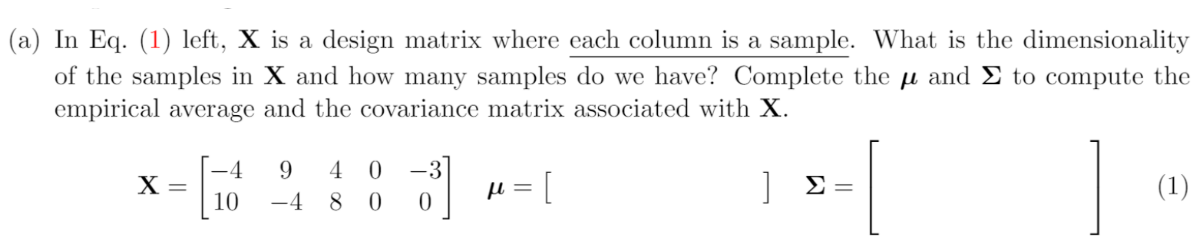
Machine Learning Exam - DONE

Exercise 1



(a) In Eq. (1) left, $\mathbf{X}$ is a design matrix where each column is a sample. What is the dimensionality of the samples in $\mathbf{X}$ and how many samples do we have? Complete the $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ to compute the empirical average and the covariance matrix associated with $\mathbf{X}$.

$$\mathbf{X} = \begin{bmatrix} -4 & 9 & 4 & 0 & -3 \\ 10 & -4 & 8 & 0 & 0 \end{bmatrix} \quad \boldsymbol{\mu} = [\qquad\qquad] \quad \boldsymbol{\Sigma} = \begin{bmatrix} \qquad\qquad \end{bmatrix} \tag{1}$$

What is the dimensionality of the samples in the design matrix?

Well the exercise tells us that the columns are the samples, so there are 2 dimensions. There are 5 samples.

Complete μ (empirical average) and Σ (covariance matrix)

The μ matrix is composed of the averages for each dimension.

$$\mu = \begin{bmatrix} \frac{-4+9+4+0-3}{5} \\ \frac{10-4+8+0+0}{5} \end{bmatrix} = \begin{bmatrix} 1.2 \\ 2.8 \end{bmatrix}$$

We compute the covariance matrix in the following way:

$$X_{\text{new}} = X - \mu = \begin{bmatrix} -5.2 & 7.8 & 2.8 & -1.2 & -4.2 \\ 7.2 & -6.8 & 5.2 & -2.8 & -2.8 \end{bmatrix}$$

$$\Sigma = \frac{1}{n} X X^T = \frac{1}{5} \begin{bmatrix} -5.2 & 7.8 & 2.8 & -1.2 & -4.2 \\ 7.2 & -6.8 & 5.2 & -2.8 & -2.8 \end{bmatrix} \begin{bmatrix} -5.2 & 7.2 \\ 7.8 & -6.8 \\ 2.8 & 5.2 \\ -1.2 & -2.8 \\ -4.2 & -2.8 \end{bmatrix}$$

$$= \frac{1}{5} \begin{bmatrix} 114.8 & -60.8 \\ -60.8 & 140.8 \end{bmatrix} = \begin{bmatrix} 22.96 & -12.16 \\ -12.16 & 28.16 \end{bmatrix}$$

Hint

There are a few methods for matrix multiplication:

Basis vectors method

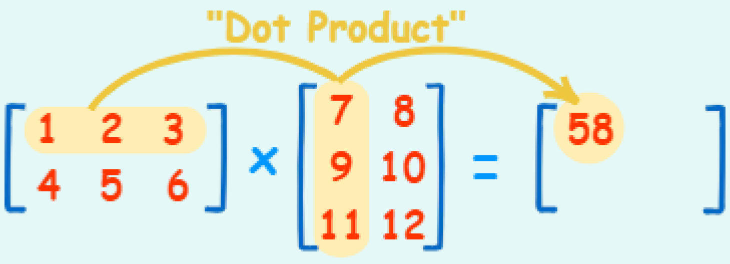
You remember that the columns of the matrix on the left are actually basis vectors. So you need to substitute them to the basis vectors of the two vectors in the matrix on the right.

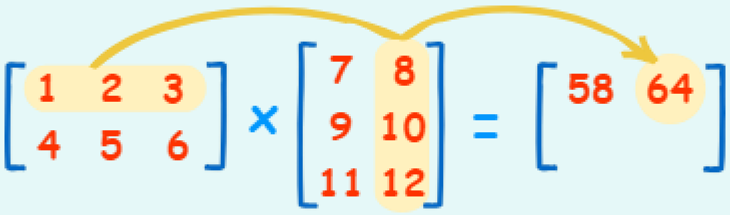
$$v_1 = -4\begin{bmatrix}-4\\10\end{bmatrix} + 9\begin{bmatrix}9\\-4\end{bmatrix} + 4\begin{bmatrix}4\\8\end{bmatrix} + 0\begin{bmatrix}0\\0\end{bmatrix} - 3\begin{bmatrix}-3\\0\end{bmatrix}$$

$$v_2 = 10\begin{bmatrix}-4\\10\end{bmatrix} + -4\begin{bmatrix}9\\-4\end{bmatrix} + 8\begin{bmatrix}4\\8\end{bmatrix} + 0\begin{bmatrix}0\\0\end{bmatrix} + 0\begin{bmatrix}-3\\0\end{bmatrix}$$

Dot product method

You compute the dot product between the rows of the first matrix and the columns of the second matrix.





(b) A point cloud with $N = 500$ poins $\mathbf{X} \doteq \{\mathbf{x}_i\}_{i=1}^{N}$ is sampled from a multi-variate Gaussian distribution in 2D and shown in Fig. 1 (a). The black diamond indicate the empirical mean of the distribution, while the black cross indicates the origin of the space. Fig. 1 (b) shows the same point cloud after a series of transformation: the process transforms the point cloud so to swap the principal components (PC).

- Describe precisely the mathematical process using linear algebra that applies these transformations to change the point cloud from Fig. 1(a) to Fig. 1(b). *Solution with explicit cosine and sine rotation matrices will NOT be considered.*
- The covariance of $\mathbf{X}$ is of type $\begin{pmatrix} \sigma^2_{x_1x_1} & -\sigma^2_{x_1x_2} \\ -\sigma^2_{x_2x_1} & \sigma^2_{x_2x_2} \end{pmatrix}$, can you rewrite the covariance of the transformed point cloud using the previous covariance values?
- What can you say about the absolute value of the determinant of the transformations that you described above?
- Now edit the transformation so that it also scales each PC to increase it by 5%.

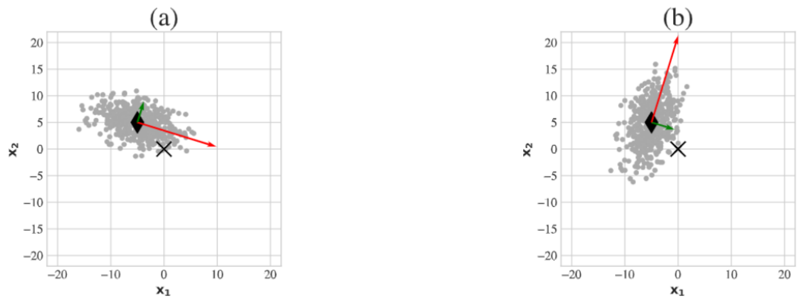
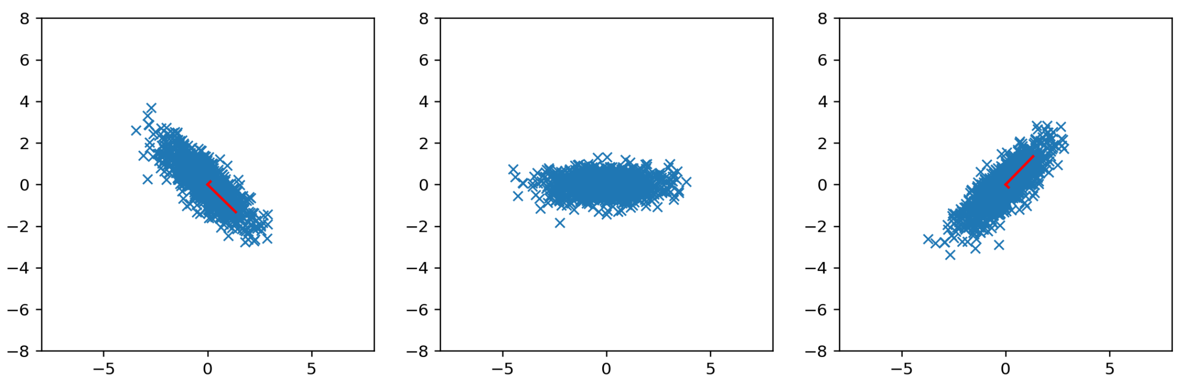


Figure 1: Point cloud transformation

How do we compute a determinant?
It might be in the exam.

Describe the process that applies those transformations

1. **We center the point cloud by subtracting the mean from all the points.**
2. We get the covariance matrix of the point cloud, which can be computed with $\frac{1}{n-1}XX^T$
3. Using PCA/Spectral decomposition, we get the eigenvectors of the covariance matrix.
4. We apply the transposed eigenvectors matrix so to bring the eigenvectors to our basis vectors.
5. Now we swap the two eigenvectors/principal components.
6. We apply the new eigenvector matrix and so that the basis vectors end up on the new eigenvectors.
7. We add the mean back to all the points.



What happens to the covariance matrix

The variance stays the same but the covariance changes sign:

```
[[1 -0.8]
 [-0.8 1]]

[[1 0.8]
 [0.8 1]]
```

What happens to the determinant

The absolute value of the determinant doesn't change at all because the space doesn't get stretched or shrinked, just rotated.

Increase the PCs by 5%

Whe need to apply a stretching transformation after step 4.
In order to do that, we need to change the basis vectors with something 5% bigger.

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1.05 & 0 \\ 0 & 1.05 \end{bmatrix}$$

(c) You have to help a colleague of yours that implemented a pipeline for supervised classification. The pipeline classifies the data linearly and it does not work well. When you debug the pipeline you realize that as a first step, the approach compressed the data to remove some attributes using Principal Component Analysis (PCA). In particular you find out that the two classes distribute in 2D as Fig. 2, where each color indicates a different class and PCA projects the data on the dimension with higher variance.

- Explain why the supervised classification with linear boundaries may not work when the data distributes as Fig. 2 using the pipeline described above.
- What can you change in the pipeline to let the linear classifier separate the data?

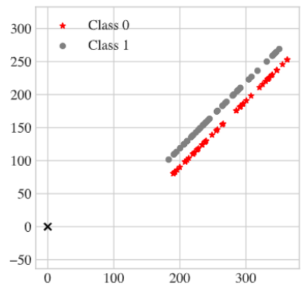
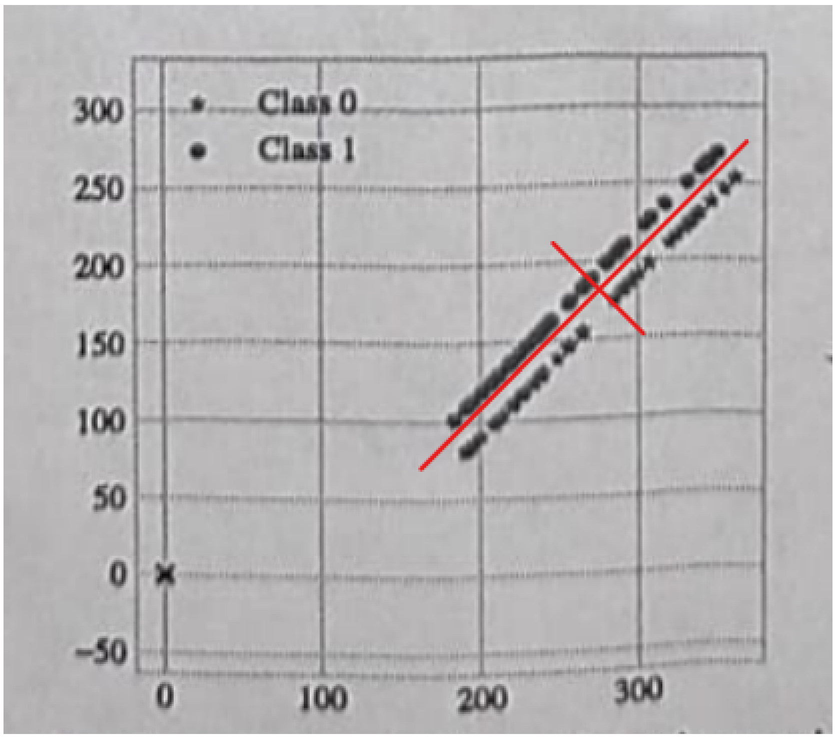


Figure 2: Supervised Classification with PCA

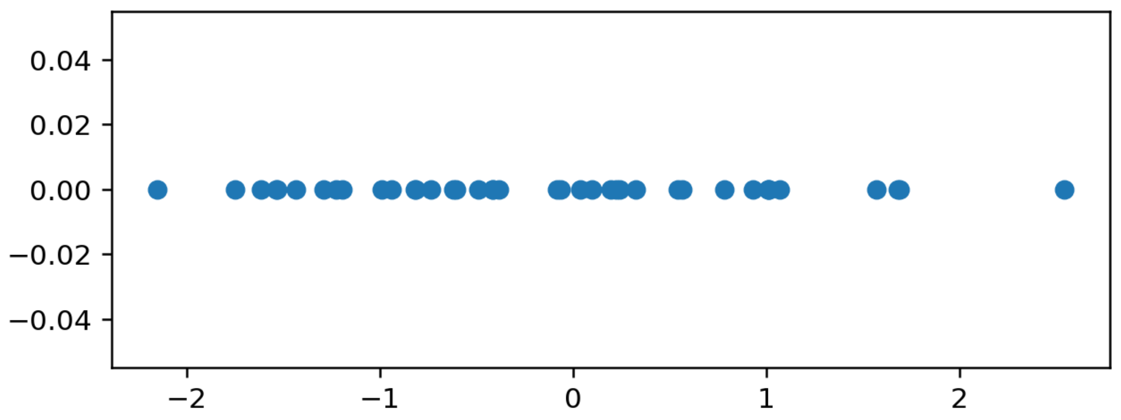
Explain why it's bad to use PCA in this case

The classifier won't work because we are projecting the data in a dimension where the classifier can't tell the difference between the two classes.

These will be the 2 principal components.



If we project all data on the first PC, we get something like this:

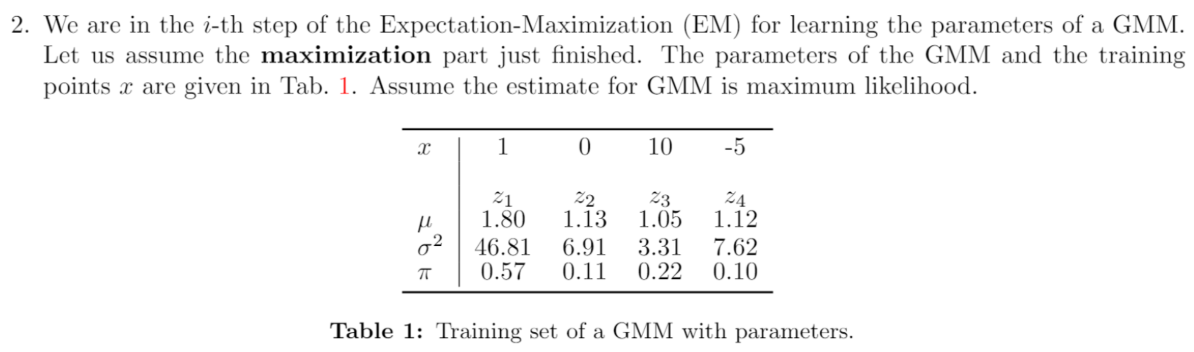


You can't distinguish classes in this scatterplot.

What would make the classifier work?

Killing the first principal component and projecting everything on the second.

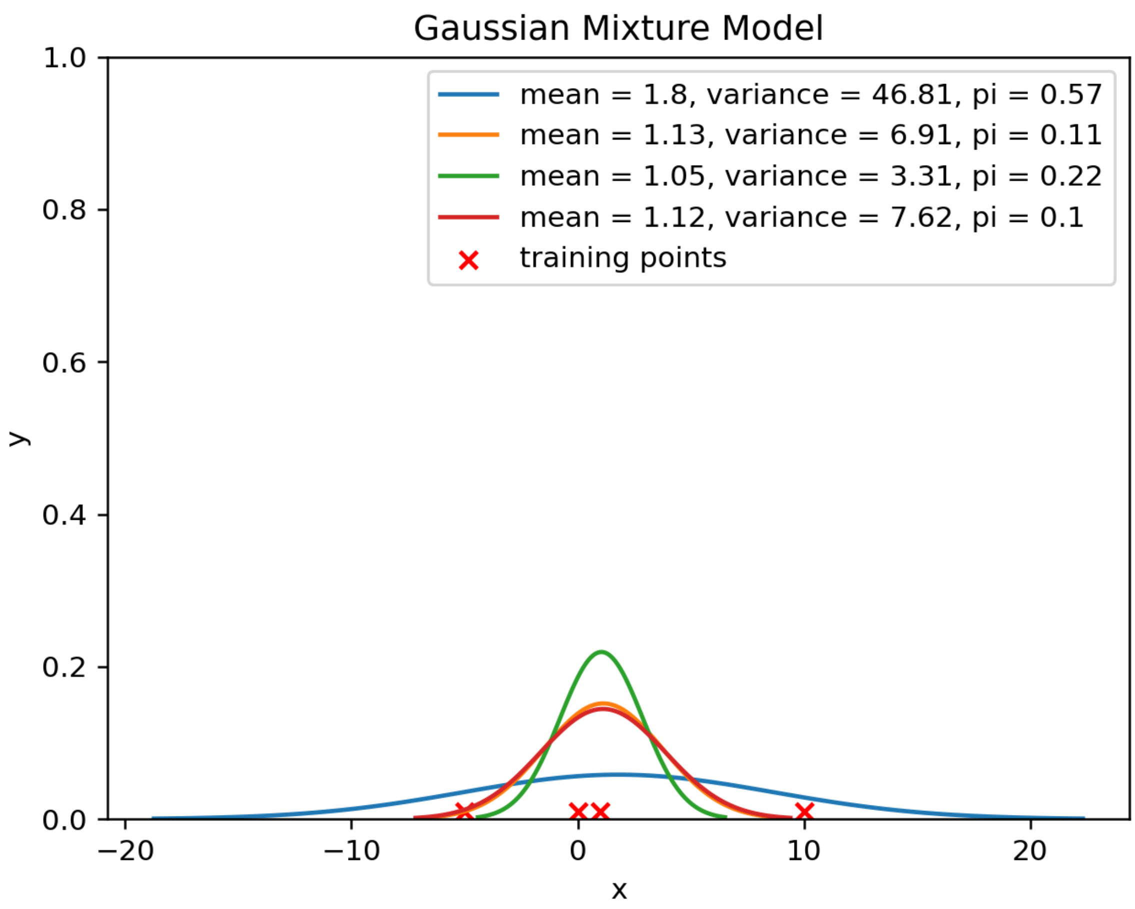
Exercise 2



2. We are in the i-th step of the Expectation-Maximization (EM) for learning the parameters of a GMM. Let us assume the **maximization** part just finished. The parameters of the GMM and the training points x are given in Tab. 1. Assume the estimate for GMM is maximum likelihood.

x	1	0	10	-5
	z_1	z_2	z_3	z_4
μ	1.80	1.13	1.05	1.12
σ^2	46.81	6.91	3.31	7.62
π	0.57	0.11	0.22	0.10

Table 1: Training set of a GMM with parameters.



Danger
It would be best to go back and study the specifics of the Expectation Maximization algorithm.

(a) Write the density of the GMM with equations using the parameters in Tab. 1, for a point x.

Write the density in equation form

Since he doesn't actually want us to compute the results(I THINK), we are gonna write the equations(not the gaussian one).

The following formula gives the density of the GMM at a point x, which basically means the total height of all the gaussians summed at that point.

$$p(x) = \sum_i \pi_i \cdot \mathcal{N}(x; \mu_i; \sigma_i^2)$$

$$p(x) = 0.57 \cdot \mathcal{N}(x; 1.80; 46.81) + 0.11 \cdot \mathcal{N}(x; 1.13; 6.91) + 0.22 \cdot \mathcal{N}(x; 1.05; 3.31) +$$

(b) In the definition of a single multi-variate Gaussian pdf, give an intuition why the determinant of the covariance matrix $\boldsymbol{\Sigma}$ is at the denominator. Moreover, although $\boldsymbol{\mu}$ is a parameter, why is not in the denominator?

Why the determinant in the denominator?

Note
The determinant $|\Sigma|$ of a matrix encodes the change in volume that results from a transformation.

In the Gaussian pdf, we have:

- Σ in the exponent, which changes the shape of the curve
- $|\Sigma|$ in the denominator, which makes the volume equal to 1.

$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right).$$

The one in the denominator is there to counter-act the change in volume that the Σ in the exponent causes(because the volume of a pdf must amount to 1).

Why is μ not in the denominator?

In Masi's words:

Quote
The location parameter μ is just translating the distribution in space somewhere thus does not affect the volume.

(c) Write down the equation to compute the responsibilities γ. Compute γ for the training point $x = 0$, assuming $p(x == 0|z_k) = 95\% \quad \forall k \in 1, \ldots, 4$. You may need only *some* parameters in Tab. 1.

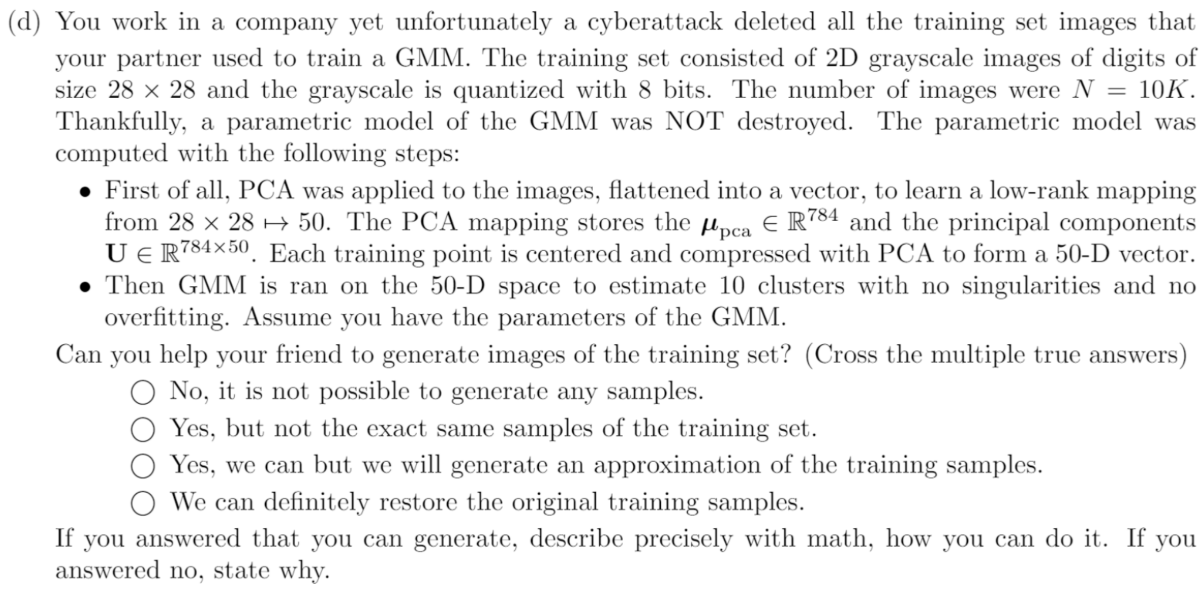
Write the equation that computes the responsibilities

$$\gamma_k = \frac{\mathcal{N}(x_i;\mu_k,\Sigma_k)\cdot\pi_k}{\sum_j \mathcal{N}(x_i;\mu_j,\Sigma_j)\cdot\pi_j}$$

The exercise is telling us that at $x = 0$, each gaussian has the same height, they just have different mixing coefficients.

$$\gamma_k = \frac{0.95 \cdot \pi_k}{\sum_j 0.95 \cdot \pi_j}$$

So the responsibilities are just the mixing coefficients.



(d) You work in a company yet unfortunately a cyberattack deleted all the training set images that your partner used to train a GMM. The training set consisted of 2D grayscale images of digits of size 28×28 and the grayscale is quantized with 8 bits. The number of images were $N = 10K$. Thankfully, a parametric model of the GMM was NOT destroyed. The parametric model was computed with the following steps:

- First of all, PCA was applied to the images, flattened into a vector, to learn a low-rank mapping from $28 \times 28 \mapsto 50$. The PCA mapping stores the $\boldsymbol{\mu}_{\text{pca}} \in \mathbb{R}^{784}$ and the principal components $\mathbf{U} \in \mathbb{R}^{784 \times 50}$. Each training point is centered and compressed with PCA to form a 50-D vector.
- Then GMM is ran on the 50-D space to estimate 10 clusters with no singularities and no overfitting. Assume you have the parameters of the GMM.

Can you help your friend to generate images of the training set? (Cross the multiple true answers)

○ No, it is not possible to generate any samples.
○ Yes, but not the exact same samples of the training set.
○ Yes, we can but we will generate an approximation of the training samples.
○ We can definitely restore the original training samples.

If you answered that you can generate, describe precisely with math, how you can do it. If you answered no, state why.

Generate 10K samples from the GMM and reproject them

In this case, after PCA we are left with 10k images in the form of vector of 50 pixels that can have value in between [0, 255].

But Masi also gives us the mean of the original images and the U matrix used to compress them.

Hint
This is useful because we can reverse the process and get a compressed version of the original images.

If we want to generate those samples using a GMM, **repeat the following process 10K times**:

1. By using inverse transform sampling on the cumulative mass function of the mixing coefficients, we select which digit we want to generate.

Hint
This is because the gaussians don't have all the same size, so some are more probable to be picked or more present in the GMM.

To solve this, we do a weighted random pick of the gaussian.

2. We sample from the selected gaussian.
3. We reproject the sampled image from the subspace to its original dimensions by multiplying it by U^T and adding μ_{pca}.

```python
import matplotlib.pyplot as plt
np.random.seed(0)

def inverse_sampling(pmf):
    return np.argmin((np.random.rand(1)[:, None] > pmf.cumsum()), axis=1)


################### model params #######################
N_samples = 1000  # we sample this amount of points
z_to_gaussian = {0: ([0, 0], [[0.5, 0], [0, 0.15]], 'red'),
                 1: ([3, 2], [[1.25, 0], [0, 0.75]], 'green'),
                 2: ([-2, -2], [[1, -0.74], [-0.74, 1]], 'blue'),
                 }
mixing = np.array([0.2, 0.5, 0.3])  # mixing coefficients
########################################################
for _ in range(0, N_samples):  # for each sample
    z = inverse_sampling(mixing)  # sample k using mixing coefficients
    *normal, color = z_to_gaussian[z[0]]  # now sample the data from  N_k
    x, y = np.random.multivariate_normal(
        *normal, 1).T  # sample 1 point at time for clarity
    plt.plot(x, y, '.', color='black')
plt.axis('equal')
plt.title('p(x)')
plt.show()
```

Exercise 3

3. We are given an already learned decision tree for multi-class classification shown in Fig. 3 below. Each shape represents a training sample where the shape identifies the class. The feature of each point are two-dimensional $\mathbf{x} = [x_1, x_2]$.

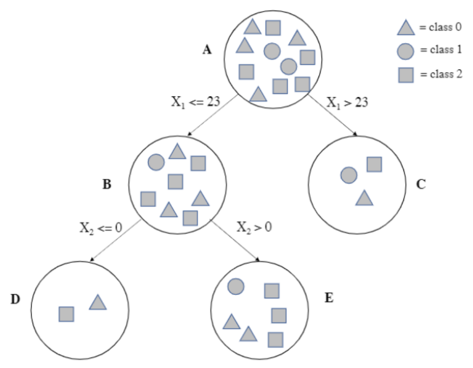


Figure 3: Decision Tree for multi-class classification.

⚡ **Danger**
Go back and study Gini impurity and Missclassification!

(a) Compute the Impurity using the entropy **for the entire TREE.**

Entropy of tree

The entropy of the tree is given by the entropy of the last leaves weighed by the number of elements contained in them:

$$\frac{2}{11}H(D) + \frac{6}{11}H(E) + \frac{3}{11}H(C)$$

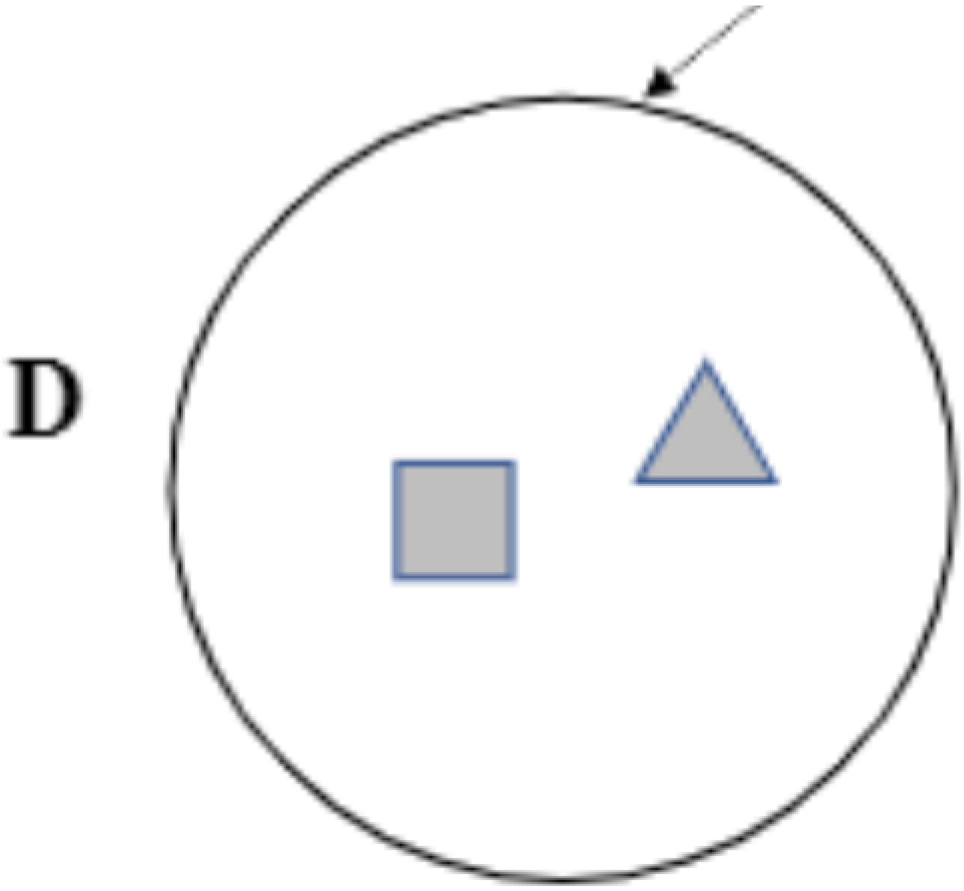
(b) You have a test sample $\mathbf{x} = [x_1, x_2] = [-1, -1000]$, which class you assign to $\mathbf{x}$—triangle, square or circle or maybe it is better to return that the algorithm is unsure?

Inference on a sample

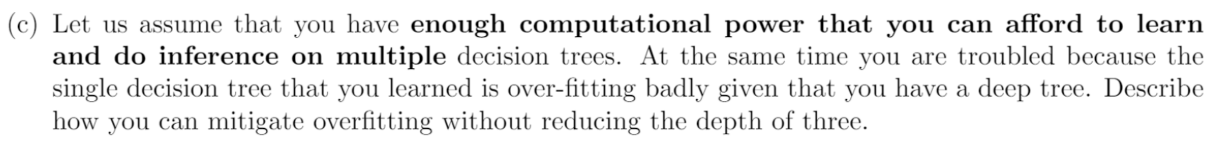
Masi gives us the sample [-1, -1000].

We know that:

1. If $x_1 <= 23$ then we go to leaf B.
2. Then if $x_2 <= 0$ then we go to leaf D.



Since there is equal probability that the class is square and triangle, it's best to say that the algorithm is unsure here.



(c) Let us assume that you have **enough computational power that you can afford to learn and do inference on multiple** decision trees. At the same time you are troubled because the single decision tree that you learned is over-fitting badly given that you have a deep tree. Describe how you can mitigate overfitting without reducing the depth of three.

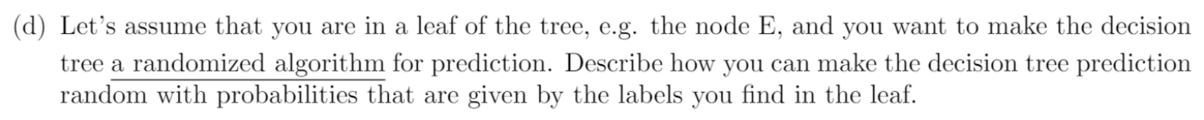
How to reduce overfitting in a decision tree

We cannot use the following strategies because:

- Minimizing the leaf size: Would change the depth of the tree
- Capping the maximum depth: Would change the depth of the tree
- Conditional leaf generation(A node will be split if this split induces a decrease of the impurity greater than or equal to a value): Would change the depth of the tree

So we use Bagging and Ensamble, which is a technique that consists in splitting the dataset and training a decision tree on each one of the splits.

Then to make inference we average the outputs of the random forest and we have got our model with reduced variance.

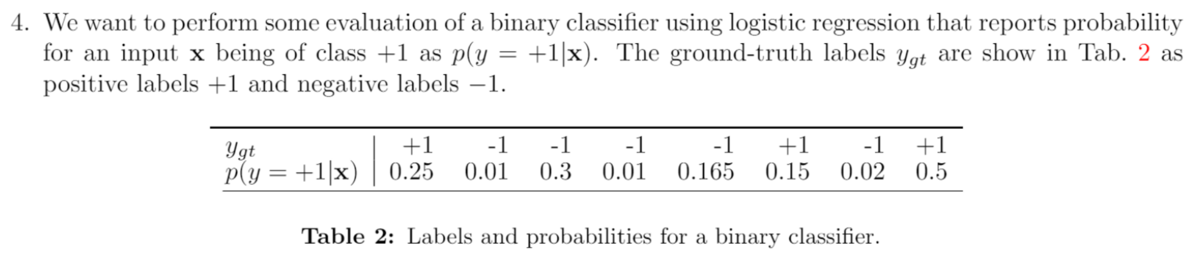


(d) Let's assume that you are in a leaf of the tree, e.g. the node E, and you want to make the decision tree a randomized algorithm for prediction. Describe how you can make the decision tree prediction random with probabilities that are given by the labels you find in the leaf.

How to make random weighed prediction based on the elements in the leaves

We apply inverse transform sampling on the probabilities of the elements in the leaves???????????????????

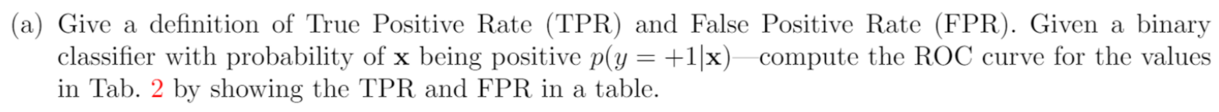
Exercise 4



4. We want to perform some evaluation of a binary classifier using logistic regression that reports probability for an input $\mathbf{x}$ being of class $+1$ as $p(y=+1|\mathbf{x})$. The ground-truth labels y_{gt} are show in Tab. 2 as positive labels $+1$ and negative labels -1.

y_{gt}	+1	-1	-1	-1	-1	+1	-1	+1
$p(y=+1\|\mathbf{x})$	0.25	0.01	0.3	0.01	0.165	0.15	0.02	0.5

Table 2: Labels and probabilities for a binary classifier.



(a) Give a definition of True Positive Rate (TPR) and False Positive Rate (FPR). Given a binary classifier with probability of $\mathbf{x}$ being positive $p(y=+1|\mathbf{x})$—compute the ROC curve for the values in Tab. 2 by showing the TPR and FPR in a table.

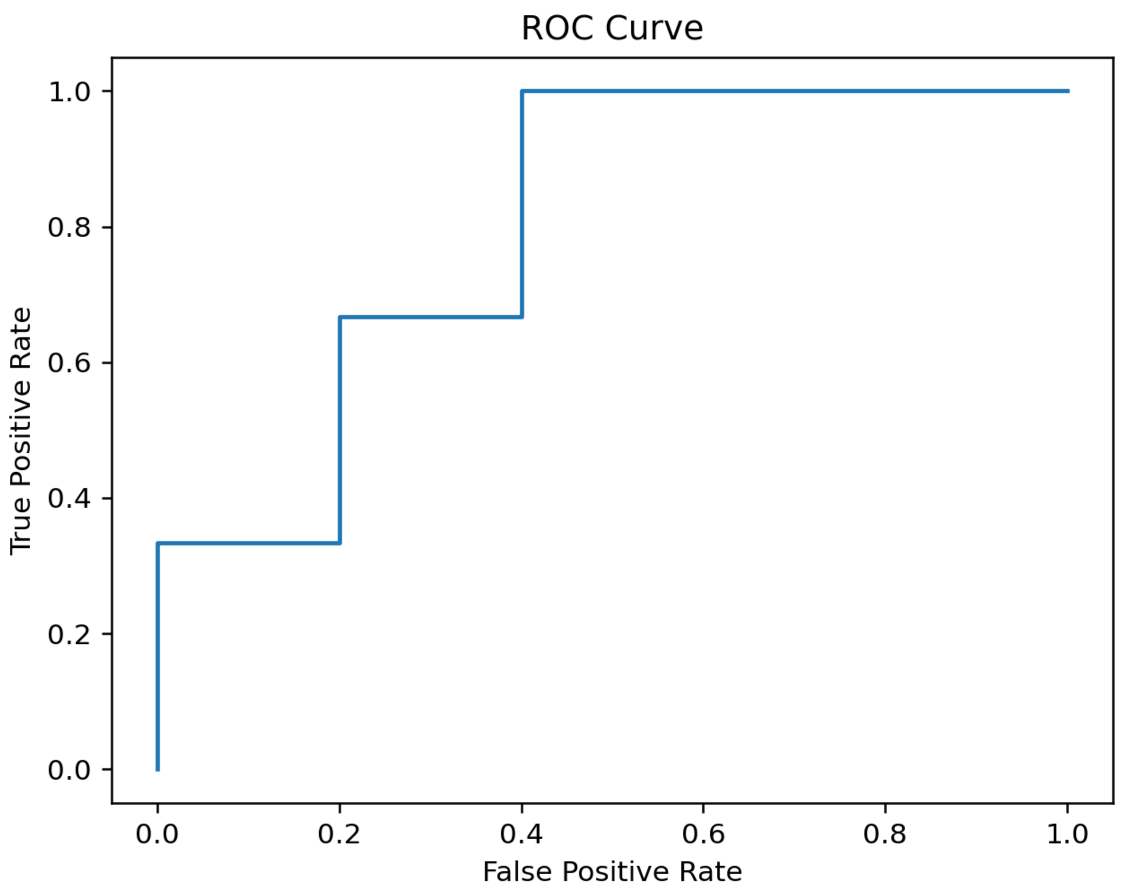
What are TPR and FPR?

The true positive rate, also referred to sensitivity or recall, is **used to measure the percentage of actual positives which are correctly identified**.

FPR is the opposite.

What is the ROC curve and how do we compute it?

ROC curve is a performance measurement for the classification problems at various threshold settings.
It plots TPR and FPR at various threshold values.



Exercise 5

5. We have to analyze a neural network in the form of a multi-layer perceptron (MLP). The neural network details follow in the sub questions below.

(a) Deduce and write how many trainable parameters you have with a MLP with input feature vectors with dimension equal to 2048, a first layer $\mathbf{W}_1$ with 1024 units/neurons. This layer $\mathbf{W}_1$ is not trained, acts as a random projection and in pytorch is set `required_grad=False`. A second layer $\mathbf{W}_2$ with 512 units/neurons, and a final multi-class classification layer $\mathbf{W}_3$ with 10 units/neurons. Assume all layers do NOT have the bias term. The network uses ReLU activation function after each layer except the classification that uses softmax. Write down the equation for the computation, not just the final value. Moreover, what is the dimensionality of the logit vector that the network produces?

How many parameters?

Layer 1) We have 2048 weights for each neuron of the first layer, so:

$$2048 \cdot 1024 = 20.971.52$$

BUT THIS LAYER IS NOT TRAINABLE, SO IT DOESNT COUNT.

Layer 2) We have 1024 weights for each neuron of the second layer, so:

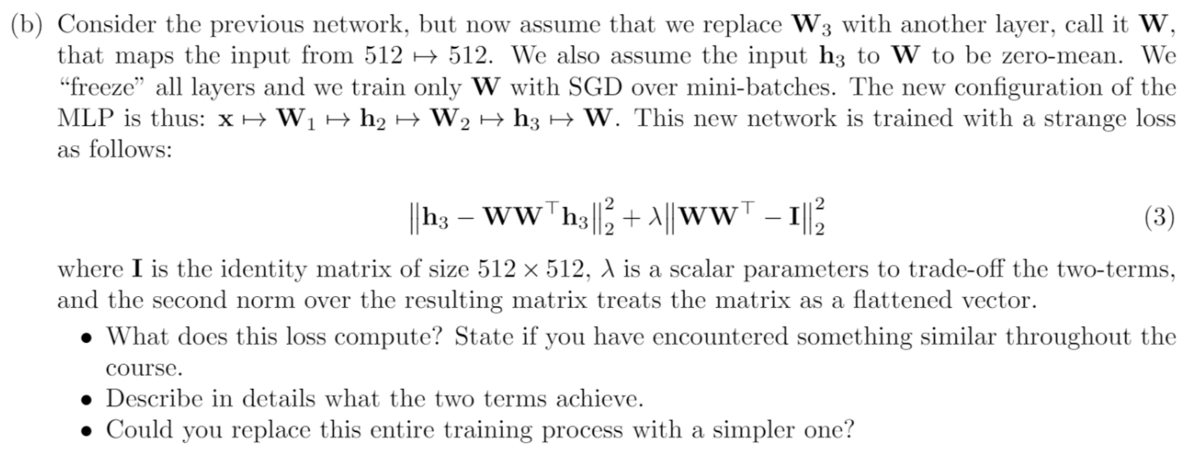
$$1024 \cdot 512 = 524.288$$

Layer 3) We have 512 weights for each neuron of the second layer, so:

$$512 \cdot 10 = 5.120$$

When we sum them all toghether, we get the total number of pararmeters:

$$524.288 + 5.120 = 529.408$$



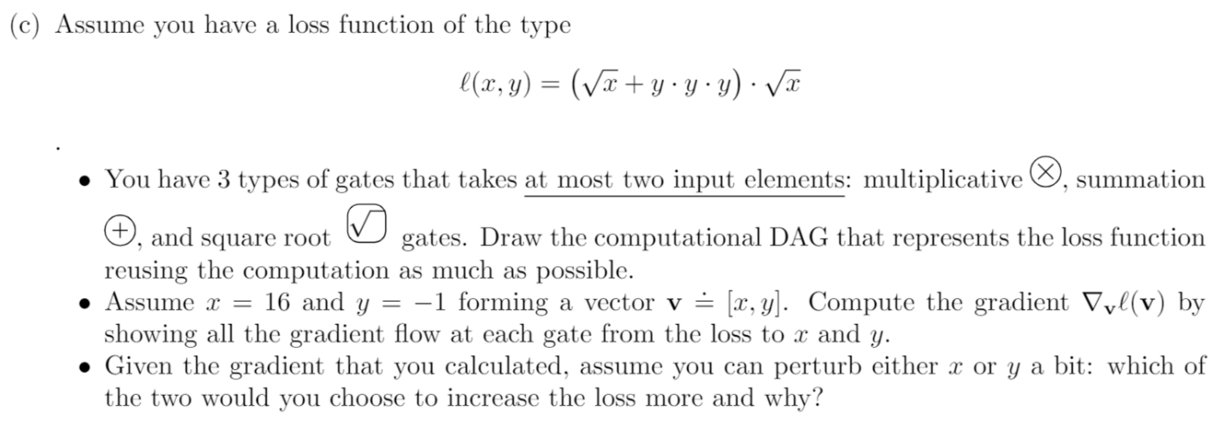
(b) Consider the previous network, but now assume that we replace $\mathbf{W}_3$ with another layer, call it $\mathbf{W}$, that maps the input from $512 \mapsto 512$. We also assume the input $\mathbf{h}_3$ to $\mathbf{W}$ to be zero-mean. We "freeze" all layers and we train only $\mathbf{W}$ with SGD over mini-batches. The new configuration of the MLP is thus: $\mathbf{x} \mapsto \mathbf{W}_1 \mapsto \mathbf{h}_2 \mapsto \mathbf{W}_2 \mapsto \mathbf{h}_3 \mapsto \mathbf{W}$. This new network is trained with a strange loss as follows:

$$\left\|\mathbf{h}_3 - \mathbf{W}\mathbf{W}^\top \mathbf{h}_3\right\|_2^2 + \lambda \left\|\mathbf{W}\mathbf{W}^\top - \mathbf{I}\right\|_2^2 \tag{3}$$

where $\mathbf{I}$ is the identity matrix of size 512×512, λ is a scalar parameters to trade-off the two-terms, and the second norm over the resulting matrix treats the matrix as a flattened vector.

- What does this loss compute? State if you have encountered something similar throughout the course.
- Describe in details what the two terms achieve.
- Could you replace this entire training process with a simpler one?

wtf?



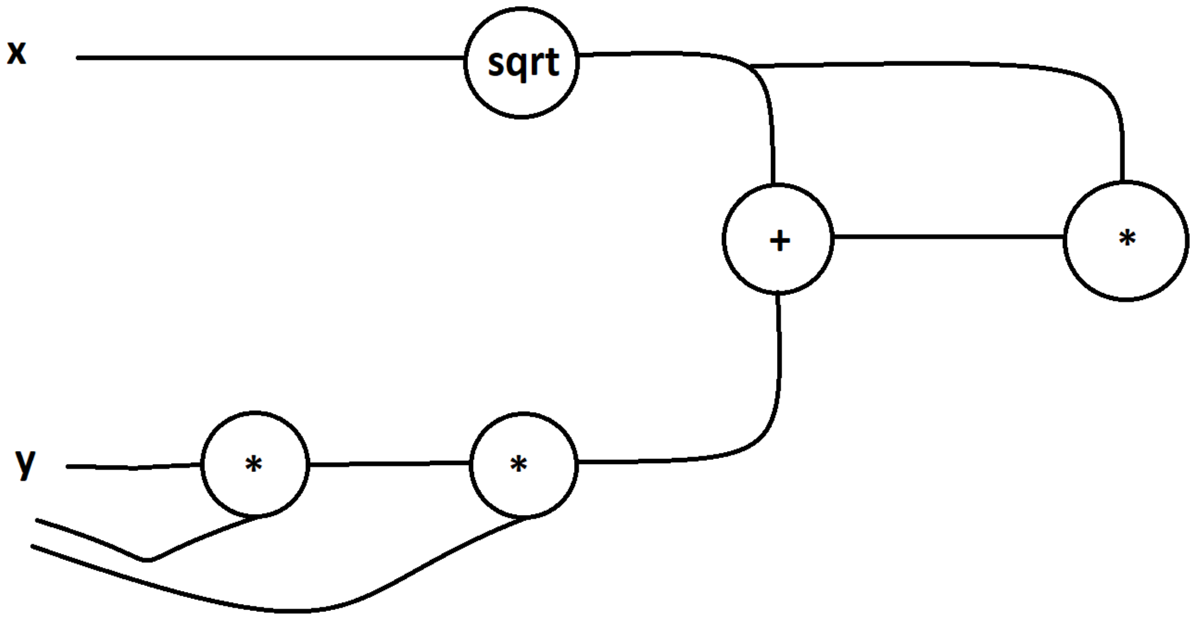
(c) Assume you have a loss function of the type

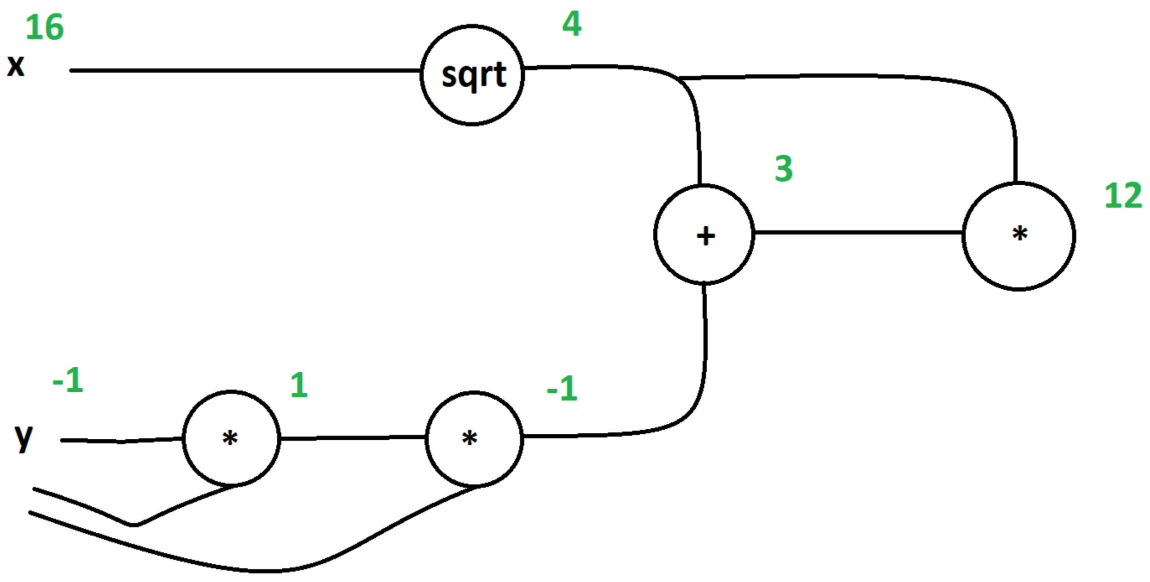
$$\ell(x, y) = \left(\sqrt{x} + y \cdot y \cdot y\right) \cdot \sqrt{x}$$

.

- You have 3 types of gates that takes at most two input elements: multiplicative $\otimes$, summation $\oplus$, and square root $\boxed{\sqrt{\ }}$ gates. Draw the computational DAG that represents the loss function reusing the computation as much as possible.
- Assume $x = 16$ and $y = -1$ forming a vector $\mathbf{v} \doteq [x, y]$. Compute the gradient $\nabla_{\mathbf{v}} \ell(\mathbf{v})$ by showing all the gradient flow at each gate from the loss to x and y.
- Given the gradient that you calculated, assume you can perturb either x or y a bit: which of the two would you choose to increase the loss more and why?

Draw the DAG





Compute the gradient using the chain rule

At the last gate, we compute the derivatives with respect to each component of $\mathcal{L}$:

$$\mathcal{L} = \mathrm{sqrt} \cdot \mathrm{plus} \quad \rightarrow \quad \frac{\partial \mathcal{L}}{\partial \mathrm{plus}} = \mathrm{sqrt}, \quad \frac{\partial \mathcal{L}}{\partial \mathrm{sqrt}} = \mathrm{plus}$$

Now, in order to apply the chain rule, we would need to know the derivatives of sqrt and plus.

$$\mathrm{sqrt} = \sqrt{x} \quad \rightarrow \quad \frac{\partial \mathrm{sqrt}}{\partial x} = \frac{1}{2} \cdot x^{\frac{1}{2}-1} = \frac{1}{2x}$$

$$\mathrm{plus} = \mathrm{cube} + \mathrm{sqrt} \quad \rightarrow \quad \frac{\partial \mathrm{plus}}{\partial \mathrm{cube}} = \mathrm{sqrt}, \quad \frac{\partial \mathrm{plus}}{\partial \mathrm{sqrt}} = \mathrm{cube}$$

The only remaining thing is cube.

$$\text{cube} = y^3 \quad \rightarrow \quad \frac{\partial \text{cube}}{\partial \text{y}} = 3y^2$$

Let's kill a fucking gate.

Useful things to remember

ROC and AUC

ROC is computed by TPR and FPR, which are both divided respectively by $Positives_{\text{ground truth}}$ and $Negatives_{\text{ground truth}}$.

The AUC is computed by computing:

$$\text{AUC} = \sum_i (FPR_i - FPR_{i-1}) \cdot TPR_i$$

Useful local gradients

Sigmoid:

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad \rightarrow \quad \sigma(x)(1-\sigma(x))$$

Softmax + Cross-entropy:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{j=1}^{q} y_j \log \frac{e^{z_j}}{\sum_{k=1}^{q} e^{z_k}}$$

$$\frac{\partial \mathcal{L}}{\partial_{z_j}}(\mathbf{y}, \hat{\mathbf{y}} \,\text{or}\, \text{z}) = \frac{e^{z_j}}{\sum_{k=1}^{q} e^{z_k}} - y_j = \text{softmax}(\mathbf{z})_j - y_j$$

Product gate:

Usually a product gate is composed of two elements, if you fix one, the other rema

Sum gate:

Usually a Sum gate is composed of two elements, if you fix one, it goes to 1 and th

+1 gate:

Since it's just a constant, the one goes to zero and the input variable goes to 1.

$*$-1 gate:

[illegible]

It should just return -1, because the variable goes to 1.