Generative networks

CS 446

Generative networks — overview

Next two lectures will model probability distributions with deep networks.

1. We'll define generative networks, the basic common sampling scheme.
2. We'll give the first training algorithm: **Variational Autoencoder (VAE)**.
3. Next lecture we'll give another: **Generative Adversarial Network (GAN)**.

Generative networks

Basic generative story

Generative networks provide a way to **sample** from any distribution.

1. Sample $\boldsymbol{z} \sim \mu$, where μ denote san efficiently sampleable distribution (e.g., uniform or Gaussian).
2. Output $g(\boldsymbol{z})$, where $g : \mathbb{R}^d \to \mathbb{R}^m$ is a deep network.

Notation: let $g\#\mu$ (pushforward of μ through g) denote this distribution.

Basic generative story

Generative networks provide a way to **sample** from any distribution.

1. Sample $\boldsymbol{z} \sim \mu$, where μ denote san efficiently sampleable distribution (e.g., uniform or Gaussian).
2. Output $g(\boldsymbol{z})$, where $g : \mathbb{R}^d \to \mathbb{R}^m$ is a deep network.

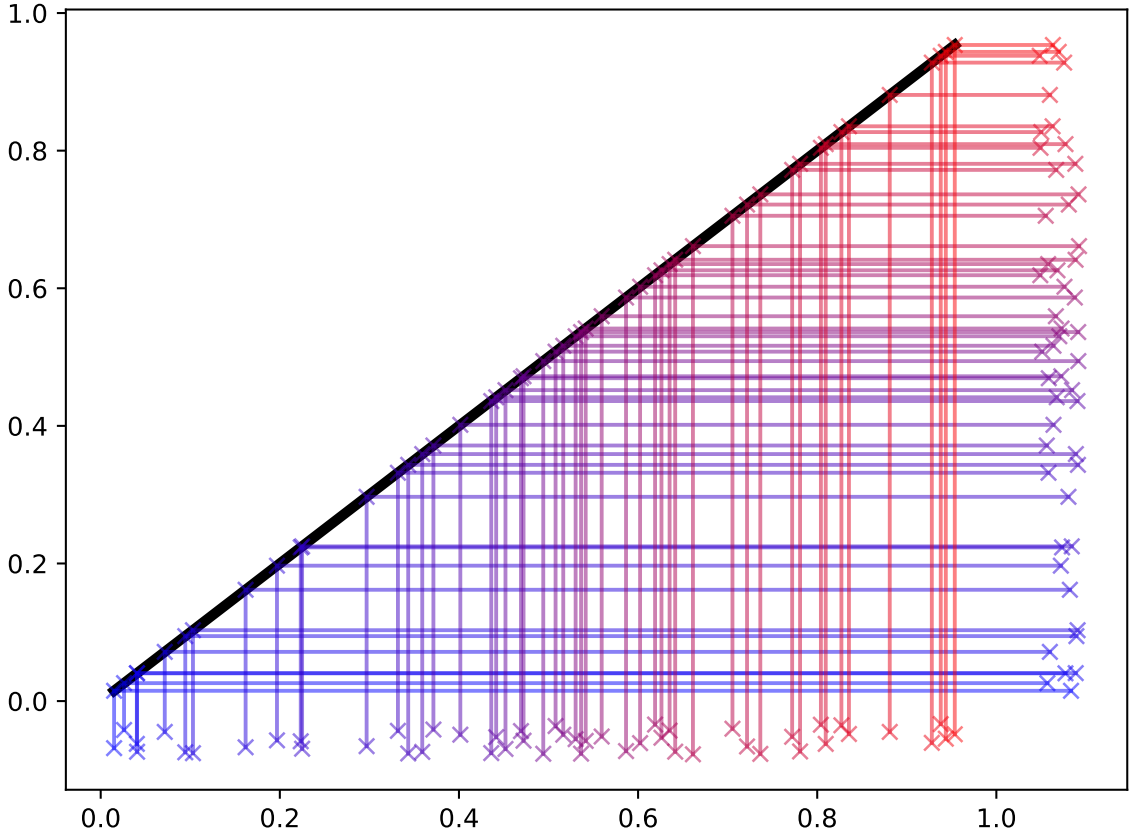
Notation: let $g\#\mu$ (pushforward of μ through g) denote this distribution.

Brief remarks:

- **Can this model any target distribution ν?** Yes, (roughly) for the same reason that g can approximate any $f : \mathbb{R}^d \to \mathbb{R}^m$.
- **Graphical models let us sample and estimate probabilities; what about here?** Nope.

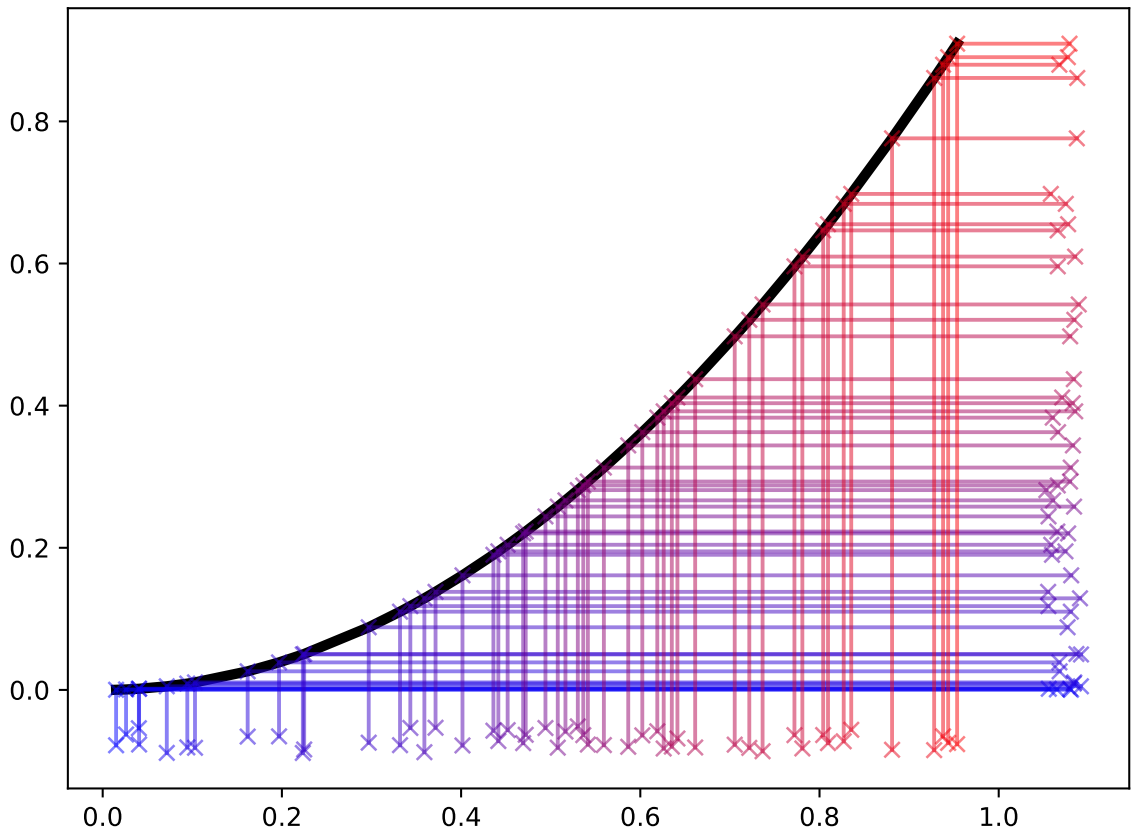
Univariate examples

$g(x) = x$, the identity function,
mapping Uniform$([0, 1])$ to itself.



Univariate examples

$g(x) = x^2$,
mapping Uniform$([0, 1])$ to something $\propto \frac{2}{\sqrt{x}}$.



Pushforwards and random variables

This definition coincides with the formal definition of a random variable

> *A random variable $\boldsymbol{X}$ is a function from a sample space to $\mathbb{R}$; moreover, $\Pr[\boldsymbol{X} = \boldsymbol{x}] = \Pr[X^{-1}(\boldsymbol{x})]$.*

Example. Let (X_1, X_2) denote two random dice rolls, and $Y = X_1 + X_2$ their sum. Then $\Pr[Y = 3] = \Pr[Y^{-1}(3)] = \Pr[(X_1, X_2) = (1, 2)] + \Pr[(X_1, X_2) = (2, 1)] = 1/18$.

Pushforwards and random variables

This definition coincides with the formal definition of a random variable

> *A random variable $\boldsymbol{X}$ is a function from a sample space to $\mathbb{R}$; moreover, $\Pr[\boldsymbol{X} = \boldsymbol{x}] = \Pr[X^{-1}(\boldsymbol{x})]$.*

Example. Let (X_1, X_2) denote two random dice rolls, and $Y = X_1 + X_2$ their sum. Then $\Pr[Y = 3] = \Pr[Y^{-1}(3)] = \Pr[(X_1, X_2) = (1, 2)] + \Pr[(X_1, X_2) = (2, 1)] = 1/18$.

Therefore: this formalism is just the usual way of constructing a distribution from some other randomness.

Pushforwards and random variables

This definition coincides with the formal definition of a random variable

> *A random variable $\boldsymbol{X}$ is a function from a sample space to $\mathbb{R}$; moreover, $\Pr[\boldsymbol{X} = \boldsymbol{x}] = \Pr[X^{-1}(\boldsymbol{x})]$.*

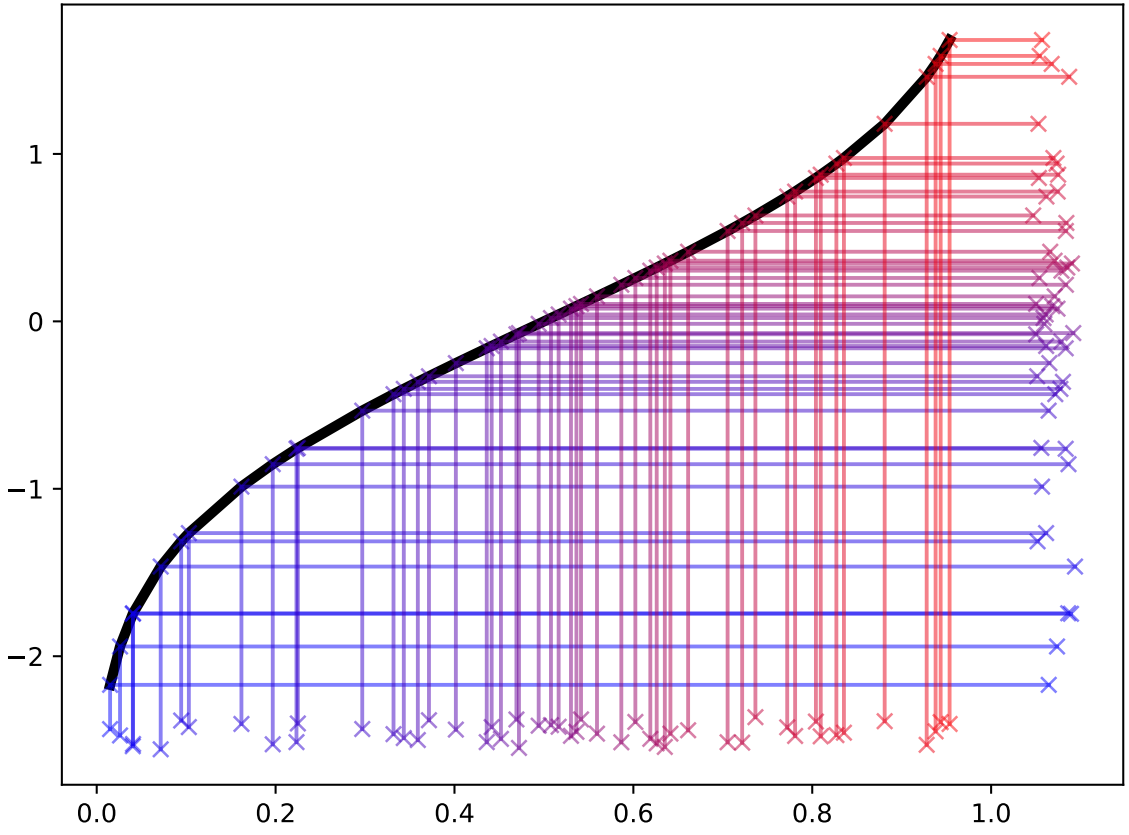
Example. Let (X_1, X_2) denote two random dice rolls, and $Y = X_1 + X_2$ their sum. Then $\Pr[Y = 3] = \Pr[Y^{-1}(3)] = \Pr[(X_1, X_2) = (1, 2)] + \Pr[(X_1, X_2) = (2, 1)] = 1/18$.

Therefore: this formalism is just the usual way of constructing a distribution from some other randomness.

Remark. If F is the CDF of some univariate distribution ν, then $F^{-1}\#\text{Uniform}([0, 1])$ has the same distribution as ν. For multivariate distributions, have something similar.

Univariate examples

g is inverse CDF of Gaussian,
input distribution is Uniform$([0, 1])$ and output is Gaussian.



Generative networks: bottom line.

Generative networks provide a way to **sample** from any distribution.

1. Sample $\boldsymbol{z} \sim \mu$, where μ denote san efficiently sampleable distribution (e.g., uniform or Gaussian).
2. Output $g(\boldsymbol{z})$, where $g : \mathbb{R}^d \to \mathbb{R}^m$ is a deep network.

Notation: let $g\#\mu$ (pushforward of μ through g) denote this distribution.

Generative networks: bottom line.

Generative networks provide a way to **sample** from any distribution.

1. Sample $\boldsymbol{z} \sim \mu$, where μ denote san efficiently sampleable distribution (e.g., uniform or Gaussian).
2. Output $g(\boldsymbol{z})$, where $g : \mathbb{R}^d \to \mathbb{R}^m$ is a deep network.

Notation: let $g\#\mu$ (pushforward of μ through g) denote this distribution.

Remarks.

- This is a sampling model; it is painful to estimate probabilities with it. (People do so but it's shaky.)
- Alternatively, we could build a neural network to estimate probabilities; but sampling from this would in turn be painful.
- **Next:** we'll give training methods (to fit g to a sample from a target distribution): VAE and GAN.

VAE warm-up: encoders and decoders

Linear encoding: PCA

Encoding/decoding was one motivation for unsupervised learning.
PCA gives the optimal linear encoding:

Linear encoding: PCA

Encoding/decoding was one motivation for unsupervised learning.
PCA gives the optimal linear encoding:

PCA Theorem (part of it). Given $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\mathsf{T}$ and $k \leq r$,

$$\min_{\substack{\boldsymbol{E} \in \mathbb{R}^{d \times k} \\ \boldsymbol{D} \in \mathbb{R}^{k \times d}}} \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{E}\boldsymbol{D}\|_\mathsf{F}^2 = \left\|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{V}_k\boldsymbol{V}_k^\mathsf{T}\right\|_\mathsf{F}^2.$$

Encoding with deep networks

Let encoders $\mathcal{E}$ and decoders $\mathcal{D}$ denote families of deep networks from $\mathbb{R}^d$ to $\mathbb{R}^k$ and back.

$$\min_{\substack{f \in \mathcal{E} \\ g \in \mathcal{D}}} \frac{1}{n} \sum_{i=1}^{n} \|\boldsymbol{x}_i - g(f(\boldsymbol{x}_i))\|_2^2.$$

Encoding with deep networks

Let encoders $\mathcal{E}$ and decoders $\mathcal{D}$ denote families of deep networks from $\mathbb{R}^d$ to $\mathbb{R}^k$ and back.

$$\min_{\substack{f \in \mathcal{E} \\ g \in \mathcal{D}}} \frac{1}{n} \sum_{i=1}^{n} \|\boldsymbol{x}_i - g(f(\boldsymbol{x}_i))\|_2^2.$$

Remarks.

- This is called an **autoencoder**.
- $\mathbb{R}^k$ is the **latent space**, $f(\boldsymbol{x}) \in \mathbb{R}^k$ is **latent representation of x**.
- As with PCA, can relate weights of f and g; e.g., g uses the transposes of f's matrices. (Not necessary, though.)
- Does small k enforce positive error?

  No; *powerful $\mathcal{E}$ and $\mathcal{D}$ can do $f(\boldsymbol{x}_i) = 1/i$ and $g(1/i) = \boldsymbol{x}_i$.*

- Therefore we must regularize in some other way!
- Does squared error matter?

  Not really; it's a choice like everything else in ML.

VAEs — ERM perspective

Recall: generative goal

- We have a sample $(\boldsymbol{x}_i)_{i=1}^n \sim \nu$.
- We want to find g so that $(g(\boldsymbol{z}_i))_{i=1}^n \approx (\boldsymbol{x}_i)_{i=1}^n$, where $(\boldsymbol{z}_i)_{i=1}^n \sim \mu$ (Gaussian, etc).

Recall: generative goal

- We have a sample $(\boldsymbol{x}_i)_{i=1}^n \sim \nu$.
- We want to find g so that $(g(\boldsymbol{z}_i))_{i=1}^n \approx (\boldsymbol{x}_i)_{i=1}^n$, where $(\boldsymbol{z}_i)_{i=1}^n \sim \mu$ (Gaussian, etc).

Issues:

- Before, we've strived for $g(\boldsymbol{x}_i) \approx \boldsymbol{z}_i$; now, though we want **distributions** $g\#\mu$ and ν to match.
- Said another way: we don't know how to match up $(g(\boldsymbol{z}_1), \ldots, g(\boldsymbol{z}_n))$ and $(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n)$.
- VAE solves this issue one way; GAN will solve it another.

Variational Autoencoder (VAE) setup

Autoencoder: deterministically map $\boldsymbol{x}_i$ to "latent" $\boldsymbol{z}_i$, back to $\boldsymbol{x}_i$.

Variational Autoencoder (VAE) setup

Autoencoder: deterministically map $\boldsymbol{x}_i$ to "latent" $\boldsymbol{z}_i$, back to $\boldsymbol{x}_i$.

Generative setting:

- We want $\boldsymbol{z}_i$ random, and to map to random $\boldsymbol{x}_i$.
- VAE solution:
  - Encode: $\boldsymbol{y}_i := f(\boldsymbol{x}_i)$.
  - Reparameterize: Sample $\boldsymbol{z}_i \sim \mathcal{N}$ which is parameterized by $\boldsymbol{y}_i$.
  - Decode: Obtain $\hat{\boldsymbol{x}}_i := g(\boldsymbol{z}_i)$, where hopefully $\hat{\boldsymbol{x}}_i \approx \boldsymbol{x}_i$.

Remark. This solves the "matching" problem by associating $\boldsymbol{x}_i$ with $\hat{\boldsymbol{x}}_i$.

VAE

- For $t = 1, 2, \ldots$:
  1. Encode $(\mu_i, \boldsymbol{\xi}_i) = f(\boldsymbol{x}_i)$.
  2. Sample $\boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \exp(\boldsymbol{\xi}_i/2))$.
  3. Decode $\hat{\boldsymbol{x}}_i = g(\boldsymbol{z}_i)$.
  4. Taking a gradient descent step on

$$R(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i) + \lambda K\left(\mathcal{N}(\boldsymbol{\mu}_i, \exp(\boldsymbol{\xi}/2)), \mathcal{N}(0,1)\right).$$

VAE

- For $t = 1, 2, \ldots$:
  1. Encode $(\mu_i, \boldsymbol{\xi}_i) = f(\boldsymbol{x}_i)$.
  2. Sample $\boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \exp(\boldsymbol{\xi}_i/2))$.
  3. Decode $\hat{\boldsymbol{x}}_i = g(\boldsymbol{z}_i)$.
  4. Taking a gradient descent step on

$$R(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i) + \lambda K\left(\mathcal{N}(\boldsymbol{\mu}_i, \exp(\boldsymbol{\xi}/2)), \mathcal{N}(0,1)\right).$$

Remarks.

- "R" is "reconstruction error" or "risk";
  reasonable choices are ℓ_2, ℓ_1, and binary cross entropy (original).
- KL has two justifications.
  - To generate fresh examples, we output $g(\boldsymbol{z})$ with $\boldsymbol{z} \sim \mathcal{N}(0,1)$. The closer the above KL is to 0, the close this sampling has to approximate the training distribution.
  - It fights against memorization: in the extreme case $\lambda = 0$, we enforce $\boldsymbol{\mu}_i$ is a standard Gaussian for every i, meaning the latent state is independent of the input.

VAE

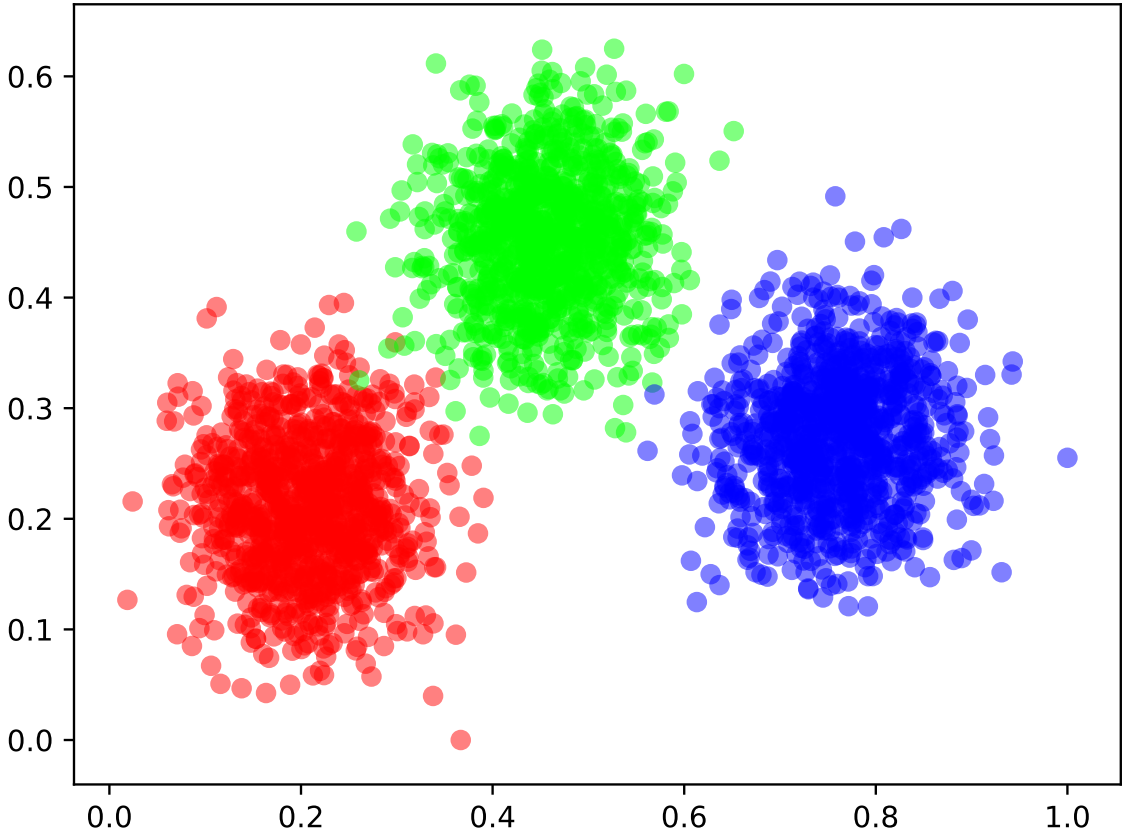
- For $t = 1, 2, \ldots$:
  1. Encode $(\mu_i, \boldsymbol{\xi}_i) = f(\boldsymbol{x}_i)$.
  2. Sample $\boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \exp(\boldsymbol{\xi}_i/2))$.
  3. Decode $\hat{\boldsymbol{x}}_i = g(\boldsymbol{z}_i)$.
  4. Taking a gradient descent step on

$$R(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i) + \lambda K\left(\mathcal{N}(\boldsymbol{\mu}_i, \exp(\boldsymbol{\xi}/2)), \mathcal{N}(0,1)\right).$$

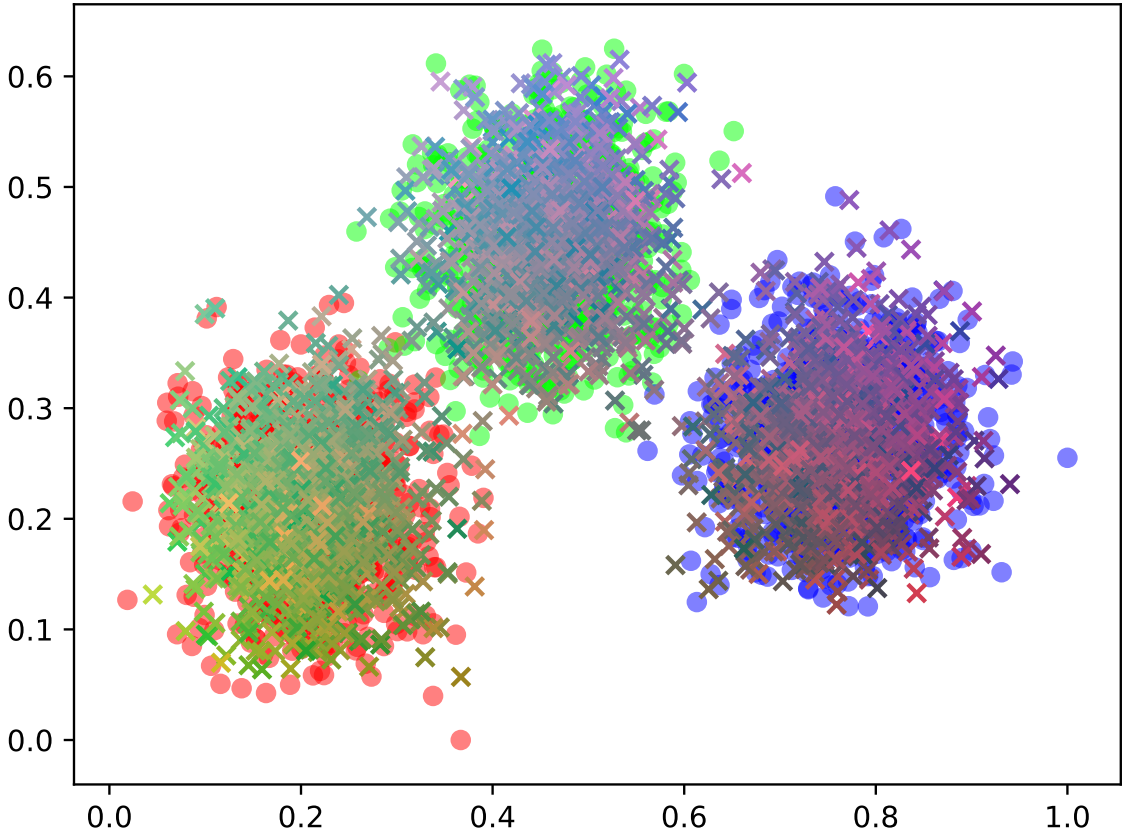
Remarks.

- It is common to use use a stochastic minibatch in every round, and perform the update using a variety of examples. For the homework, the easiest option of using all the data each round will be sufficient.
- Using (rescaled) log variance is practically motivated. Amongst other reasons, it's an easy way to enforce positive variance.
- Overall interpretation: For each $\boldsymbol{x}_i$, construct distribution μ_i (via decoding), so that (stochastic encoding) $\hat{\boldsymbol{x}}_i \sim g\#\mu_i$ and $\boldsymbol{x}_i$ are close, as are μ_i and μ (which means future examples from simply $g\#\mu$ are good).

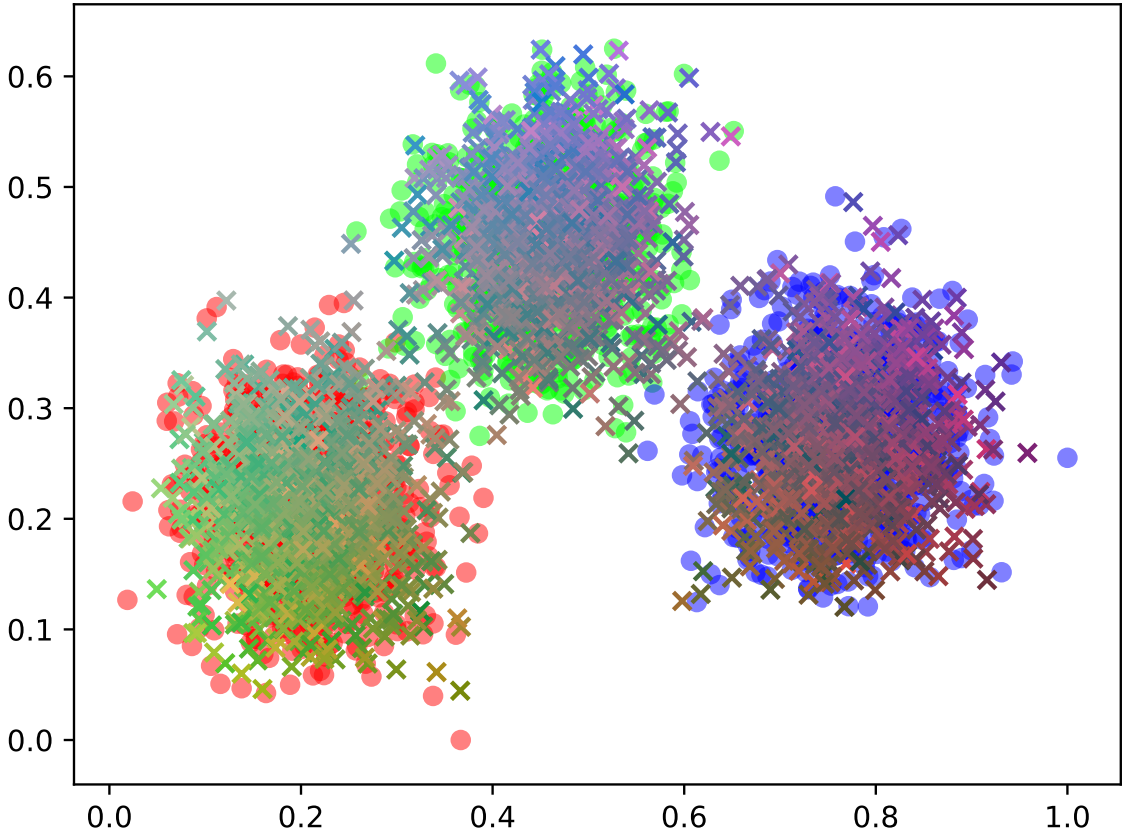
Gaussian mixture examples



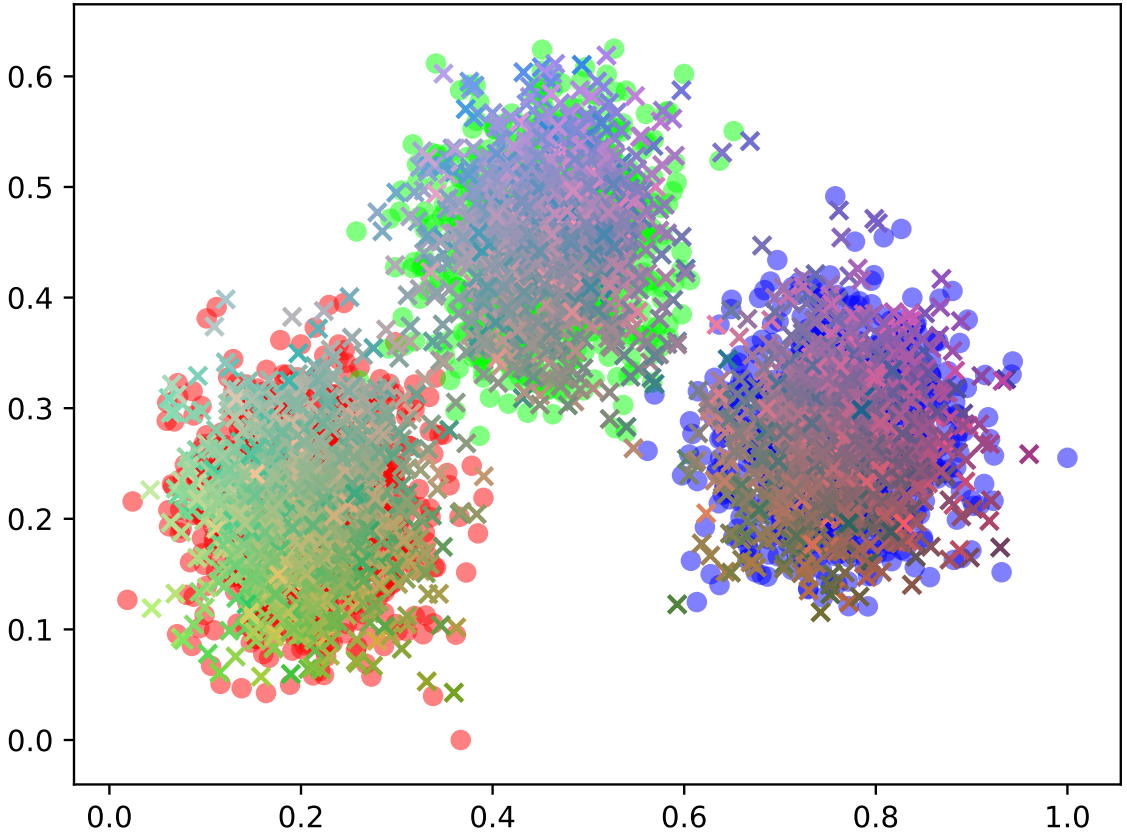
Gaussian mixture examples



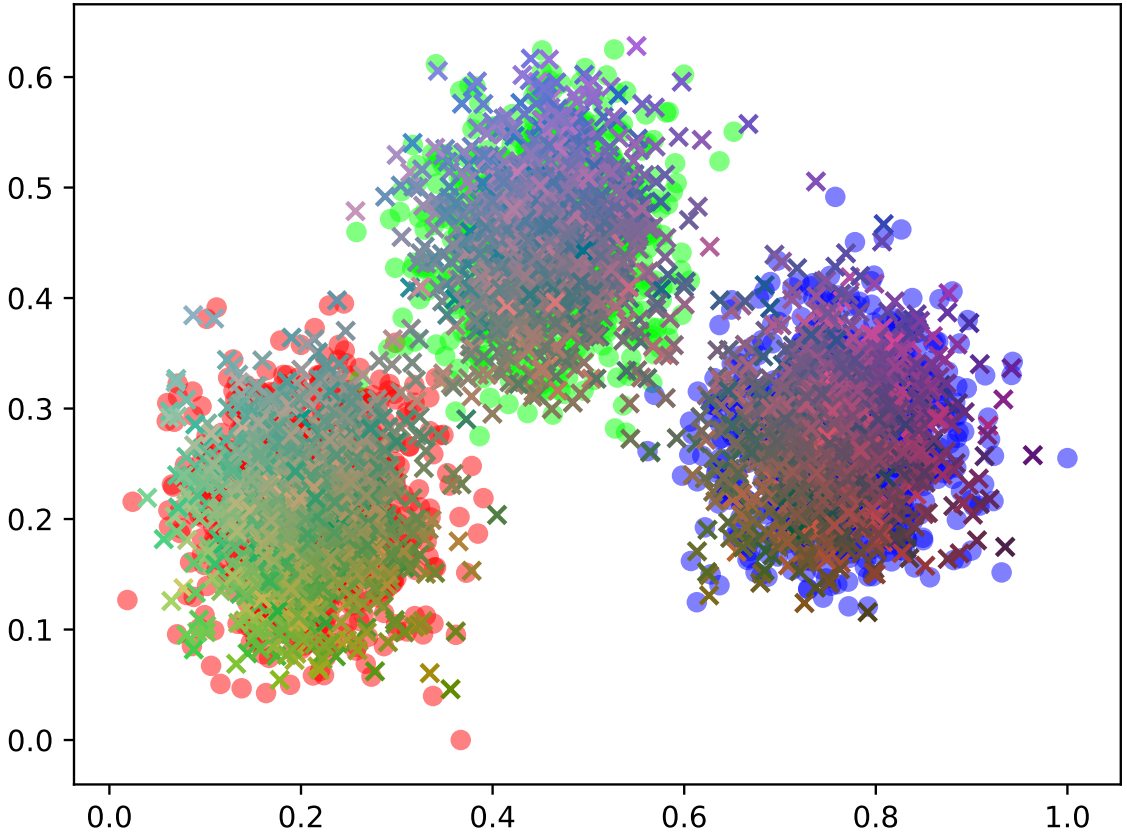
Gaussian mixture examples



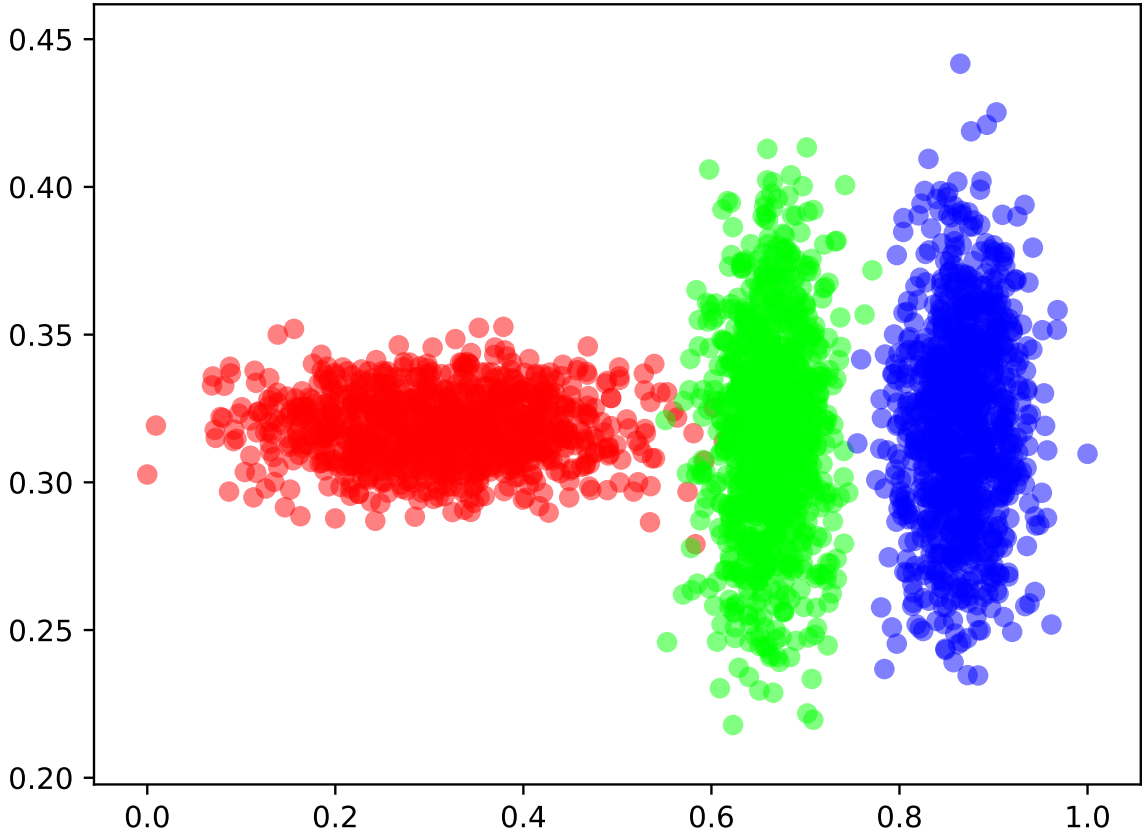
Gaussian mixture examples



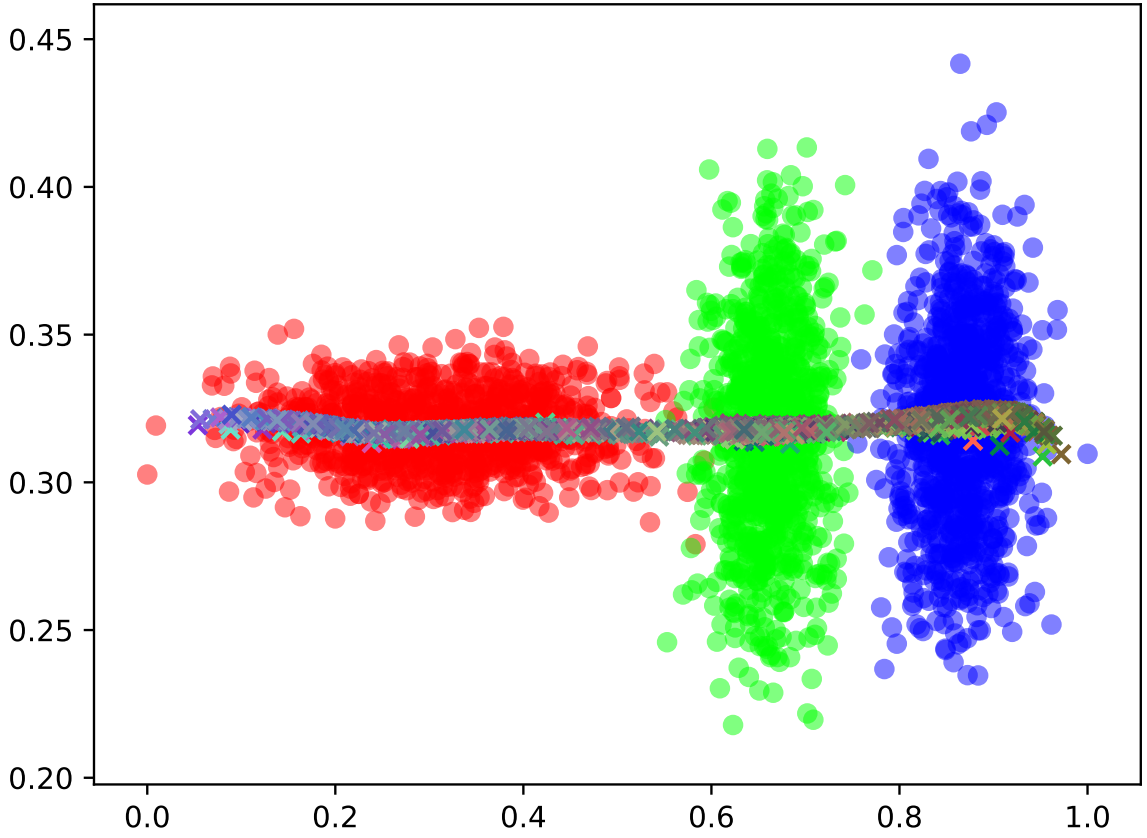
Gaussian mixture examples



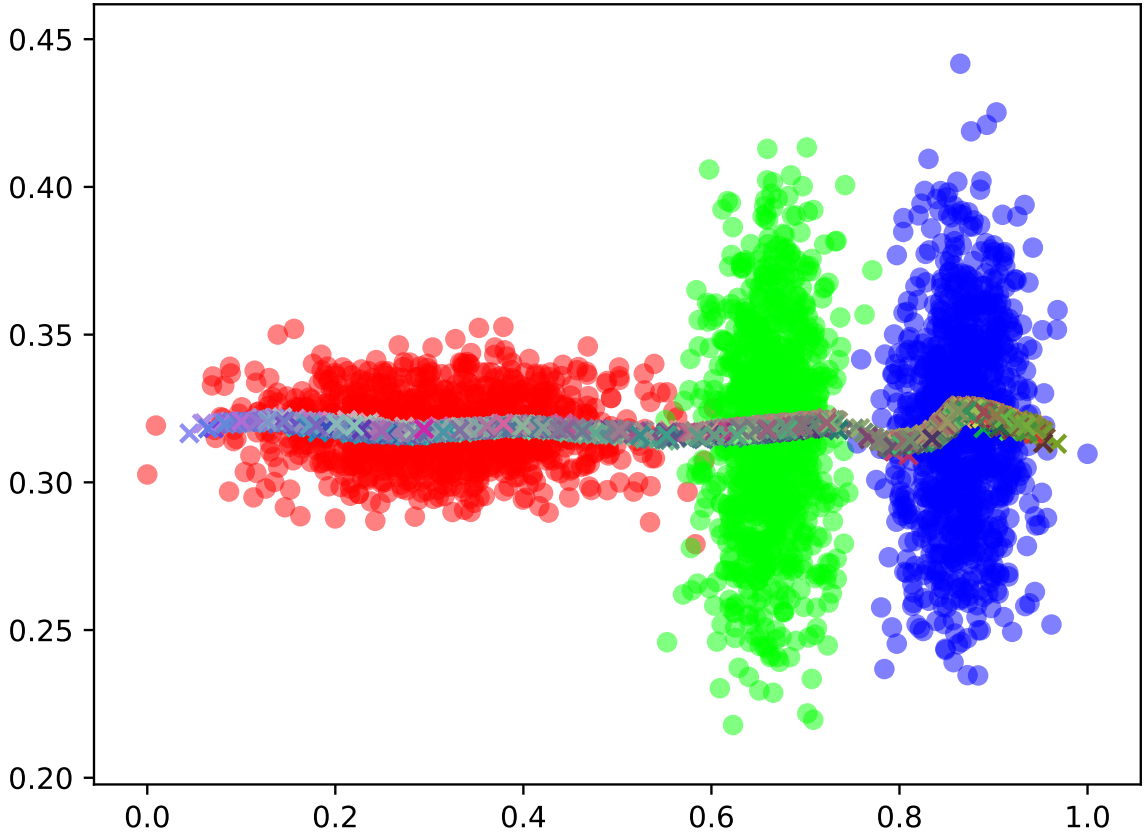
Gaussian mixture examples



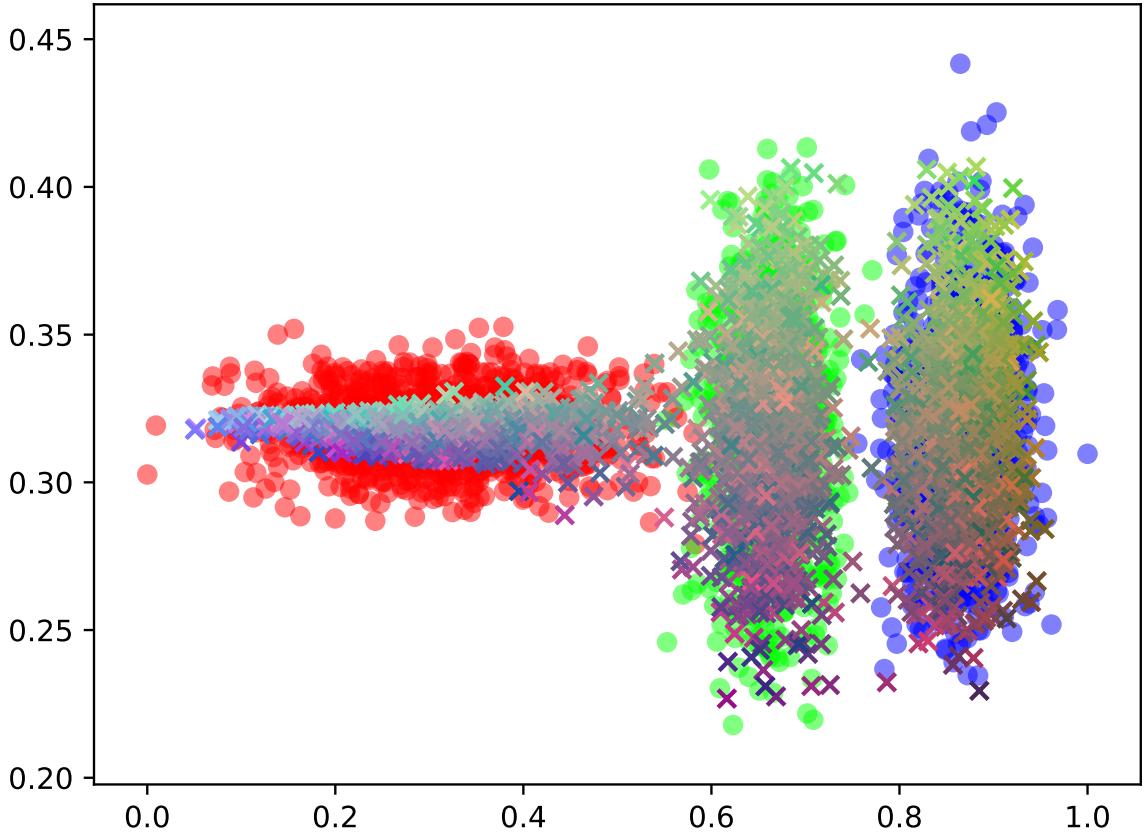
Gaussian mixture examples



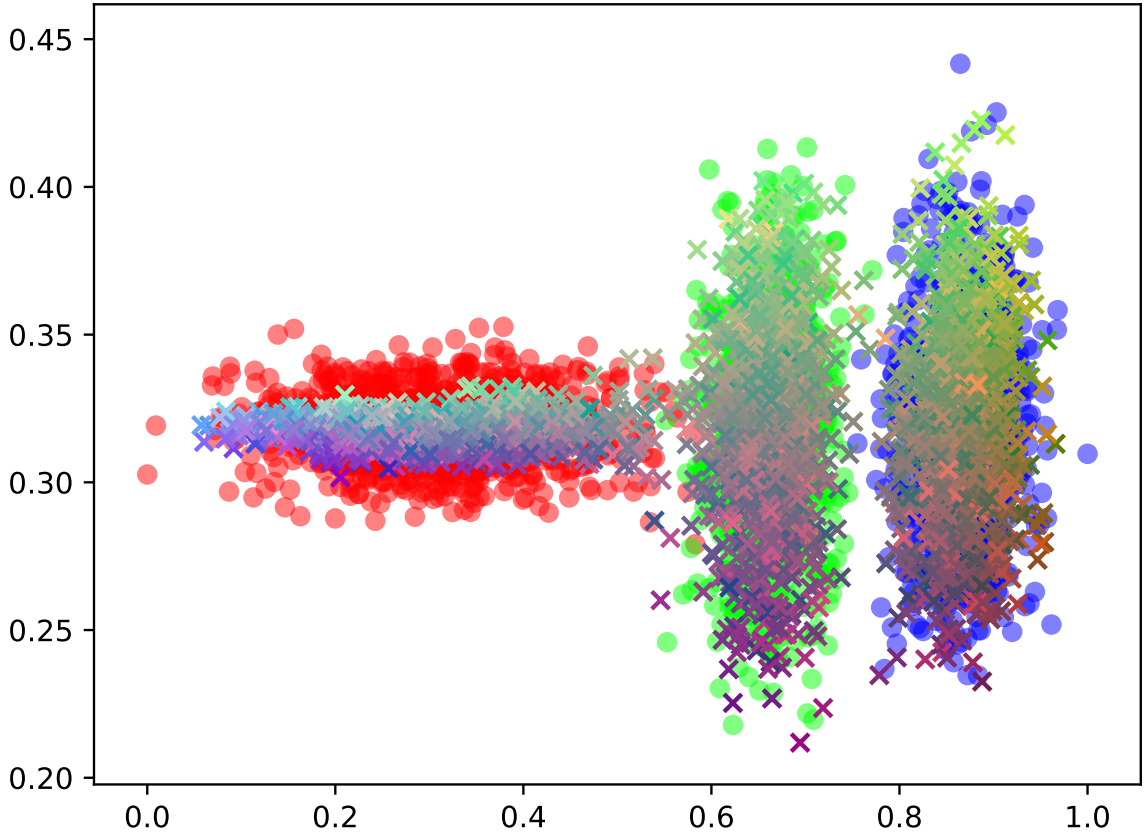
Gaussian mixture examples



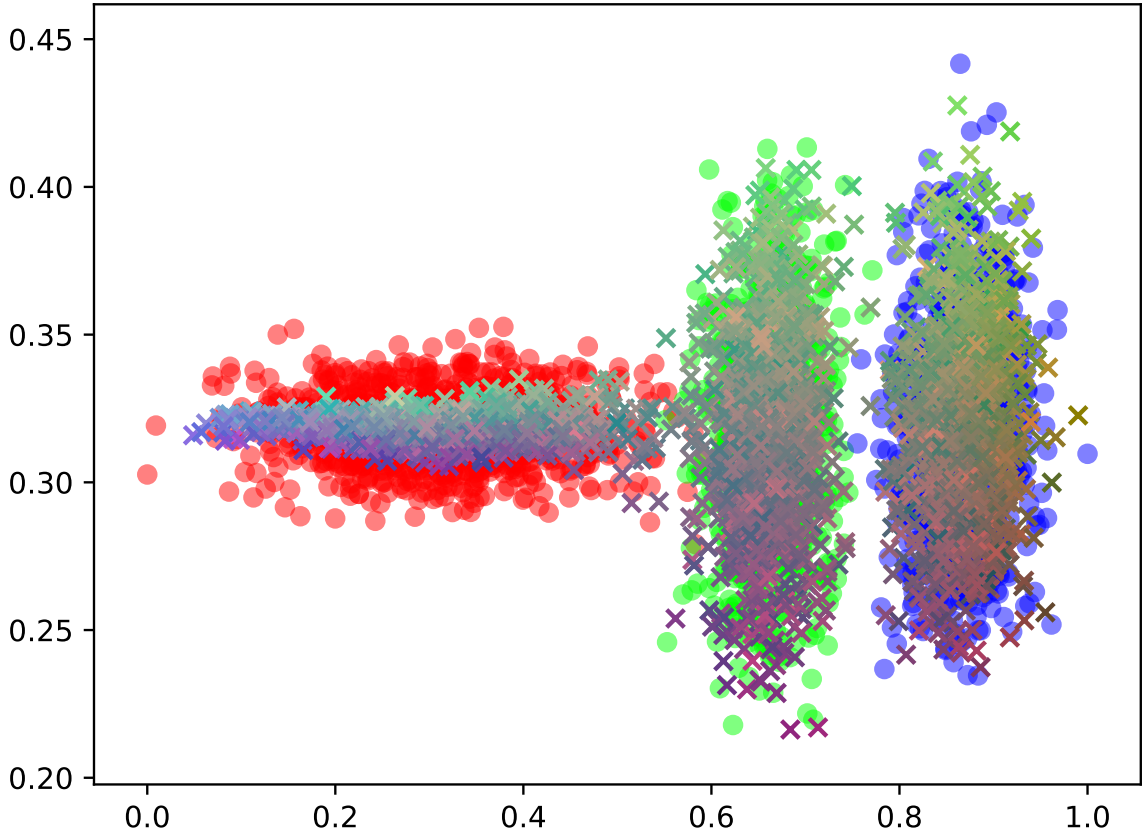
Gaussian mixture examples



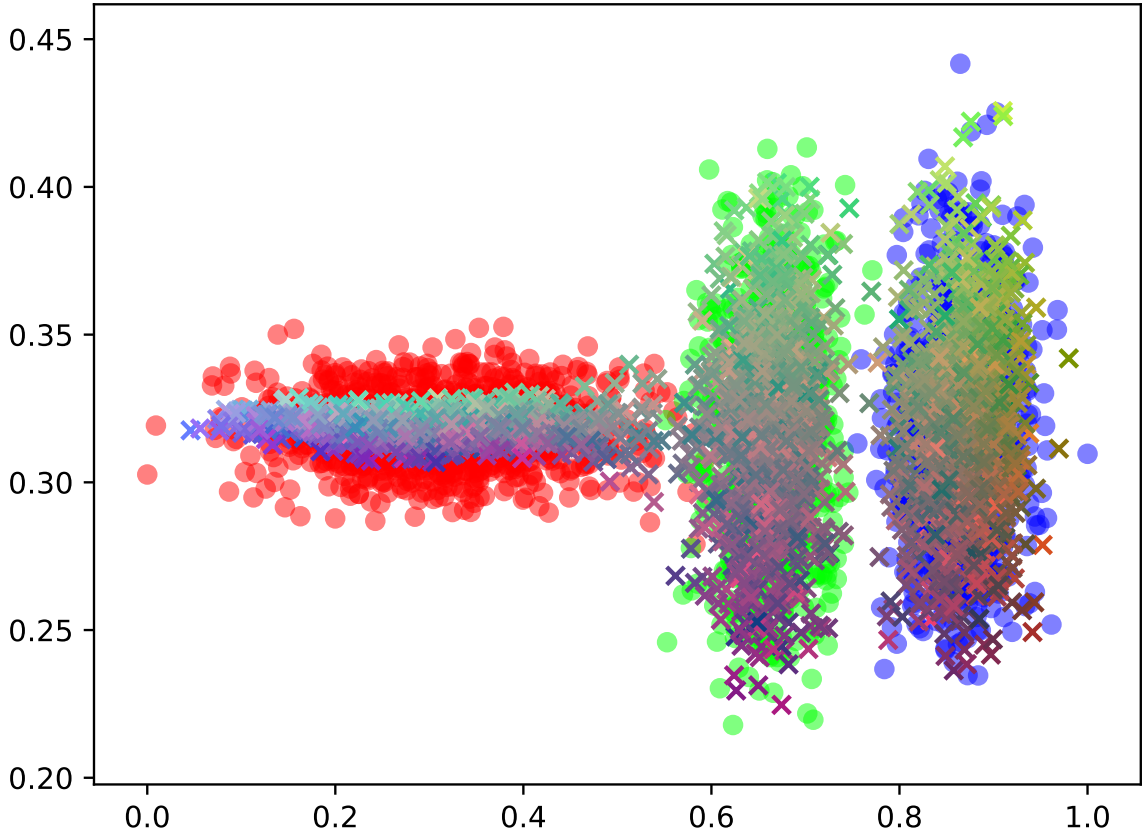
Gaussian mixture examples



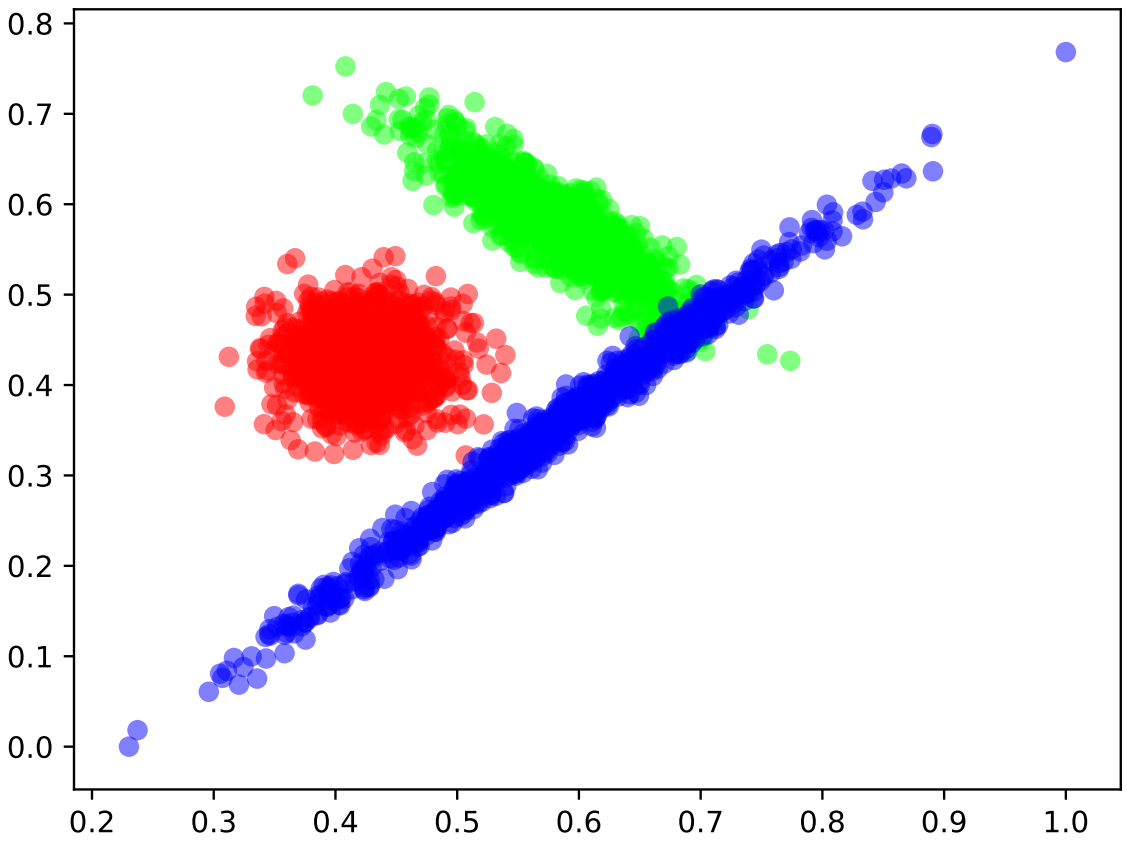
Gaussian mixture examples



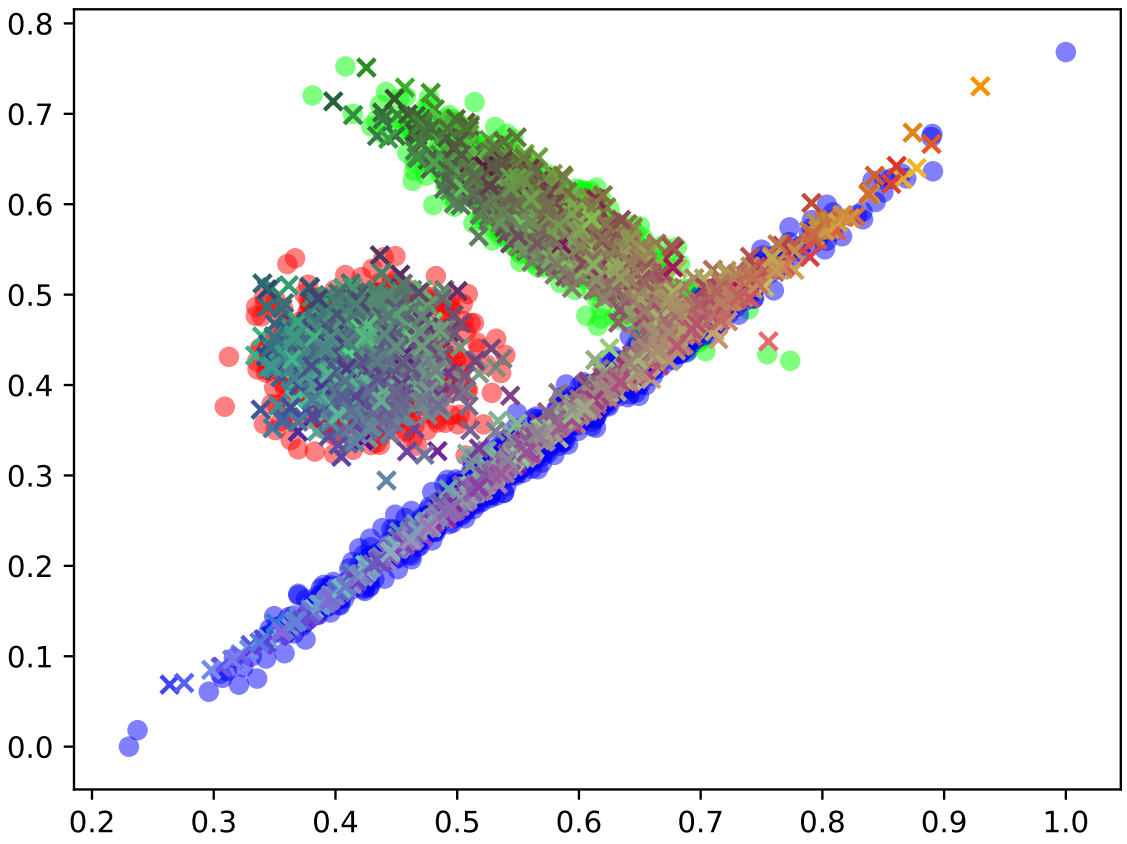
Gaussian mixture examples



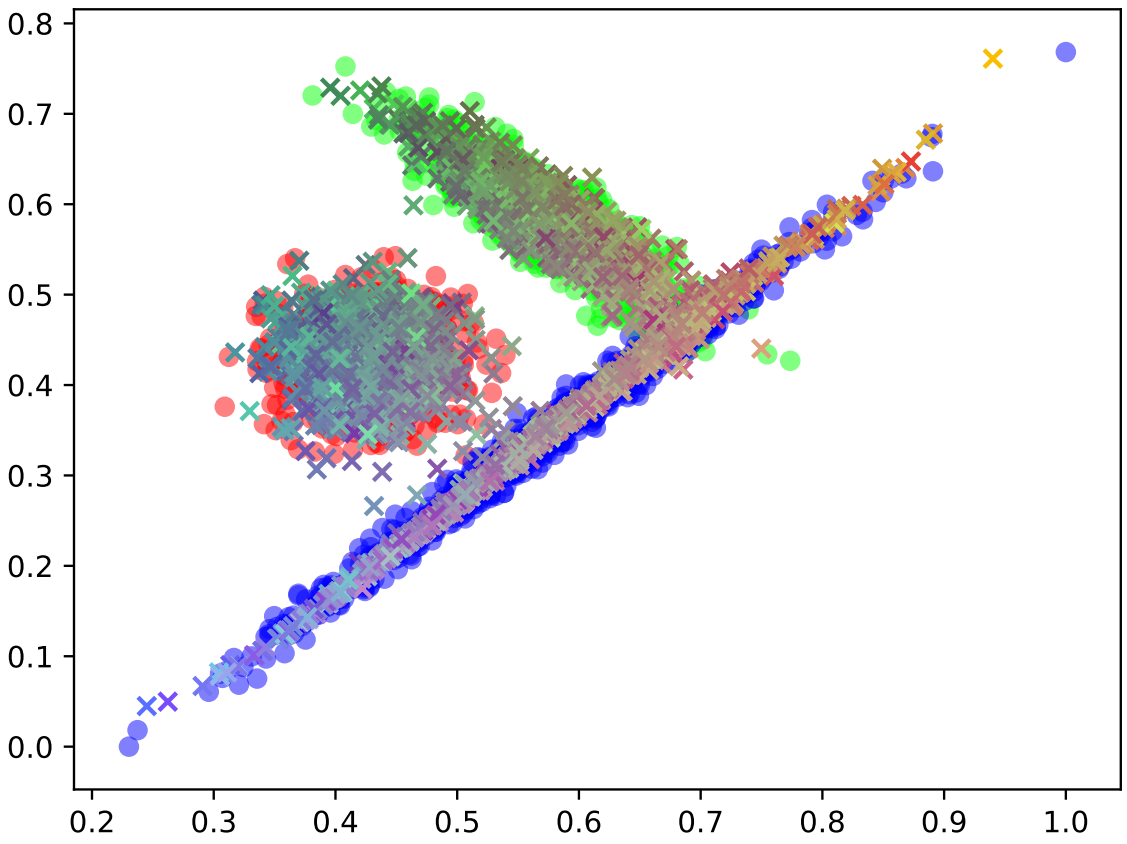
Gaussian mixture examples



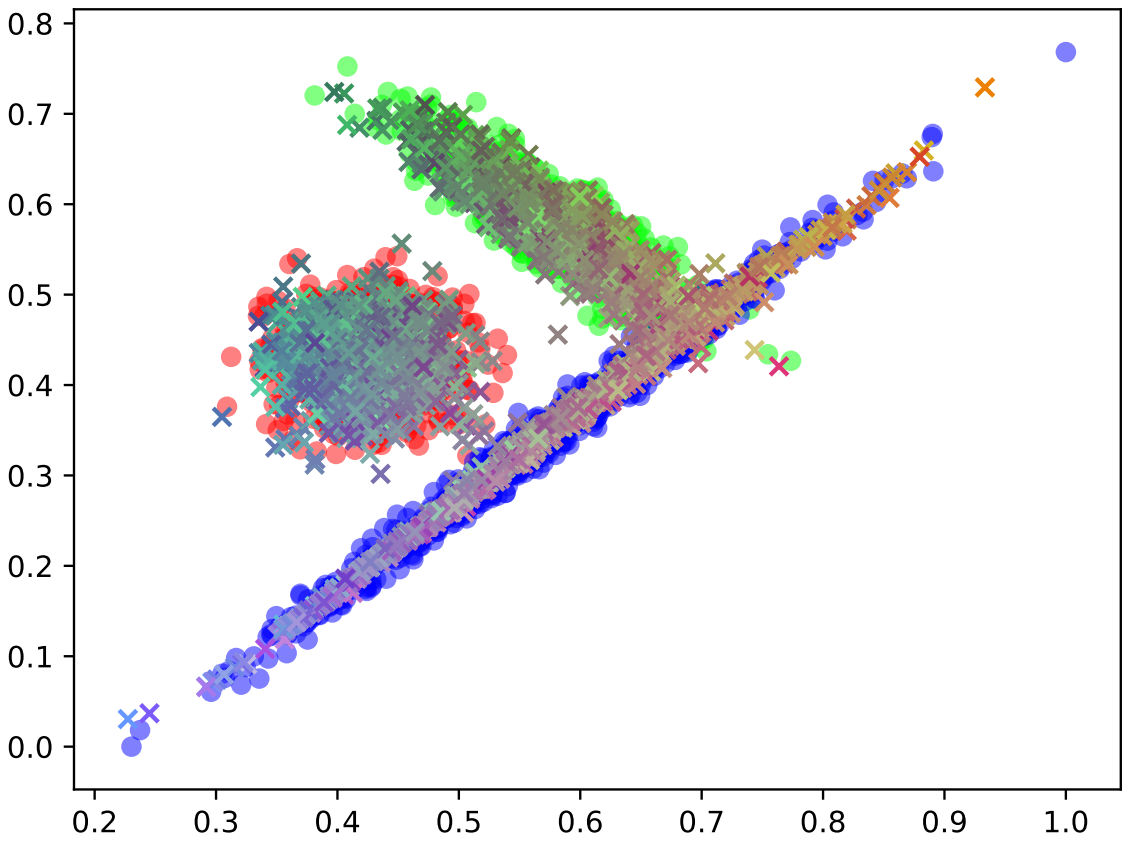
Gaussian mixture examples



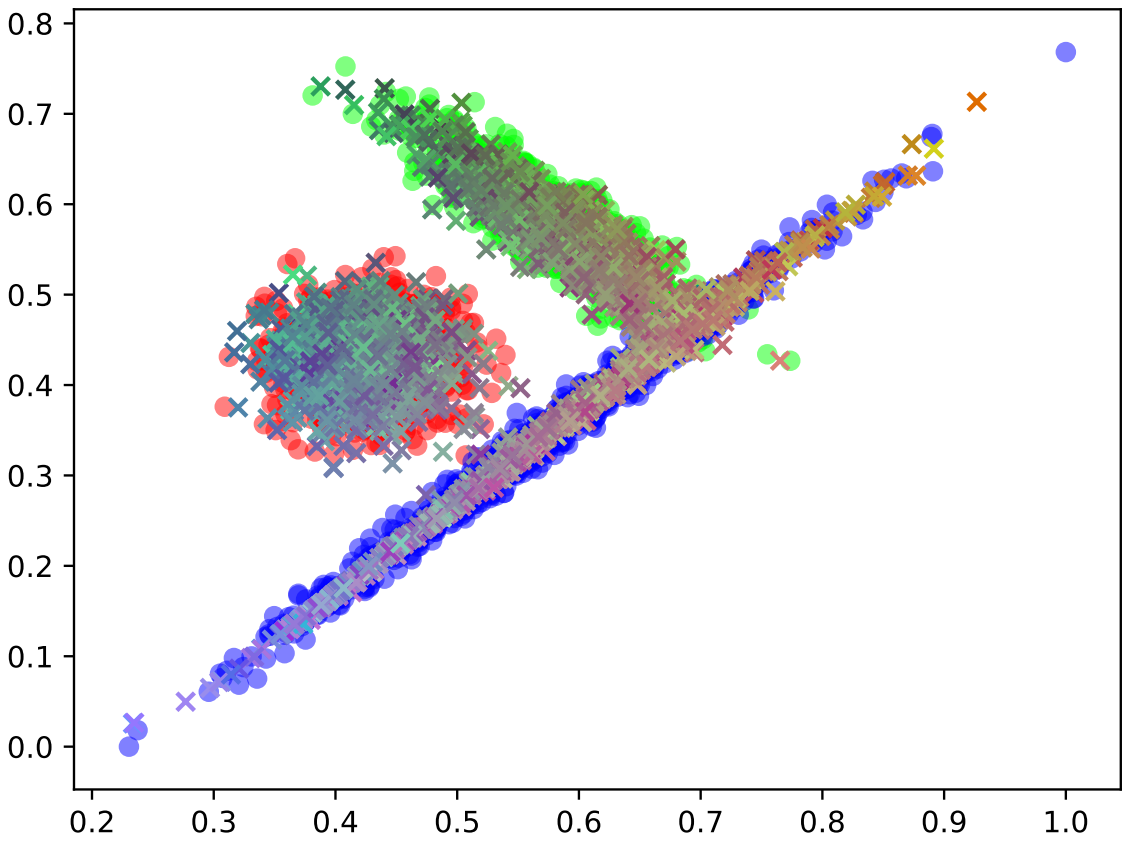
Gaussian mixture examples



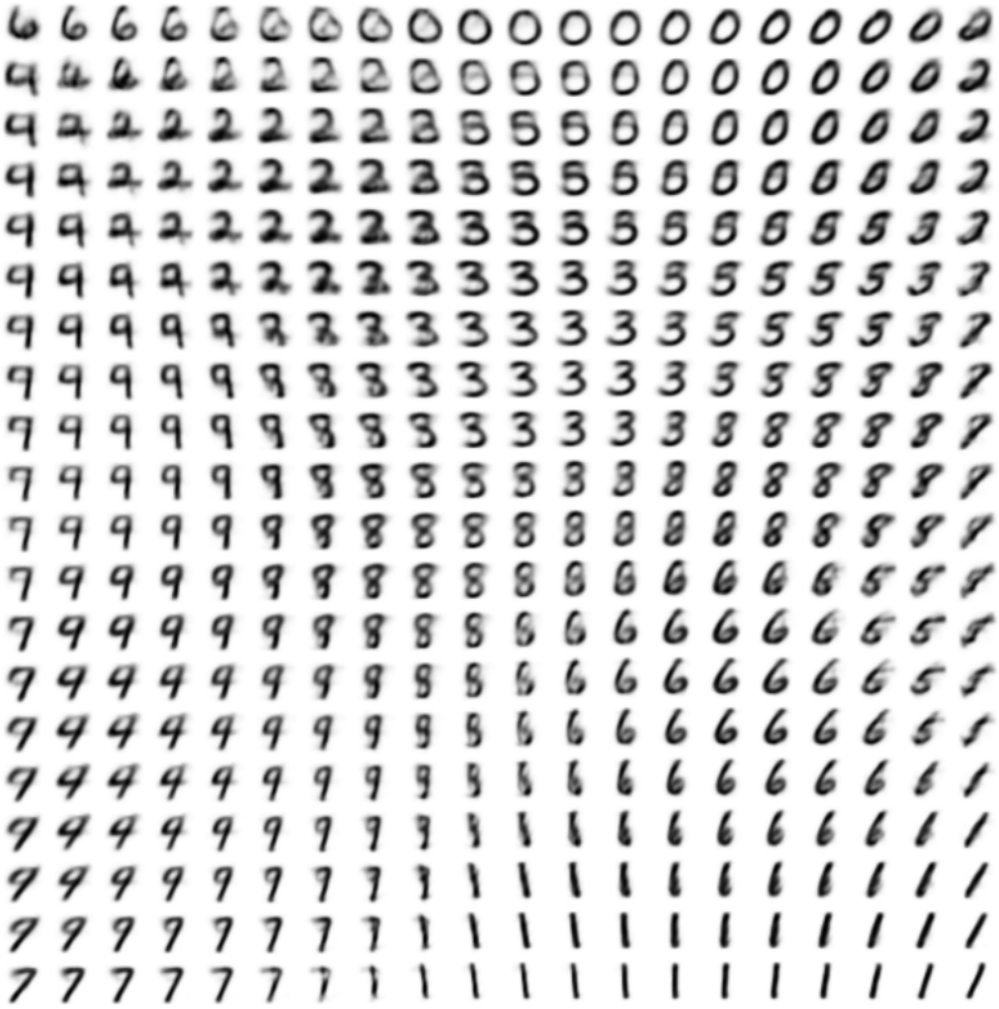
Gaussian mixture examples



Gaussian mixture examples



Latent space



Digit VAE with R^2 latent representation.

VAEs — MLE perspective

Variational E-M recap

Let $q(\boldsymbol{z})$ be distribution over latent parameters. By Jensen's inequality,

$$\begin{aligned}
\ln p(\boldsymbol{x}) &= \ln \int p(\boldsymbol{x}, \boldsymbol{z}) \, \mathrm{d}\boldsymbol{z} \\
&= \ln \int \frac{p(\boldsymbol{x}, \boldsymbol{z}) q(\boldsymbol{z})}{q(\boldsymbol{z})} \, \mathrm{d}\boldsymbol{z} \\
&\geq \int q(\boldsymbol{z}) \ln \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z})} \, \mathrm{d}\boldsymbol{z}
\end{aligned}$$

and this expression is maximized (no inequalities) when $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x})$.

Variational E-M recap

Let $q(\boldsymbol{z})$ be distribution over latent parameters. By Jensen's inequality,

$$\begin{aligned}\ln p(\boldsymbol{x}) &= \ln \int p(\boldsymbol{x}, \boldsymbol{z}) \, \mathrm{d}\boldsymbol{z} \\ &= \ln \int \frac{p(\boldsymbol{x}, \boldsymbol{z}) q(\boldsymbol{z})}{q(\boldsymbol{z})} \, \mathrm{d}\boldsymbol{z} \\ &\geq \int q(\boldsymbol{z}) \ln \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z})} \, \mathrm{d}\boldsymbol{z}\end{aligned}$$

and this expression is maximized (no inequalities) when $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x})$.

- We'll use this to interpret VAE.
- In VAE, q corresponds to encoder, $\ln p$ to decoder.

Variational E-M perspective on VAEs

Write

$$\int q(\boldsymbol{z}) \ln \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z})} \, \mathrm{d}\boldsymbol{z} = \int q(\boldsymbol{z}) \ln \frac{p(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z})}{q(\boldsymbol{z})} \, \mathrm{d}\boldsymbol{z}$$
$$= \int q(\boldsymbol{z}) \ln \frac{p(\boldsymbol{z})}{q(\boldsymbol{z})} \, \mathrm{d}\boldsymbol{z} + \int q(\boldsymbol{z}) \ln p(\boldsymbol{x}|\boldsymbol{z}) \, \mathrm{d}\boldsymbol{z}.$$

q is the encoder distribution on latent $\boldsymbol{z}$.

Variational E-M perspective on VAEs

Write

$$\int q(\boldsymbol{z}) \ln \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z})} \, \mathrm{d}\boldsymbol{z} = \int q(\boldsymbol{z}) \ln \frac{p(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z})}{q(\boldsymbol{z})} \, \mathrm{d}\boldsymbol{z}$$
$$= \int q(\boldsymbol{z}) \ln \frac{p(\boldsymbol{z})}{q(\boldsymbol{z})} \, \mathrm{d}\boldsymbol{z} + \int q(\boldsymbol{z}) \ln p(\boldsymbol{x}|\boldsymbol{z}) \, \mathrm{d}\boldsymbol{z}.$$

q is the encoder distribution on latent $\boldsymbol{z}$.

- The first term is the KL between our q and the real prior p on latent $\boldsymbol{z}$. We "guess" a Gaussian prior.

Variational E-M perspective on VAEs

Write

$$\int q(\boldsymbol{z}) \ln \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z})} \, \mathrm{d}\boldsymbol{z} = \int q(\boldsymbol{z}) \ln \frac{p(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z})}{q(\boldsymbol{z})} \, \mathrm{d}\boldsymbol{z}$$
$$= \int q(\boldsymbol{z}) \ln \frac{p(\boldsymbol{z})}{q(\boldsymbol{z})} \, \mathrm{d}\boldsymbol{z} + \int q(\boldsymbol{z}) \ln p(\boldsymbol{x}|\boldsymbol{z}) \, \mathrm{d}\boldsymbol{z}.$$

q is the encoder distribution on latent $\boldsymbol{z}$.

Variational E-M perspective on VAEs

Write

$$\int q(\boldsymbol{z}) \ln \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z})} \, \mathrm{d}\boldsymbol{z} = \int q(\boldsymbol{z}) \ln \frac{p(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z})}{q(\boldsymbol{z})} \, \mathrm{d}\boldsymbol{z}$$
$$= \int q(\boldsymbol{z}) \ln \frac{p(\boldsymbol{z})}{q(\boldsymbol{z})} \, \mathrm{d}\boldsymbol{z} + \int q(\boldsymbol{z}) \ln p(\boldsymbol{x}|\boldsymbol{z}) \, \mathrm{d}\boldsymbol{z}.$$

q is the encoder distribution on latent $\boldsymbol{z}$.

- The second term is the reconstruction error.
  - First note that it's an expectation of $\ln p(\boldsymbol{x}|\boldsymbol{z})$, where $\boldsymbol{z} \sim q$. We can approximate this by averaging $\ln p(\boldsymbol{x}|\boldsymbol{z})$ over draws from q.
  - Then, we pick a probability model for p, and fit parameters of it with our encoder. E.g., squared error $\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2$ corresponds to p being Gaussian and our encoder outputting a mean $\hat{x}$. We will return to this in the homework.
- Note we are sampling *and* modeling $p(\boldsymbol{x}|\boldsymbol{z})$, but we are cutting corners.

Summary (of part 1)

Summary

- The sampling scheme: draw $\boldsymbol{x} \sim \mu$ efficiently, then compute $g(\boldsymbol{x})$, where g is a deep network.
- The basic VAE scheme and its objective function.
  (The ERM perspective.)