theta_0) =$ y * sd

Check Syntax Submit View Answer **100.00%**

You have infinitely many submissions remaining.

Ex1d:

What is the sign of the signed distance when the prediction is incorrect?

Which of the following is true: could be either

View Answer **100.00%**

You have 0 submissions remaining.

Ex1e:

What is the sign of the margin when the prediction is incorrect?

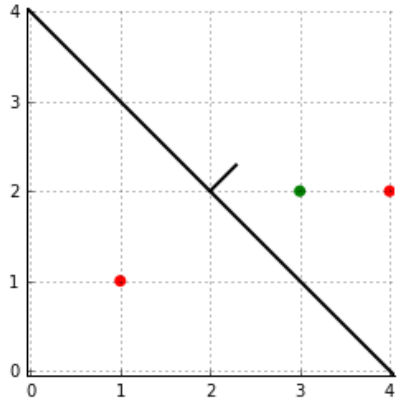
Which of the following is true: negative

Submit View Answer **100.00%**

You have 1 submission remaining.

2) Margin practice

What are the margins of the labeled points (x,y) = ((3, 2), +1), ((1, 1), -1), and ((4, 2), -1) with respect to the separator defined by $\theta = (1, 1), \theta_0 = -4$? The situation is illustrated in the figure below.



Enter the three margins in order as a Python list of three numbers. Note that you can enter sqrt(x) as x**0.5 in Python. [1/(2)**0.5, 2/(2)**0.5, -2/(2)**0.5]

Submit View Answer **100.00%**

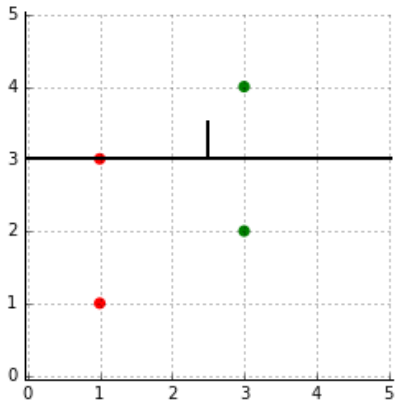
You have infinitely many submissions remaining.

3) Max Margin Separator

Consider the four points and separator:

```
data = np.array([[1, 1, 3, 3],[3, 1, 4, 2]])
labels = np.array([[-1, -1, 1, 1]])
th = np.array([[0, 1]]).T
th0 = -3
```

The situation is shown below:



Ex3a:

Enter the four margins in order as a Python list of four numbers.

[0, 2, 1, -1]

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

Ex3b:

A maximum margin separator is a separator that maximizes the minimum margin between that separator and all points in the dataset.

Enter θ and θ_0 for a maximum margin separator as a Python list of three numbers.

[1, 0, -2]

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

Ex3c:

If you scaled this separator by a positive constant k (i.e., replace θ by $k\theta$, and θ_0 by $k\theta_0$), would it still be a maximum margin separator? Yes

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

For these exercises, it will be helpful to review the notes on Regression.

1) Intro to linear regression

So far, we have been looking at classification, where predictors are of the form

$$y = \text{sign}(\theta^T x + \theta_0)$$

making a binary classification as to whether example x belongs to the positive or negative class of examples.

In many problems, we want to predict a real value, such as the actual gas mileage of a car, or the concentration of some chemical. Luckily, we can use most of a mechanism we have already spent building up, and make predictors of the form:

$$y = \theta^T x + \theta_0.$$

This is called a *linear regression* model.

We would like to learn a linear regression model from examples. Assume X is a d by n array (as before) but that Y is a 1 by n array of floating-point numbers (rather than +1 or -1). Given data (X, Y) we need to find θ, θ_0 that does a good job of making predictions on new data drawn from the same source.

We will approach this problem by formulating an objective function. There are many possible reasonable objective functions that implicitly make slightly different assumptions about the data, but they all typically have the form:

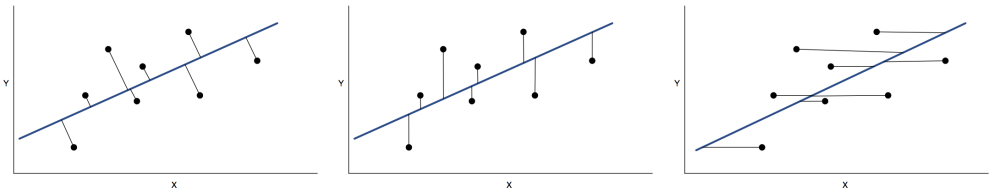
$$J(\theta, \theta_0) = \frac{1}{n} \sum_{i=1}^{n} L(x^{(i)}, y^{(i)}, \theta, \theta_0) + \lambda R(\theta, \theta_0).$$

For regression, we most frequently use *squared loss*, in which

$$L_s(x, y, \theta, \theta_0) = (y - \theta^T x - \theta_0)^2.$$

The term with $R(\theta, \theta_0)$ is termed the *regularizer*, and penalizes more complex predictors. We will explore different choices of regularizer later in this set of exercises.

Ex1.1: Which of the following pictures illustrates the squared loss metric? Assume that the dark line is described by θ, θ_0, the black dots are the (x, y) data, and the light lines indicate the errors.



A B C

Select the picture which best illustrates the squared loss metric: B

100.00%

You have 0 submissions remaining.

Solution: B

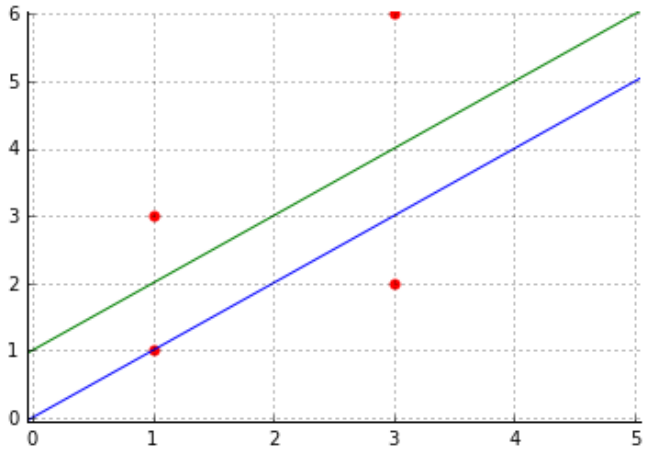
Explanation:

Squared loss measures the squared distance between the actual labels y and the predicted labels $\hat{y} = \theta^T x + \theta_0$. Therefore, the picture should depict distances in only y, which corresponds to B.

Note that A is the perpendicular distance from point to separator, not the difference between actual and predicted y.

2) Linear Regression

Consider the data set and regression lines in the plot below.



- The equation of the blue (lower) line is: $y = x$
- The equation of the green (upper) line is: $y = x + 1$
- The data points (in x, y pairs) are: $((1, 3), (1, 1), (3, 2), (3, 6))$

Ex2.1: What is the squared error of each of the points with respect to the **blue** line?

Provide a Python list of four numbers (in the order of the points given above).

[4, 0, 1, 9]

Submit View Answer **100.00%**

You have 18 submissions remaining.

The gradient of the mean squared error regression criterion has the form of a sum over contributions from individual points. The formula for the gradient of the squared error with respect to parameters of a line, θ, θ_0 for a single point (x, y) (without regularizer), is:

$$\left(-2(y-\theta^T x-\theta_0)x, \quad -2(y-\theta^T x-\theta_0)\right).$$

Ex2.2: What is the gradient contribution from each point to the parameters of the blue (lower) line?

Provide a list of four pairs of numbers (as tuples, in the order of the points given above).

[(-4, -4), (0, 0), (6, 2), (-18, -6)]

Submit View Answer **100.00%**

You have 19 submissions remaining.

Ex2.3: What is the squared error of each of the points with respect to the green line?

Provide a list of four numbers (in the order of the points given above).

[1, 1, 4, 4]

Submit View Answer **100.00%**

You have 19 submissions remaining.

Ex2.4: What is the gradient contribution from each point to the parameters of the green line?

Provide a list of four pairs of numbers (as tuples, in the order of the points given above).

[(-2, -2), (2, 2), (12, 4), (-12, -4)]

Submit View Answer **100.00%**

You have 17 submissions remaining.

Ex2.5: Mark all of the following that are true:

- [ ] The blue line minimizes mean squared error
- [x] The green line minimizes mean squared error
- [ ] The mean squared error from all the points to the blue line is 0
- [ ] The mean squared error from all the points to the green line is 0
- [ ] The sum of the gradient contributions from all the points for the blue line is 0
- [x] The sum of the gradient contributions from all the points for the green line is 0
- [ ] Neither line minimizes mean squared error
- [ ] It is impossible to minimize mean squared error
- [ ] Both lines minimize mean squared error

Submit View Answer **100.00%**

You have 5 submissions remaining.

3) Ridge regression

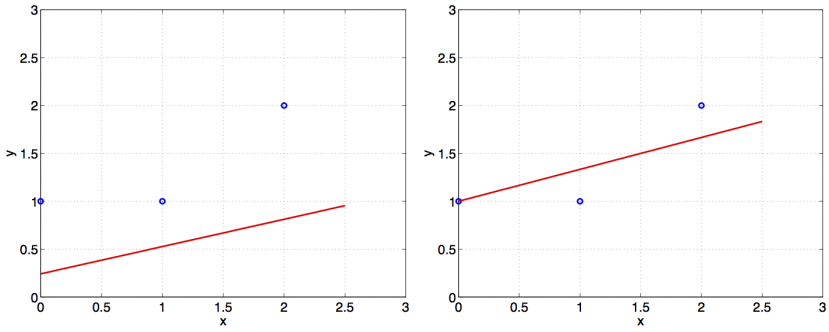
It may be help to review the notes on regularization.

If we add a squared-norm regularizer to the empirical risk, we get the so-called *ridge regression* objective:

$$J_{ridge}(\theta, \theta_0) = \frac{1}{n}\sum_{i=1}^{n} L_s(x^{(i)}, y^{(i)}, \theta, \theta_0) + \lambda||\theta||^2.$$

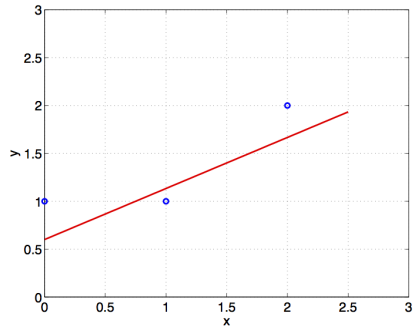
It's a bit tricky to solve this analytically, because you can see that the penalty is on θ but not on θ_0.

The figures below plot linear regression results on the basis of only three data points $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)})$. We used various types of regularization to obtain the plots (see below) but got confused about which plot corresponds to which regularization method. Please assign each plot to one (and only one) of the following regularization methods.

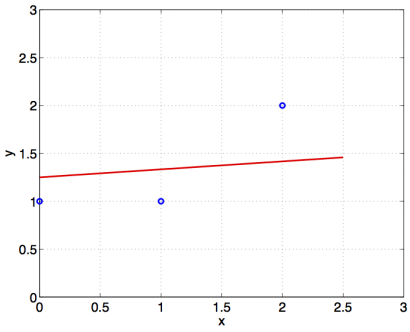


A

B



C



D

Ex3.1:

$\frac{1}{3}\sum_{i=1}^{3}(y^i - wx^i - w_0)^2 + \lambda w^2$ where $\lambda = 1$ B

Submit View Answer **100.00%**

You have 2 submissions remaining.

Ex3.2:

$\frac{1}{3}\sum_{i=1}^{3}(y^i - wx^i - w_0)^2 + \lambda w^2$ where $\lambda = 10$ D

Submit View Answer **100.00%**

You have 2 submissions remaining.

Ex3.3:

$\frac{1}{3}\sum_{i=1}^{3}(y^i - wx^i - w_0)^2 + \lambda(w^2 + w_0^2)$ where $\lambda = 1$ C

Submit View Answer **100.00%**

You have 1 submission remaining.

Ex3.4:

$\frac{1}{3}\sum_{i=1}^{3}(y^i - wx^i - w_0)^2 + \lambda(w^2 + w_0^2)$ where $\lambda = 10$ A

Submit View Answer **100.00%**

You have 1 submission remaining.

For these exercises, you should review the notes on Neural Networks.

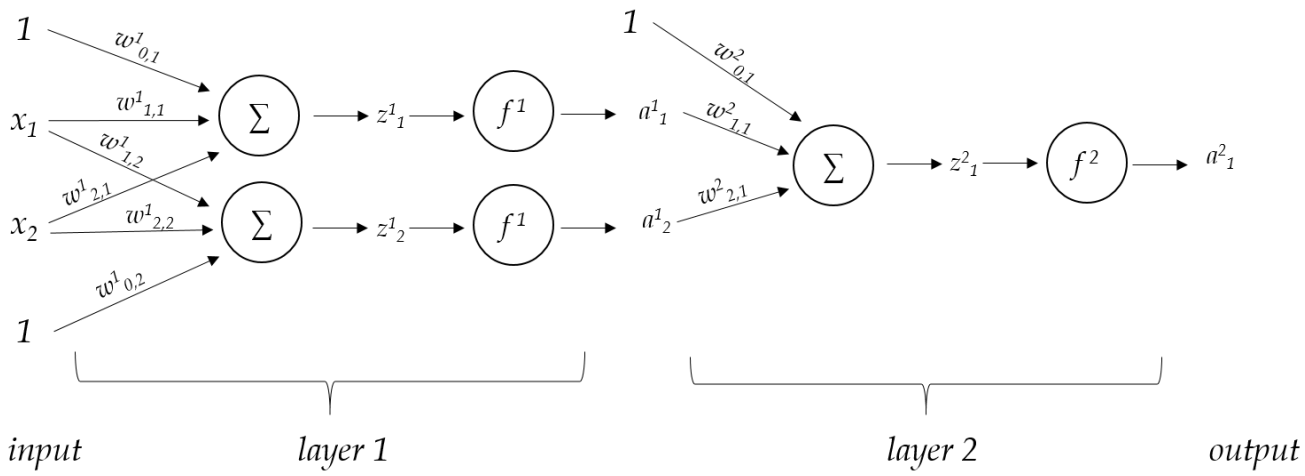
1) Prediction

Consider the following data set:

```
X = np.array([[0, 1, 2],
              [0, 1, 2]])
Y = np.array([[0, 1, 0]])
```

The columns of X and Y are the data points and corresponding labels.

We will be looking at the behavior of the following simple two-layer network:



Assume that within each layer, each unit has the step activation function $f(z)$ given by

$$f(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}.$$

Let the weights in the first layer (layer 1) be:

- $w^1_{0,1} = -0.5, w^1_{1,1} = 1, w^1_{2,1} = 0$
- $w^1_{0,2} = 1.5, w^1_{1,2} = -1, w^1_{2,2} = 0$

1A) Enter a matrix Z where each column represents the outputs of the hidden units ($f^1(z^1_1)$ and $f^1(z^1_2)$) for each of the input vectors in `X`.

Enter a Python list of lists [[a,b,c],[d,e,f]], each list is a row of the matrix.

[[0, 1, 1], [1, 1, 0]]

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

1B) Pick weights for the second layer $w^2_{0,1}, w^2_{1,1}, w^2_{2,1}$ so that the desired outputs are predicted correctly.

Enter a Python list of 3 numbers $[w^2_{0,1}, w^2_{1,1}, w^2_{2,1}]$ [-1.5, 1, 1]

Submit View Answer **100.00%**

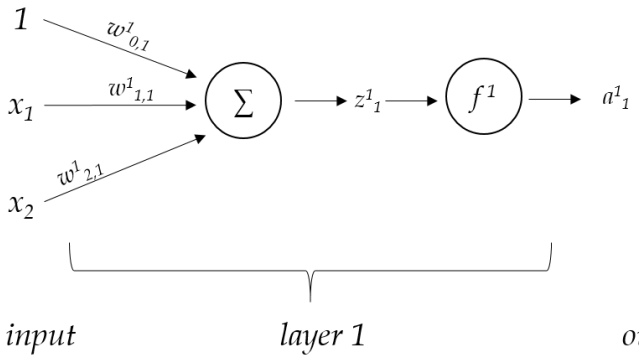
You have infinitely many submissions remaining.

2) Training

Now, we will consider the classification of a different set of `X` and `Y` data, with a different single layer network having the following structure and activation function:

$$z = w^1_{1,1}x_1 + w^1_{2,1}x_2 + w^1_{0,1}$$

In this network we have $f^1(z) = z$, so our output $a^1_1 = z^1_1$.



Assume the initial weights are $w^1_{0,1} = 1, w^1_{1,1} = 1, w^1_{2,1} = 1$, and the step size is 0.5 (not usually a good idea, but okay for now).

The current training example is $x^{(i)} = [1, 2]^T$, $y^{(i)} = -1$.

2A) What is the output value a^1_1, given current input $x^{(i)}$ and the current weights?

Enter a number 4

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

2B) What will the values of weights $w^1_{0,1}, w^1_{1,1}, w^1_{2,1}$ be after one step of stochastic gradient descent at the given training example $x^{(i)} = [1, 2]^T$, $y^{(i)} = -1$ using our definition of hinge loss $L_h(v) = max(0, 1 - v)$?

Enter a Python list of 3 numbers $[w^1_{0,1}, w^1_{1,1}, w^1_{2,1}]$ (to 3 decimal places)

[0.5, 0.5, 0]

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

2C) What would the output value a^1_1 be, for this same input x, with these new weights?

Enter a number 1

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

2D) What would happen to the v_i if we did another SGD update, for that same point, with step size 0.5, as before?

Enter a Python list of 3 numbers $[w_{0,1}^1, w_{1,1}^1, w_{2,1}^1]$ [0, 0, -1]

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

2E) Now what would the output be?

Enter a number -2

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

2F) What if we do one more update, for that same point?

Enter a Python list of 3 numbers $[w_{0,1}^1, w_{1,1}^1, w_{2,1}^1]$ [0, 0, -1]

100.00%

You have infinitely many submissions remaining.

Solution: `[0.0, 0.0, -1.0]`

Explanation:

Nothing happens; the derivatives are zero since the hinge loss is zero.

For these exercises, you should review the notes on Convolutional Neural Networks.

1) Weights

1.A) In a fully-connected feedforward network, the number of weights (including biases) between two adjacent layers each having 100 units is:

Enter a number: 10100

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

Consider a 1D CNN consisting of an input layer with 100 units, 10 filters each with 5 inputs (i.e., each filter is size 5) having a stride of 1, that produces an output layer (or feature map). Assume zero padding on the borders when applying the filters.

1.B) How many units are there in the output layer (feature map)?

Enter a number: 1000

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

1.C) How many weights (including bias) are needed to specify the output layer (feature map)?

Enter a number: 60

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

Now consider a 1D CNN with an input layer having 100 units, followed by a max pooling layer with a pooling filter of size 3 and a stride of 2, producing the output layer.

1.D) How many units are there in the output layer?

Enter a number: 50

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

1.E) How many weights (including bias) are needed to specify the max pooling output layer?

Enter a number: 0

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

2) Pooling

When processing an image we ultimately want to convert the input image into a smaller set of activations that we can feed into a fully-connected net that will produce the desired outputs, such as detecting the presence of a cat in the image irrespective of where the cat might be in that image. If one of the detectors in a previous layer had a strong output, we want to preserve that information, no matter where it happened in the input. This is the role of a *max pooling* layer; it is usually chosen to reduce the size of the output layer by some factor.

With a 1D input, given a "max filter" of size 3 and a stride of 2, what is the ratio (input layer size)/(output layer size)? Assume that the input layer size is much larger than the filter size and stride.

Enter a number: 2

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

3) CNN concepts

Here are the the building blocks we will use to build a CNN:

A. Convolutional layers
B. Max pooling layers
C. Fully connected layers

Assume that the task of our CNN is to detect a cat in an image.

3.A) Which of these blocks allows the network to be less sensitive to the exact locations of different cat parts?

Choose one: B

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

3.B) Which of these blocks allows the network to combine all cat features for high-level reasoning?

Choose one: C

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

3.C) Which of these blocks are responsible for detecting low and mid-level features of a cat in an image?

Choose one: A

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

For these exercises, you should read the notes on State Machines and Markov Decision Processes.

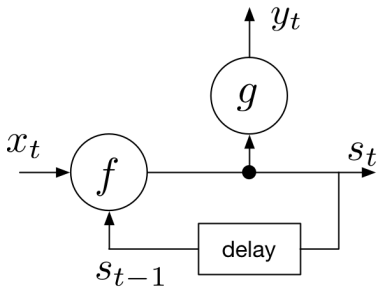
1) State Machines

State machines are a powerful and pervasive concept in information processing, playing important roles in computer science, signal processing, and control. Fundamentally, you can think of a state machine as a module that maps a sequence of input values $x_1, \ldots, x_T$ (they could be numbers, words, graphs, or anything) into a sequence of output values (also of any type) $y_1, \ldots, y_T$.

In the simplest case, we might have a simple functional dependence, so that $y_t = h(x_t)$, but more generally, we are interested in the case where y_t might potentially depend on all previous input values, so $y_t = h(x_1, \ldots, x_t)$. Generally, describing a functional dependency like this is very difficult, because the function h seems to need a different number of arguments on each time step. However, we can describe a very large class of h functions compactly, as a recursive combination of two functions, f and g, with an intermediate *state* value capturing the information that flows between applications of these functions. Mathematically, we have

$$
\begin{aligned}
y_t &= g(s_t) \\
s_t &= f(s_{t-1}, x_t) \\
s_0 &= 0
\end{aligned}
$$

Here is a figure illustrating the process: you can think of it as a circuit with the state fed back into f, but with a one-step delay.



If f and g are linear functions, then this is an LTI (linear time-invariant) system, a popular model in control systems analysis. In general, an LTI system can depend on a finite number of previous input values and a finite number of previous output values; we can model that dependence by allowing s_t to be a vector, storing as many of these values as we'd like.

If x_t, y_t, and s_t are all elements of finite sets, then this describes an FSM (finite-state machine), a popular model in computer science and some kinds of discrete control.

For each of the following state machines, provide the output sequence $[y_1, y_2, \ldots, y_T]$ given the input sequence $[x_1, x_2, \ldots, x_T]$:

1A)

```
s_0 = 0
f(s, x_i) = max(s, x_i)
g(s) = s * 2
Input: [0, 1, 2, 1]
```

Enter a Python list of four numbers: [0, 2, 4, 4]

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

1B)

```
s_0 = (0, 0)
f(s, x_i) = (s[0] + x_i, s[1] + 1)
g(s) = s[0] / s[1]
Input: [0, 1, 2, 1]
```

Enter a Python list of four numbers: [0.0, 0.5, 1.0, 1.0]

Submit View Answer **100.00%**

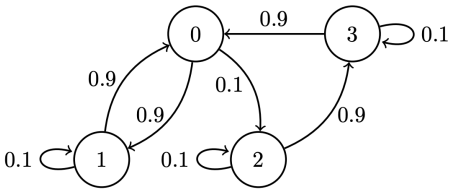
You have infinitely many submissions remaining.

2) Tiny Policy Evaluation

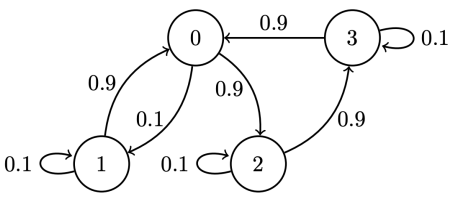
Consider an MDP with states (0, 1, 2, 3) and actions ('b', 'c'). The reward function is:

$$R(s,a) = \begin{cases} 1 & \text{if } s = 1 \\ 2 & \text{if } s = 3 \\ 0 & \text{otherwise} \end{cases}$$

You get the reward associated with a state on the step when you exit that state. The transition function for each action is below, where $T[i, x, j]$ is the conditional probability $P(s_{t+1} = j | a = x, s_t = i)$. The state transition diagrams are given as well, but you may find it easier to calculate the state values using the transition matrices.



$$T(s_t, \text{'b'}, s_{t+1}) = \begin{bmatrix} 0.0 & 0.9 & 0.1 & 0.0 \\ 0.9 & 0.1 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.1 & 0.9 \\ 0.9 & 0.0 & 0.0 & 0.1 \end{bmatrix}$$



$$T(s_t, \text{'c'}, s_{t+1}) = \begin{bmatrix} 0.0 & 0.1 & 0.9 & 0.0 \\ 0.9 & 0.1 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.1 & 0.9 \\ 0.9 & 0.0 & 0.0 & 0.1 \end{bmatrix}$$

Note that the **only** effect of the action is to change the transition probability from state 0 (the first row of the transition matrix: rows correspond to the input states, and columns correspond to the output states).

We would like to find the value function associated with the policy that **always** chooses action 'c'.

2A) What are the horizon 0 undiscounted values of the states under this policy?

Enter a Python list of four numbers: [0, 0, 0, 0]

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

2B) For horizon 1?

Enter a Python list of four numbers: [0.0, 1.0, 0.0, 2.0]

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

2C) For horizon 2?

Enter a Python list of four numbers: [0.1, 1.1, 1.8, 2.2]

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

For these exercises, you should read the notes on Reinforcement Learning.

1) Q-Learning

Let's simulate the Q-learning algorithm! Assume there are states 0, 1, 2, 3 and actions ('b', 'c), and discount factor $\gamma = 0.9$. Furthermore, assume that all the Q values are initialized to 0 and that the learning rate $\alpha = 0.5$.

Each row, t, in the table represents a record of experience at time t: (s_t, a_t, r_t).

In each row t, indicate what update $Q(s_t, a_t) \leftarrow q$ will be made by the Q learning algorithm based on (s_t, a_t, r_t, s_{t+1}). Note that s_{t+1} is on the next row (you might need to look ahead to the next part of the problem to see that next state value.) You will want to keep track of the overall table $Q(s_t, a_t)$ as these updates take place, spanning the multiple parts of this question.

As a reminder, the Q-learning update formula is the following:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s', a'))$$

You are welcome to do this problem by hand, though writing a small program to solve may be a good idea. To help with that, here is a variable with the history of experience:

```
experience = [(0, 'b', 0), #t = 0
              (2, 'b', 0),
              (3, 'b', 2),
              (0, 'b', 0), #t = 3
              (2, 'b', 0),
              (3, 'c', 2),
              (0, 'c', 0), #t = 6
              (1, 'b', 1),
              (0, 'b', 0),
              (2, 'c', 0), #t = 9
              (3, 'c', 2),
              (0, 'c', 0),
              (1, 'c', 1), #t = 12
              (0, 'c', 0),
              (2, 'b', 0),
              (3, 'b', 2), #t = 15
              (0, 'b', 0),
              (2, 'c', 0),
              (3, '', 0), #t = 18
              ]
```

1A)

```
 t: S  A  R
-----------
 0: 0 'b' 0
 1: 2 'b' 0
 2: 3 'b' 2
```

Enter a list of 3 numbers giving the updated Q values just after each of these times:

[0,0,1]

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

1B)

```
3: 0 'b' 0
4: 2 'b' 0
5: 3 'c' 2
```

Enter a list of 3 numbers: [0, 0.45, 1]

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

1C)

```
6: 0 'c' 0
7: 1 'b' 1
8: 0 'b' 0
```

Enter a list of 3 numbers: [0, 0.5, 0.2025]

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

1D)

```
 9: 2 'c' 0
10: 3 'c' 2
11: 0 'c' 0
```

Enter a list of 3 numbers: [0.45, 1.591125, .225]

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

1E)

```
12: 1 'c' 1
13: 0 'c' 0
14: 2 'b' 0
```

Enter a list of 3 numbers: [0.60125, 0.315, 0.94100]

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

1F)

```
15: 3 'b' 2
16: 0 'b' 0
17: 2 'c' 0
18: 3 - -
```

Enter a list of 3 numbers: [1.64175, 0.5247028125, 0.9637875]

Submit View Answer **100.00%**

You have infinitely many submissions remaining.