CS 446 MJT — Homework 6

your NetID here

Version 2

Instructions.

- Homework is due **Tuesday, April 30, at 11:59pm**; no late homework accepted.
- Everyone must submit individually at gradescope under `hw6` and `hw6code`.
- The "written" submission at `hw6` **must be typed**, and submitted in any format gradescope accepts (to be safe, submit a PDF). You may use LaTeX, markdown, google docs, MS word, whatever you like; but it must be typed!
- When submitting at `hw6`, gradescope will ask you to mark out boxes around each of your answers; please do this precisely!
- Please make sure your NetID is clear and large on the first page of the homework.
- Your solution **must** be written in your own words. Please see the course webpage for full academic integrity information. Briefly, you may have high-level discussions with at most 3 classmates, whose NetIDs you should place on the first page of your solutions, and you should cite any external reference you use; despite all this, your solution must be written in your own words.
- We reserve the right to reduce the auto-graded score for `hw6code` if we detect funny business (e.g., rather than implementing an algorithm, you keep re-submitting the assignment to the auto-grader, eventually completing a binary search for the answers).
- There are **no regrade requests** on `hw6code`, which is the code auto-grader; however, you can re-submit and re-grade as many times as you like before the deadline! Start early and report any issues on piazza!
- Methods and functions in the template and utility code include docstrings to describe the inputs and outputs. The autograder relies on correct implementations of these methods. Follow the docstrings to avoid failing tests.

1. **k-means.**

Recall the k-means problem with n data points $S = (\boldsymbol{x}_i)_{i=1}^n$ in $\mathbb{R}^d$, k centers $(\boldsymbol{\mu}_i)_{i=1}^k$ in $\mathbb{R}^d$, and corresponding clusters $(S_j)_{j=1}^k$. The objective of k-means is then to minimize the cost:

$$\phi_S(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k) = \sum_{i=1}^{n} \min_j \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2.$$

In this problem you will develop an alternate formulation of this cost function in terms of pairwise distances.

(a) Let $S_{\boldsymbol{z}} \subset S$ be the cluster induced by using a particular point $\boldsymbol{z} \in \mathbb{R}^d$ as a center. Then the cost of using any point $\boldsymbol{z}$ as a center is then given by

$$\phi_{S_{\boldsymbol{z}}}(\boldsymbol{z}) = \sum_{\boldsymbol{x} \in S_z} \|\boldsymbol{x} - \boldsymbol{z}\|^2.$$

Let $\boldsymbol{\mu}(S_{\boldsymbol{z}})$ be the sample mean of the cluster $S_{\boldsymbol{z}}$. Prove that the cost $\phi_{S_{\boldsymbol{z}}}(\boldsymbol{z})$ is equivalent to

$$\phi_{S_{\boldsymbol{z}}}(\boldsymbol{z}) = \phi_{S_{\boldsymbol{z}}}(\boldsymbol{\mu}(S_{\boldsymbol{z}})) + |S_{\boldsymbol{z}}| \|\boldsymbol{\mu}(S_{\boldsymbol{z}}) - \boldsymbol{z}\|^2.$$

(b) Show that

$$\phi_{S_j}(\boldsymbol{\mu}_j) = \frac{1}{2|S_j|} \sum_{\boldsymbol{a}, \boldsymbol{b} \in S_j} \|\boldsymbol{a} - \boldsymbol{b}\|^2.$$

Conclude that solving the k-means problem is equivalent to solving

$$\min_{S_1, \ldots, S_k} \sum_{j=1}^{k} \frac{1}{2|S_j|} \sum_{\boldsymbol{a}, \boldsymbol{b} \in S_j} \|\boldsymbol{a} - \boldsymbol{b}\|^2.$$

Solution.

(Your solution here.)

(a)

$$\begin{aligned}
\phi_{S_{\boldsymbol{z}}}(\boldsymbol{z}) &= \sum_{\boldsymbol{x} \in S_z} \|\boldsymbol{x} - \boldsymbol{z}\|^2 \\
&= \sum_{\boldsymbol{x} \in S_z} \|\boldsymbol{x} - \boldsymbol{\mu}(\boldsymbol{S}_z) + \boldsymbol{\mu}(\boldsymbol{S}_z) - \boldsymbol{z}\|^2 \\
&= \sum_{\boldsymbol{x} \in S_z} (\|\boldsymbol{x} - \boldsymbol{\mu}(\boldsymbol{S}_z)\|^2 + \|\boldsymbol{\mu}(\boldsymbol{S}_z) - \boldsymbol{z}\|^2) + 2 \sum_j (\boldsymbol{\mu}(\boldsymbol{S}_z)_j - \boldsymbol{z}_j) \sum_{\boldsymbol{x} \in S_z} (\boldsymbol{x}_j - \frac{1}{|\boldsymbol{S}_z|} \sum_{\boldsymbol{x}' \in S_z} \boldsymbol{x}'_j) \\
&= \phi_{S_{\boldsymbol{z}}}(\boldsymbol{\mu}(S_{\boldsymbol{z}})) + |S_{\boldsymbol{z}}| \|\boldsymbol{\mu}(S_{\boldsymbol{z}}) - \boldsymbol{z}\|^2.
\end{aligned}$$

(b)

$$\begin{aligned}
\frac{1}{2|S_j|}\sum_{\boldsymbol{a},\boldsymbol{b}\in S_j}\|\boldsymbol{a}-\boldsymbol{b}\|^2 &= \frac{1}{2|S_j|}\sum_k\sum_{\boldsymbol{a},\boldsymbol{b}\in S_j}(\boldsymbol{a}_k-\boldsymbol{\mu}_{jk}+\boldsymbol{\mu}_{jk}-\boldsymbol{b}_k)^2 \\
&= \frac{1}{2|S_j|}\sum_k\sum_{\boldsymbol{a},\boldsymbol{b}\in S_j}[(\boldsymbol{a}_k-\boldsymbol{\mu}_{jk})^2+2(\boldsymbol{a}_k-\boldsymbol{\mu}_{jk})(\boldsymbol{\mu}_{jk}-\boldsymbol{b}_k)+(\boldsymbol{\mu}_{jk}-\boldsymbol{b}_k)^2] \\
&= \sum_k[\frac{1}{2|\boldsymbol{S}_j|}(|\boldsymbol{S}_j|\sum_{\boldsymbol{a}\in S_j}(\boldsymbol{a}_k-\boldsymbol{\mu}_{jk})^2+|\boldsymbol{S}_j|\sum_{\boldsymbol{a}\in S_j}(\boldsymbol{\mu}_{jk}-\boldsymbol{b}_k)^2+2\sum_{\boldsymbol{a},\boldsymbol{b}\in S_j}(\boldsymbol{a}_k-\boldsymbol{\mu}_{jk})(\boldsymbol{\mu}_{jk}-\boldsymbol{b}_k)] \\
&= \sum_k[\sum_{\boldsymbol{a}\in S_j}(\boldsymbol{a}_k-\boldsymbol{\mu}_{jk})^2+\frac{1}{|\boldsymbol{S}_j|}\sum_{\boldsymbol{a}_k\in S_j}(\boldsymbol{a}_k-\boldsymbol{\mu}_{jk})(\boldsymbol{\mu}_{jk}-\boldsymbol{b}_k)] \\
&= \sum_{\boldsymbol{a}\in S_j}(\boldsymbol{a}-\boldsymbol{\mu}_{jk})^2 \\
&= \phi_{S_j}(\boldsymbol{\mu}_j)
\end{aligned}$$

The objective of k-means is to minimize

$$\begin{aligned}
\min_{\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_k}\phi_S(\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_k) &= \min_{S_1,\ldots,S_k}\sum_{j=1}^k\phi_{S_j}(\boldsymbol{\mu}_j) \\
&= \min_{S_1,\ldots,S_k}\sum_{j=1}^k\frac{1}{2|S_j|}\sum_{\boldsymbol{a},\boldsymbol{b}\in S_j}\|\boldsymbol{a}-\boldsymbol{b}\|^2
\end{aligned}$$

2. **Wasserstein Distance.**

Consider two discrete distributions with weights $(\alpha_i)_{i=1}^n$ and $(\beta_j)_{j=1}^m$ on points $(\boldsymbol{x}_i)_{i=1}^n$ and $(\boldsymbol{z}_j)_{j=1}^m$. The Wasserstein distance between these two distributions (let's call them μ and ν) is

$$W(\mu,\nu) = \max_{\|f\|_{\text{Lip}}\leq 1} \sum_{i=1}^n \alpha_i f(\boldsymbol{x}_i) - \sum_{j=1}^m \beta_j f(\boldsymbol{z}_j).$$

(a) Suppose $n = m$ and $\alpha_i = \beta_i = 1/n$, meaning both distributions are uniform. Show that for any permutation π of $(1,\ldots,n)$.

$$W(\mu,\nu) \leq \max_i \|\boldsymbol{x}_i - \boldsymbol{z}_{\pi(i)}\|.$$

Note that this implies $W(\mu,\nu) \leq \min_\pi \max_i \|\boldsymbol{x}_i - \boldsymbol{z}_{\pi(i)}\|$.

(b) Choose $((\alpha_i,\boldsymbol{x}_i))_{i=1}^n$ and $((\beta_j,\boldsymbol{z}_j))_{j=1}^m$ with $m = n$ so that

$$0 < W(\mu,\nu) = \min_\pi \max_i \|\boldsymbol{x}_i - \boldsymbol{z}_{\pi(i)}\|.$$

(c) Choose $((\alpha_i,\boldsymbol{x}_i))_{i=1}^n$ and $((\beta_j,\boldsymbol{z}_j))_{j=1}^m$ with $m = n$ so that

$$0 < W(\mu,\nu) \leq \frac{1}{100} \min_\pi \max_i \|\boldsymbol{x}_i - \boldsymbol{z}_{\pi(i)}\|.$$

Solution. *(Your solution here.)*

(a) Because $\|f\|_{\text{Lip}} \leq 1$, $\|f(x) - f(y)\| \leq \|x - y\|$

$$\begin{aligned}
W(\mu,\nu) &= \max_{\|f\|_{\text{Lip}}\leq 1} \sum_{i=1}^n \alpha_i f(\boldsymbol{x}_i) - \sum_{j=1}^m \beta_j f(\boldsymbol{z}_j) \\
&= \max_{\|f\|_{\text{Lip}}\leq 1} (\sum_{i=1}^n f(\boldsymbol{x}_i) - \sum_{i=1}^n f(\boldsymbol{z}_j)) \\
&\leq \max_{\|f\|_{\text{Lip}}\leq 1} \max_i [f(\boldsymbol{x}_i) = f(\boldsymbol{z}_{\pi(i)}] \\
&\leq \max_i \|\boldsymbol{x}_i - \boldsymbol{z}_{\pi(i)}\|
\end{aligned}$$

(b) Make $\alpha_i = \beta_i$, $\|\boldsymbol{x}_i - \boldsymbol{z}_{\pi(i)}\| = \textit{constant}$ for any permutation π and for any i

(c) Assume $\|\boldsymbol{x}_1 - \boldsymbol{z}_1\| = \min_\pi \max_i \|\boldsymbol{x}_i - \boldsymbol{z}_{\pi(i)}\|$. Also assume $\epsilon\|\boldsymbol{x}_i - \boldsymbol{z}_{\pi(i)}\| \geq \|\boldsymbol{x}_i - \boldsymbol{z}_{\pi(i)}\|$ for any i and π. Choose $\alpha_i = \beta_i$ for any i. Then we have

$$\begin{aligned}
W(\mu,\nu) &= \max_{\|f\|_{\text{Lip}}\leq 1} \sum_{i=1}^n \alpha_i f(\boldsymbol{x}_i) - \sum_{j=1}^m \beta_j f(\boldsymbol{z}_j) \\
&= \sum_{i=1}^n \alpha_i \|\boldsymbol{x}_i - \boldsymbol{z}_i\| \\
&\leq \sum_{i=2}^n \alpha_i \epsilon \|\boldsymbol{x}_1 - \boldsymbol{z}_1\| + \alpha_1 \|\boldsymbol{x}_1 - \boldsymbol{z}_1\| \\
&= (1-\alpha_1)\epsilon \|\boldsymbol{x}_1 - \boldsymbol{z}_1\| + \alpha_1 \|\boldsymbol{x}_1 - \boldsymbol{z}_1\| \\
&\leq \frac{1}{100} \|\boldsymbol{x}_1 - \boldsymbol{z}_1\|
\end{aligned}$$

So we have

$$(1-\alpha_1)\epsilon + \alpha_1 \leq \frac{1}{100}$$

Then choose any α_1 and ϵ satisfy the inequality above.

3. **Boosting.**

In this problem we will consider boosting applied to interval classifiers on the real line. An interval classifier has the form $h(x) := \mathbb{1}[a \le x \le b]$; let $\mathcal{H}$ denote all such classifiers (meaning for all $a \le b$). Boosting therefore outputs a function of the form

$$g(x) = \sum_{j=1}^{m} \alpha_j h_j(x) = \sum_{j=1}^{m} \alpha_j \cdot \mathbb{1}[a_j \le x \le b_j].$$

For all parts of this problem let $(x_i, y_i)_{i=1}^n$ be a data set of n points $x_i \in \mathbb{R}$ along with associated labels $y_i \in \{-1, 1\}$. Assume that the x_i are in sorted order and distinct, meaning $x_i < x_{i+1}$.

(a) Let $(q_1, \ldots, q_n)$ be any weights on the training set, meaning $q_i \ge 0$ and $\sum_i q_i = 1$. Show that

$$\min_{h \in \mathcal{H}} \sum_{i=1}^{n} q_i \mathbb{1}[2h(x_i) - 1 \ne y_i] \;\le\; \min \left\{ \sum_{\substack{i \in \{1,\ldots,n\} \\ y_i > 0}} q_i, \sum_{\substack{i \in \{1,\ldots,n\} \\ y_i < 0}} q_i \right\}.$$

Remark. This calculation is related to the "weak learning assumption" discussed in lecture. The only difference is these predictors map to $\{0, 1\}$, rather than $\{-1, +1\}$.

(b) Show that

$$\min_{h \in \mathcal{H}} \sum_{i=1}^{n} \frac{1}{n} \mathbb{1}[2h(x_i) - 1 \ne y_i] \;\le\; \frac{n - L}{n},$$

where L is the length of the longest contiguous subsequence of examples having the same labels, meaning $y_j = y_{j+1} = \cdots = y_{j+L-1}$ for some j.

(c) Show that there exists an integer m, reals $(\alpha_1, \ldots, \alpha_m)$, and interval classifiers $(h_1, \ldots, h_m)$ with $h_j \in \mathcal{H}$ so that, for every i,

$$y_i = \sum_{j=1}^{m} \alpha_j h_j(x_i).$$

In other words, that there exists a perfect boosted interval classifier.

Solution. *(Your solution here.)*

(a) For any $\boldsymbol{x}_i$, we can always make $h_1(x) := \mathbb{1}[a_1 \le x_i \le b_1]$, where $a_1 \le \boldsymbol{x}_i \le b_1$ for all i or for none of i. Then we have

$$\begin{aligned} \min_{h \in \mathcal{H}} \sum_{i=1}^{n} q_i \mathbb{1}[2h(x_i) - 1 \ne y_i] \;&\le\; \sum_{i=1}^{n} q_i \mathbb{1}[2h_1(x_i) - 1 \ne y_i] \\ &=\; \min \left\{ \sum_{\substack{i \in \{1,\ldots,n\} \\ y_i > 0}} q_i, \sum_{\substack{i \in \{1,\ldots,n\} \\ y_i < 0}} q_i \right\} \end{aligned}$$

(b) For any $\boldsymbol{x}_i$ with the longest contiguous subsequence of examples $y_j = y_{j+1} = \cdots = y_{j+L-1}$ for some j, we can always make $h_1(x) := \mathbb{1}[a_1 \le x_k \le b_1]$, where $a_1 \le \boldsymbol{x}_k \le b_1$ for all $j \le k \le j+L-1$ if $y_j = 1$ or for none of $j \le k \le j + L - 1$ if $y_j = 0$, then we have

$$\begin{aligned} \min_{h \in \mathcal{H}} \sum_{i=1}^{n} \frac{1}{n} \mathbb{1}[2h(x_i) - 1 \ne y_i] \;&\le \sum_{i=1}^{n} \frac{1}{n} \mathbb{1}[2h(x_i) - 1 \ne y_i] \\ &\le \frac{n - L}{n} \end{aligned}$$

(c) Make $m = \sum_{i=1}^{n}(y_i + 1)/2$, $h_i(x) := \mathbb{1}[a_i \leq x \leq b_i]$. And $a_i \leq x \leq b_i$ for only $x_\pi(i)$ where $x_\pi(i)$ is the i^{th} sample whose label $y_\pi(i) = 1$. Then we have a perfect boosted interval classifier, for every i

$$y_i = \sum_{j=1}^{m} \alpha_j h_j(x_i).$$

4. **Variational Autoencoders.**

In this problem you will implement a Variational Autoencoder (VAE) to model points sampled from an unknown distribution. This will be done by constructing an encoder network and a decoder network. The encoder network $f_{\text{enc}} : X \subset \mathbb{R}^2 \to \mathbb{R}^h \times \mathbb{R}^h$ takes as input a point $\boldsymbol{x}$ from the input space and outputs parameters $(\boldsymbol{\mu}, \boldsymbol{\xi})$ where $\boldsymbol{\xi} = \log \boldsymbol{\sigma}^2$. The decoder network $f_{\text{dec}} : \mathbb{R}^h \to \mathbb{R}^2$ takes as input a latent vector $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ and outputs an element $\hat{\boldsymbol{x}} \in \mathbb{R}^2$ that we would hope is similar to members of the input space X. You will train this model by minimizing the (regularized) empirical risk

$$\widehat{\mathcal{R}}_{\text{VAE}}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_{\text{dec}}(f_{\text{enc}}(\boldsymbol{x})), \boldsymbol{x}) + \lambda \text{KL}\left(\mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}_i), \exp(\boldsymbol{\xi}(\boldsymbol{x}_i)/2)), \mathcal{N}(0, I)\right).$$

(a) Let $\boldsymbol{\Sigma} = \text{diag}(\boldsymbol{\sigma}^2)$. In your written submission show that

$$\text{KL}\left(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \mathcal{N}(0, I)\right) = -\frac{1}{2}\left[h + \sum_{j=1}^{h}\left(\log \sigma_j^2 - \mu_j^2 - \sigma_j^2\right)\right],$$

where $\text{KL}(p, q) = \int p(\boldsymbol{x}) \ln \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} \, \mathrm{d}\boldsymbol{x}$ is the *KL divergence* between two densities p, q. You may use the fact that the KL-divergence between two h-dimensional normal distributions $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0), \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ is given by

$$\text{KL}(\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0), \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)) = \frac{1}{2}\left(\text{tr}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) - h + \ln \frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_0|}\right).$$

(b) Use the empirical risk discussed above to implement a VAE in the class `VAE`. Use ReLU activations between each layer, except on the last layer of the decoder use sigmoid. Use the ADAM optimizer to optimize in the `step()` function. Make use of the PyTorch library for this. Use `torch.optim.Adam()`, there is no need to implement it yourself. Please refer to the docstrings in hw6.py for more implementation details.

(c) Implement the `fit` function using the `net.step()` function from the `VAE` class. See the docstrings in `hw6.py` for more details.

(d) Fit a `VAE` on the data generated by `generate_data` in `hw6_utils.py`. Use a learning rate $\eta = 0.01$, latent space dimension $h = 6$, KL-divergence scaling factor $\lambda = 5 \times 10^{-5}$, and train for 8000 iterations. Use least squares as the loss, that is, let $\ell(f(\boldsymbol{x}), \hat{\boldsymbol{x}}) = \|f(\boldsymbol{x}) - \hat{\boldsymbol{x}}\|_2^2$. Include separate plots of each of the following in your written submission:

i. Your empirical risk $\widehat{\mathcal{R}}_{\text{VAE}}$ on the training data vs iteration count;

ii. The data points $(\boldsymbol{x})_{i=1}^n$ along with their encoded and decoded approximations $\hat{\boldsymbol{x}} = f_{\text{dec}}(f_{\text{enc}}(\boldsymbol{x}))$;

iii. The data points $(\boldsymbol{x})_{i=1}^n$ along with their encoded and decoded approximations $\hat{\boldsymbol{x}}$, and n generated points $f_{\text{dec}}(\boldsymbol{z})$ where $\boldsymbol{z} \sim \mathcal{N}(0, I)$.

After you are done training, save your neural network to a file using `torch.save(model.cpu().state_dict(), "vae.pb")`. You will submit this file to the autograder with your code submission.

(e) What is the difference between the $\hat{\boldsymbol{x}}$ and $f_{\text{dec}}(\boldsymbol{z})$ in general? Why are they different in the plots?

(f) Repeat part (d) except this time use L1 as your loss, that is let $\ell(f(\boldsymbol{x}), \hat{\boldsymbol{x}}) = \|f(\boldsymbol{x}) - \hat{\boldsymbol{x}}\|_1 = \sum_{j=1}^{2} |x_j - \hat{x}_j|$. Again, be sure to include the plots in your written submission.

(g) Fit a `VAE` with $\lambda \in \{1, 0.01, 0.001\}$ and L1 loss on the same data again , but this time only plot (iii) from part (d). Discuss your results. Do you expect the VAE to generalize more closely to the true distribution better or worse as you increase λ? Out of all of the parameters you tried including 5×10^{-5}, which λ parameter seems to give the right balance? Be sure to provide a brief justification for your choice.

Solution.

(a)

$$\begin{aligned}
\mathrm{KL}\left(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \mathcal{N}(0, I)\right) &= \frac{1}{2}\left(\mathrm{tr}(I^{-1}\boldsymbol{\Sigma}) + (-\boldsymbol{\mu}_0)^\top(-\boldsymbol{\mu}_0) - h + \ln\frac{1}{\prod\sigma_j^2}\right) \\
&= \frac{1}{2}\left[\sum_{j=1}^{h}\sigma_j^2 - h + \sum_{j=1}^{h}\mu_j^2 - \sum_{j=1}^{h}\log\sigma_j^2\right] \\
&= -\frac{1}{2}\left[h + \sum_{j=1}^{h}\left(\log\sigma_j^2 - \mu_j^2 - \sigma_j^2\right)\right],
\end{aligned}$$

(b)

(c)

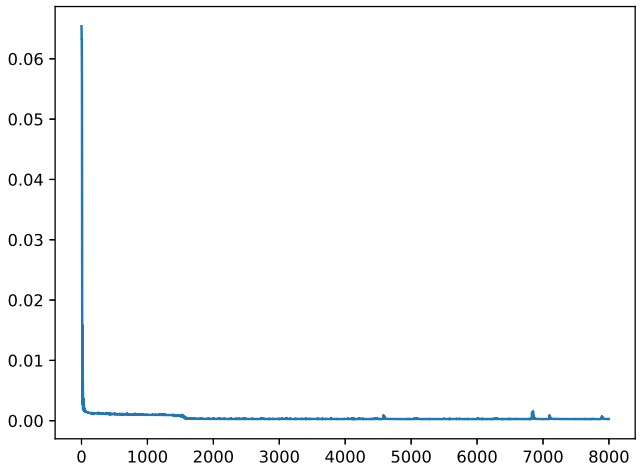

Figure 1: Risk with MSE

(d)

(e) $\hat{\boldsymbol{x}}$ is the reconstructed samples, and $f_{\mathrm{dec}}(\boldsymbol{z})$ is the generated samples, which is generated by the decode network whose input is sampled from normal distribution.

They are different in the plot because the hidden layer, that is the distribution from which the input of decode network sampled, is not standard gaussian. $\boldsymbol{z}$ is sampled from standard gaussian.

(f)

(g) As the λ increases from 5e-5 to 0.01, the VAE gets better. There are less nosiy points between the two clusters. However, when $\lambda = 1$, the algorithm can't converge. I tried to increase the iterations, but still it didn't converge. I also tried 0.1, it also didn't converge. So for all the values I have tried, $\lambda = 0.01$ is the best choice as it fits the data best.

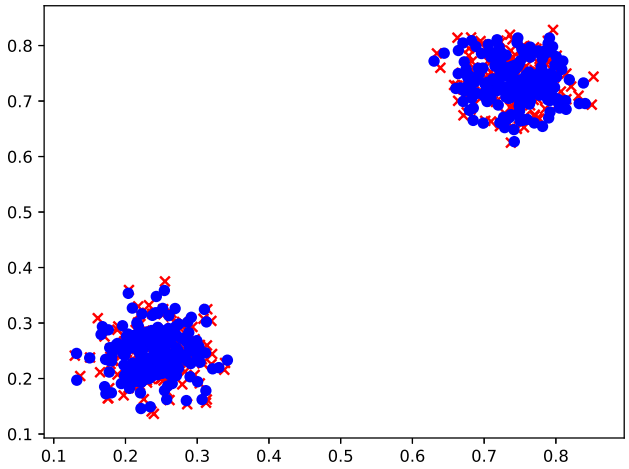

Figure 2: Data points and reconstruction with MSE

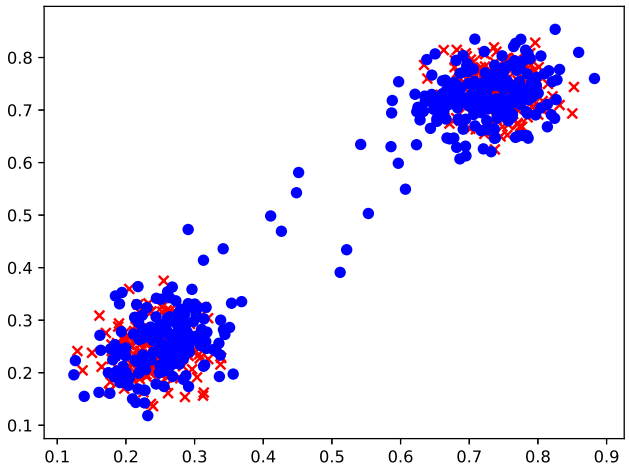

Figure 3: Data points and gen samples with MSE

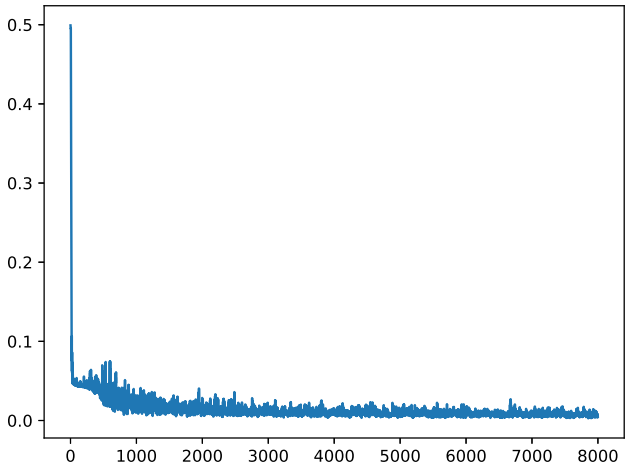

Figure 4: Risk with L1 loss

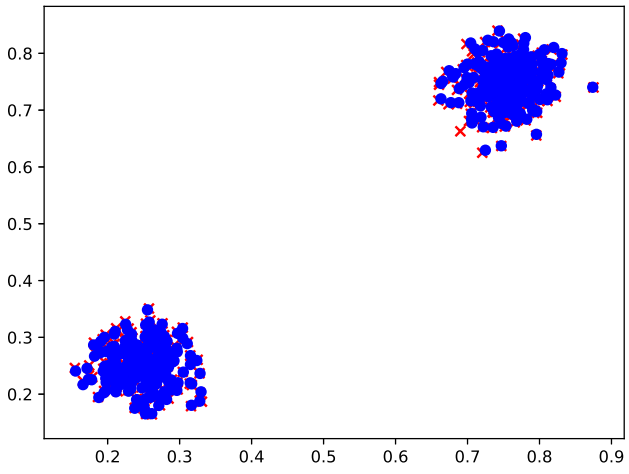

Figure 5: Data points and reconstruction with L1 loss

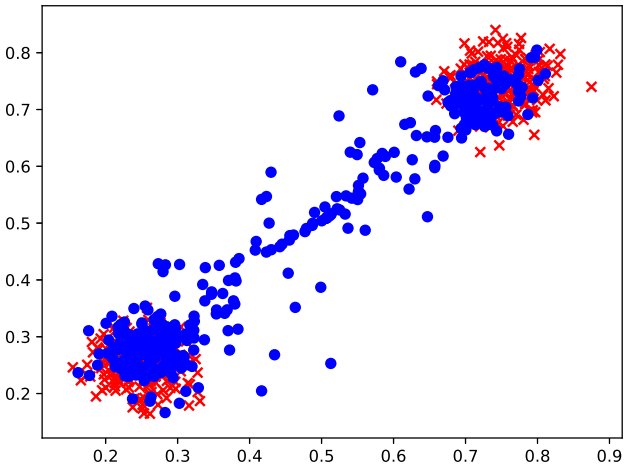

Figure 6: Data points and gen samples with L1 loss

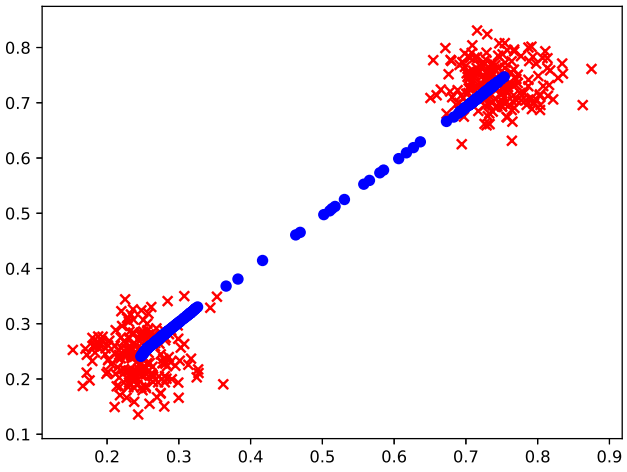

Figure 7: Data points and gen samples with $\lambda = 1$

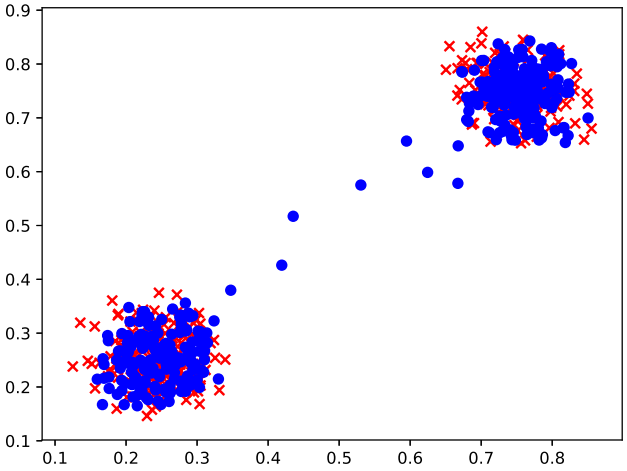

Figure 8: Data points and gen samples with $\lambda = 0.01$

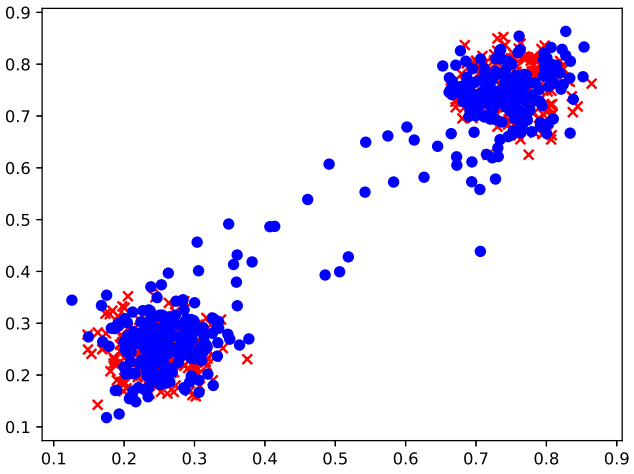

Figure 9: Data points and gen samples with $\lambda = 0.001$

5. **Naive Bayes (Extra credit!).**

Let $\boldsymbol{X} = (X_1, \ldots, X_d)$ be a vector of d binary random variables whose distribution, labeled by a boolean function $f : \{0,1\}^d \to \{0,1\}$. Naive bayes proceeds by forming estimates of various probabilities, and predicting with

$$\hat{f}(\boldsymbol{x}) = \arg\max_y \widehat{\Pr}(Y = y) \prod_{i=1}^{d} \widehat{\Pr}(X_i = x_i | Y = y).$$

(a) Suppose $f(\boldsymbol{x}) = \mathbb{1}\left(\sum_{j=1}^{d} x_j \geq \frac{d}{2}\right)$, and that the various $\widehat{\Pr}$ estimates in $\hat{f}$ are exact. Show that the naive Bayes predictor $\hat{f}(\boldsymbol{x})$ classifies perfectly in this case. For this problem you can assume d is odd.

Hint. Use symmetry arguments to make computing the probabilities easier.

(b) Under the same setup from part(a), construct a boolean function $f : \{0,1\}^3 \to \{0,1\}$ for which naive Bayes will be unable to correctly classify every binary vector $\boldsymbol{x} \in \{0,1\}^3$. Be sure to verify that your construction works.

Solution. *(Your solution here.)*

(a) It is easy to prove that $\widehat{\Pr}(Y = 1) = \widehat{\Pr}(Y = 0) = \widehat{\Pr}(X_i = 1) = \widehat{\Pr}(X_i = 0) = \frac{1}{2}$ So

$$\begin{aligned}
\hat{f}(\boldsymbol{x}) &= \arg\max_y \widehat{\Pr}(Y = y) \prod_{i=1}^{d} \widehat{\Pr}(X_i = x_i | Y = y) \\
&= \arg\max_y \prod_{i=1}^{d} \widehat{\Pr}(Y = y | X_i = x_i) \widehat{\Pr}(X_i = x_i) / \widehat{\Pr}(Y = y) \\
&= \arg\max_y \prod_{i=1}^{d} \widehat{\Pr}(Y = y | X_i = x_i)
\end{aligned}$$

Moreover,

$$\begin{aligned}
\widehat{\Pr}(Y = 0 | X_i = 0) &= \sum_{k=\lceil d/2 \rceil}^{d-1} \binom{k-1}{d-1} (\frac{1}{2})^{k-1} (\frac{1}{2})^{d-k} = \widehat{\Pr}(Y = 1 | X_i = 1) \\
&> \widehat{\Pr}(Y = 0 | X_i = 1) = \sum_{k=\lceil d/2 \rceil}^{d-1} \binom{k}{d-1} (\frac{1}{2})^{k} (\frac{1}{2})^{d-k-1} = \widehat{\Pr}(Y = 1 | X_i = 0)
\end{aligned}$$

Aussme the number of $X_i = 1$ is a, then

$$\begin{aligned}
\frac{\prod_{i=1}^{d} \widehat{\Pr}(Y = 1 | X_i = x_i)}{\prod_{i=1}^{d} \widehat{\Pr}(Y = 0 | X_i = x_i)} &= \frac{\widehat{\Pr}(1|1)^a}{\widehat{\Pr}(0|1)^a} \frac{\widehat{\Pr}(1|0)^{d-a}}{\widehat{\Pr}(0|0)^{d-a}} \\
&= \left(\frac{\widehat{\Pr}(1|1)}{\widehat{\Pr}(1|0)} \right)^{2a-d}
\end{aligned}$$

Because $\frac{\widehat{\Pr}(1|1)}{\widehat{\Pr}(1|0)} > 1$, when $a > \frac{2}{d}$, $f(\boldsymbol{x}) = \hat{f}(\boldsymbol{x}) = 1$, when $a < \frac{2}{d}$, $f(\boldsymbol{x}) = \hat{f}(\boldsymbol{x}) = 0$. So the predictor is perfect.

(b) Asumme $f(\boldsymbol{x})$ is the mapping

(x_1, x_2, x_3)	y
(0, 0, 0)	1
(0, 0, 1)	1
(0, 1, 0)	0
(0, 1, 1)	0
(1, 0, 0)	0
(1, 0, 1)	0
(1, 1, 0)	1
(1, 1, 0)	1

So

$$\widehat{\Pr}(Y = 0|X_i = 0) = \widehat{\Pr}(Y = 1|X_i = 0) = \widehat{\Pr}(Y = 0|X_i = 1) = \widehat{\Pr}(Y = 1|X_i = 1) = \frac{1}{2}$$

As shown in a),

$$\frac{\prod_{i=1}^{d} \widehat{\Pr}(Y = 1|X_i = x_i)}{\prod_{i=1}^{d} \widehat{\Pr}(Y = 0|X_i = x_i)} = 1$$

So the predictor is totally failed.