10707
Deep Learning: Spring 2020

Andrej Risteski

Machine Learning Department

Lecture 15:
Variational autoencoders, evaluating representations

Recap: the simplest of representation learners

Sparse coding: learn features, s.t. each input can be written as a *sparse linear combination* of some of these features.

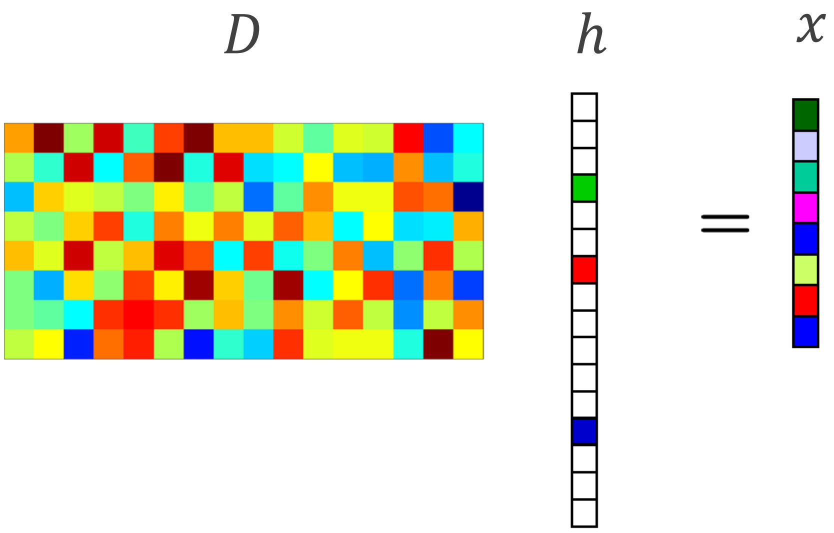
Originally made famous by *Olshausen and Field*, '96 as a model for how early visual processing works (edge detection etc.)

Autoencoders: learn encoding with some constraints (e.g. functional form, sparsity, denoising ability) from which the inputs can be approximately reconstructed.

Sparse coding

Goal: learn a *dictionary* D of features, s.t. each sample x is (approximately) writeable as a *sparse* (i.e. mostly zeros) linear combination of these features.

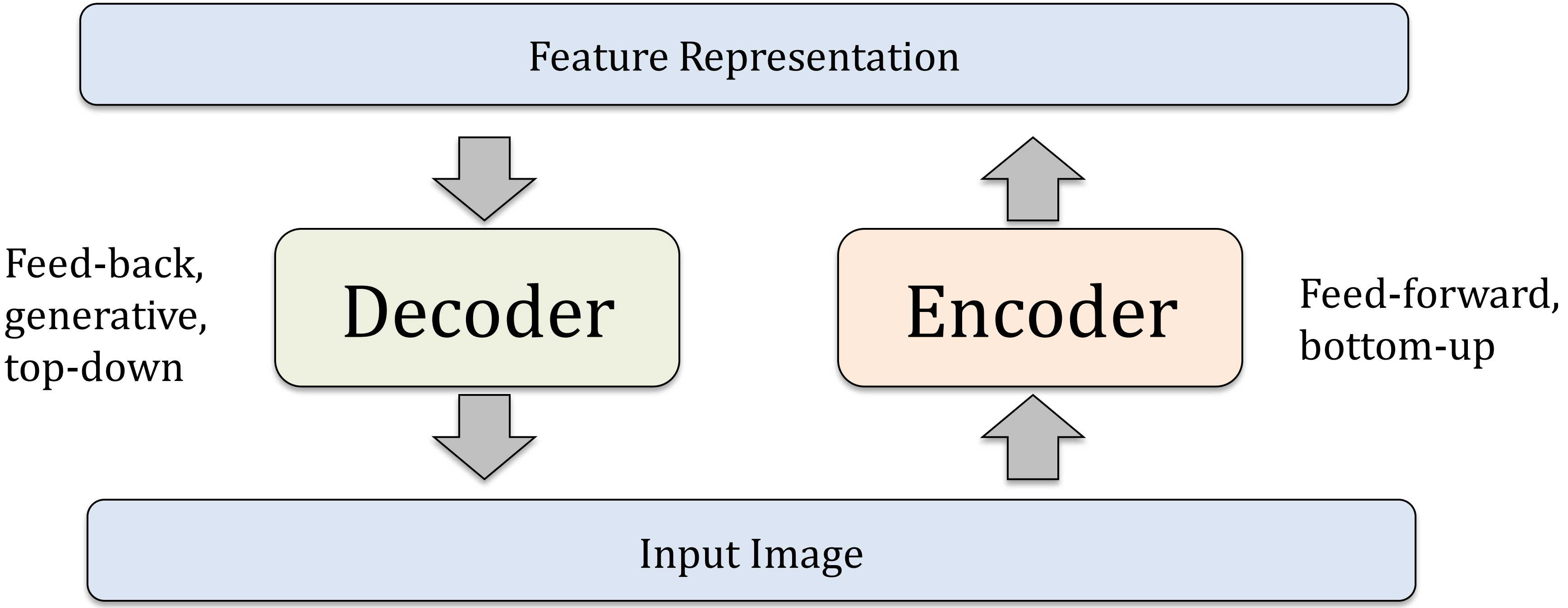
$$\forall x: \quad x \approx Dh, \qquad \|h\|_0 \text{ small}$$



h is the representation of sample x

Autoencoders

The idea behind autoencoders: learn features, s.t. input is reconstructable from them

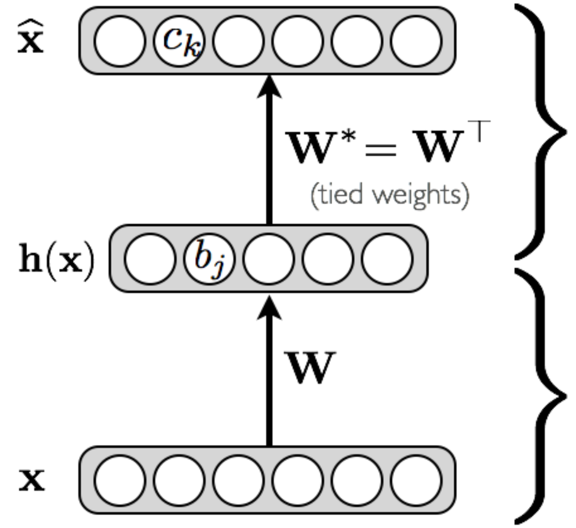


- Details of what goes insider the encoder and decoder matter!
- Need *constraints* to **avoid learning an identity**.

Autoencoders

Some way to prevent identity:

- *Weight tying* of encoder/decoder. (Often magical!)
- *Smaller dimension* for latent variables
- Enforce *sparsity* of the latent representation
- Encourage decoder to be robust to adding noise to x. (*Denoising autoencoder*)
- **Encode to distribution rather than pointmass. (*Variational autoencoder*)**



Typical losses

Loss function for inputs between 0 and 1

$$l(f(\mathbf{x})) = -\sum_k (x_k \log(\widehat{x}_k) + (1 - x_k) \log(1 - \widehat{x}_k))$$

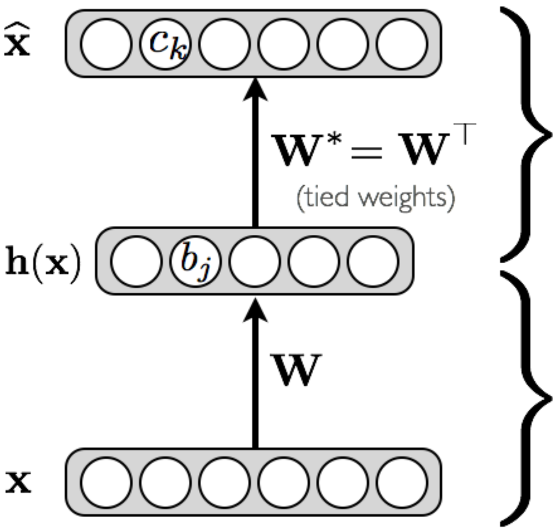
- *Cross-entropy error* ($f(\mathbf{x}) \equiv \widehat{\mathbf{x}}$)

Loss function for real-valued inputs

$$l(f(\mathbf{x})) = \frac{1}{2} \sum_k (\widehat{x}_k - x_k)^2$$

- l_2 error
- we use a linear activation function at the output

Intuitions for weight tieing



Original intuition: similar as doing 2 steps in a Gibbs sampler in RBM's.
(Though not randomized.)

Better intuition: one step of ISTA algorithm for dictionary learning!

Variants, variants, variants

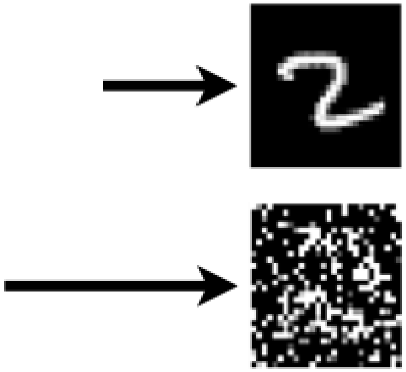
Undercomplete Representation

Hidden layer is *undercomplete* if smaller than the input layer (bottleneck layer, e.g. dimensionality reduction):

- hidden layer "*compresses*" the input
- will compress well only for the training distribution (*maybe not even*)

Hidden units will be

- good features for the training distribution (*potentially*…)
- will not be robust to other types of input (*not trained to compress these*)



Slide Credit: Hugo Larochelle

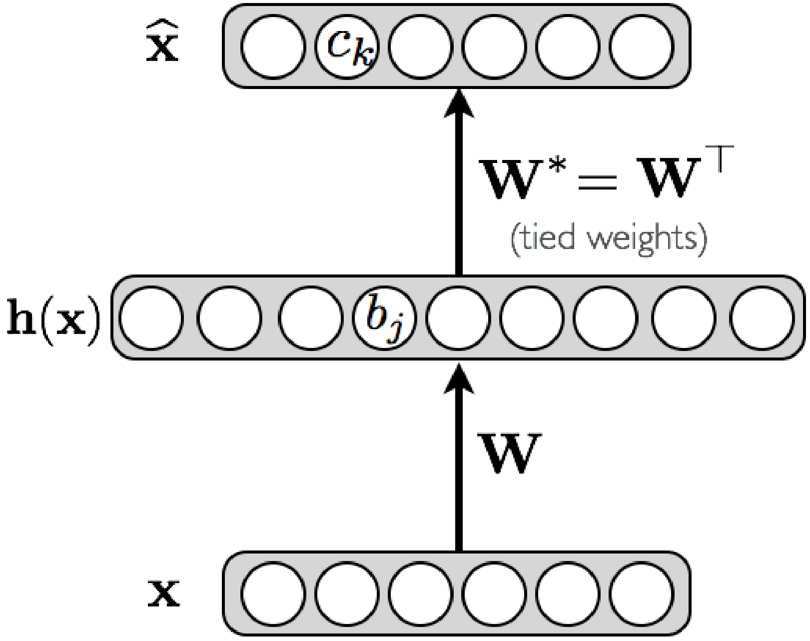
Overcomplete Representation

Hidden layer is *overcomplete* if greater than the input layer

- ➢ no compression in hidden layer
- ➢ each hidden unit could copy a different input component

No guarantee that the hidden units will extract meaningful structure

Other constraints must be made, e.g. sparsity, denoising, etc.

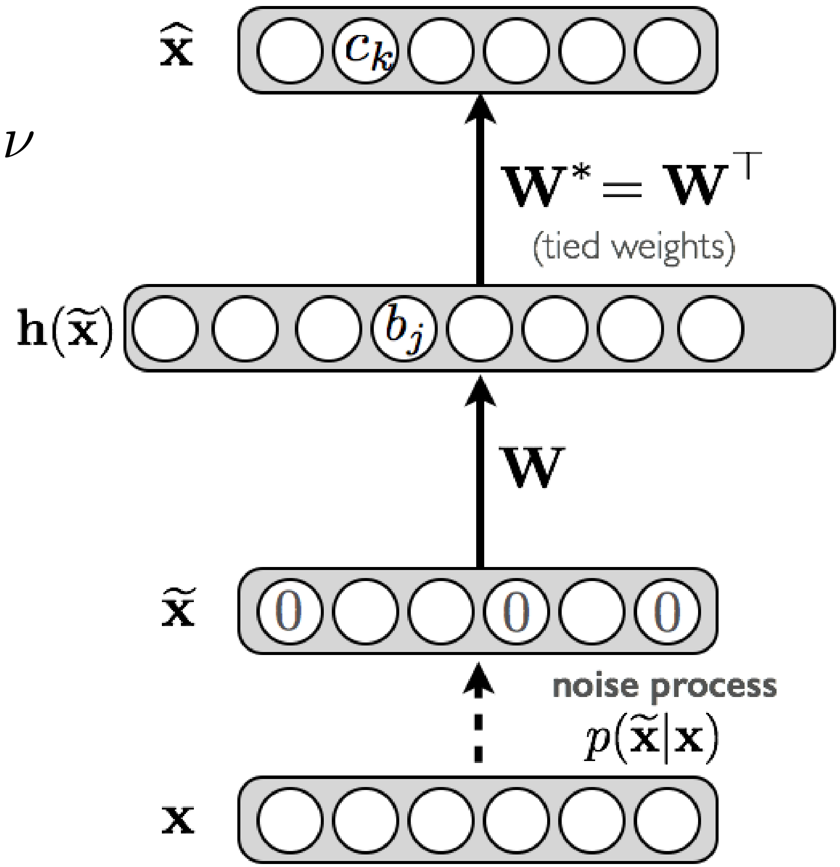


Slide Credit: Hugo Larochelle

Denoising Autoencoder

Idea: representation should be *robust to introduction of noise*:

- *Dropout*: random assignment of subset of inputs to 0, with probability ν
- *Gaussian additive* noise

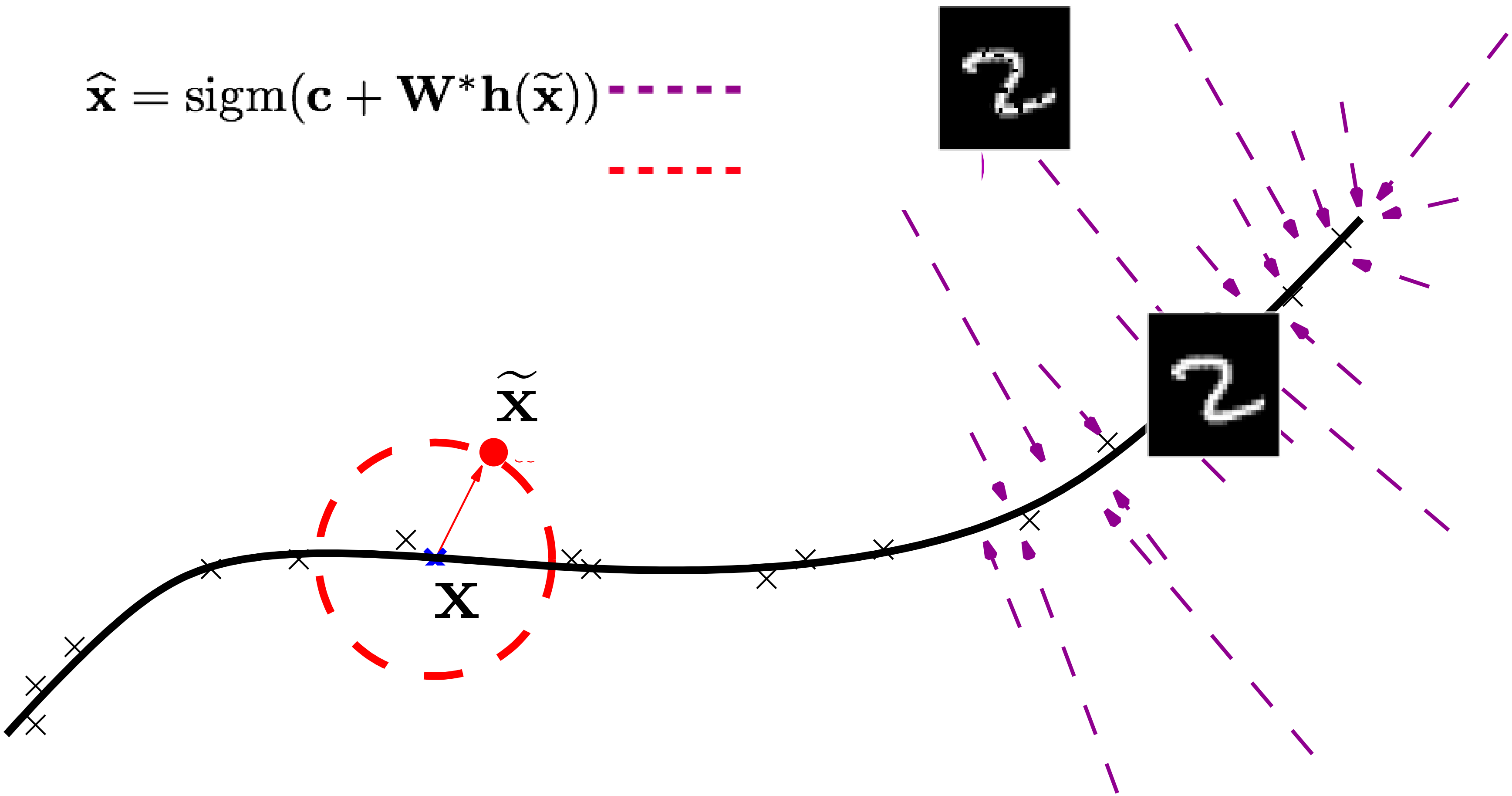


- Reconstruction $\widehat{\mathbf{x}}$ computed from the corrupted input $\widetilde{\mathbf{x}}$

- Loss function compares $\widehat{\mathbf{x}}$ reconstruction with the noiseless input $\mathbf{x}$

Slide Credit: Hugo Larochelle

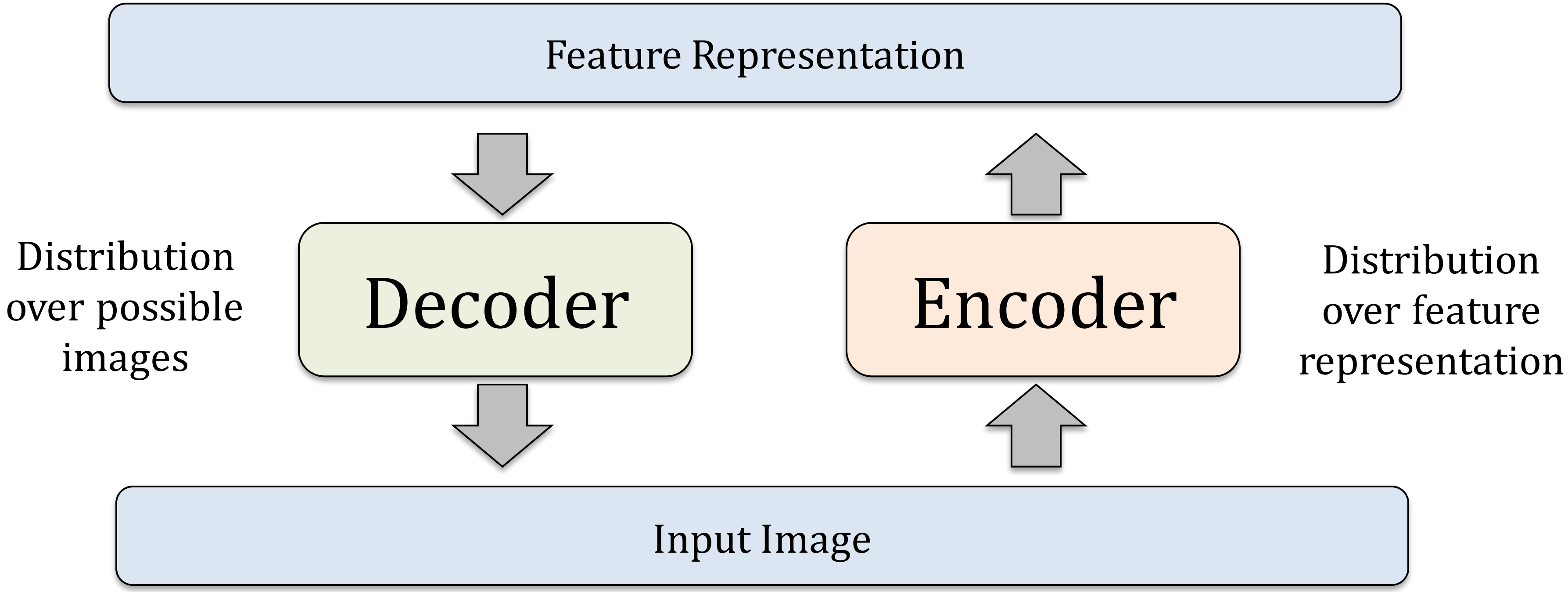
Denoising Autoencoder



Variational autoencoders

The idea: the encoder can output a *distribution*, rather than a *point mass*.

We will derive this via a variational approach.

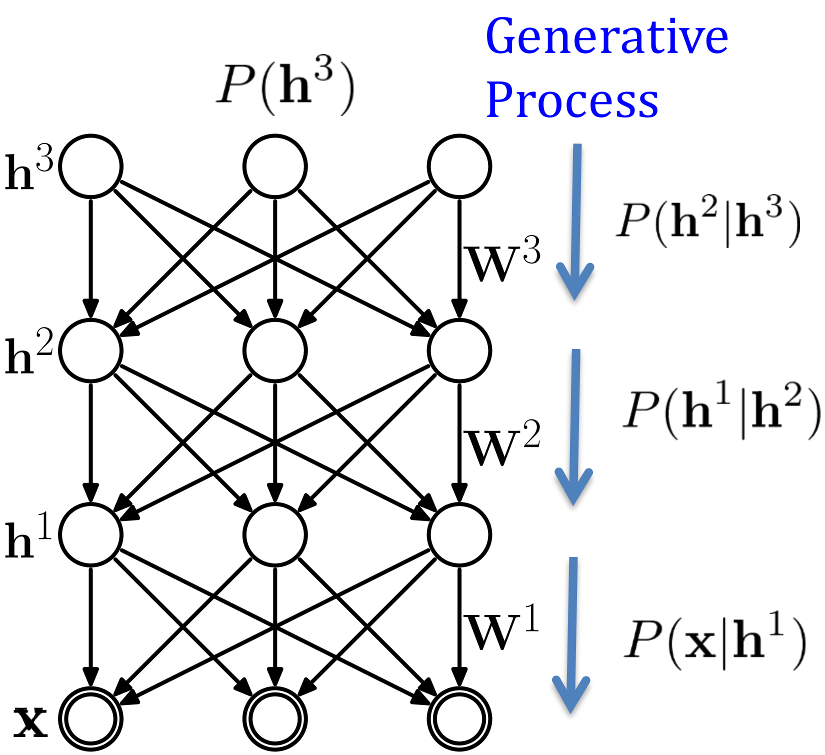


Variational autoencoders

"*Decoder/generator*": directed Bayesian network with Gaussian layers

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{\mathbf{h}^1,\ldots,\mathbf{h}^L} p(\mathbf{h}^L|\boldsymbol{\theta})p(\mathbf{h}^{L-1}|\mathbf{h}^L,\boldsymbol{\theta})\cdots p(\mathbf{x}|\mathbf{h}^1,\boldsymbol{\theta})$$

Each term may denote a complicated nonlinear relationship



Input data

Typically, directed layers are parametrized as:

$$p(\boldsymbol{h}^{L-1}|\boldsymbol{h}^L,\boldsymbol{\theta}) = \mathcal{N}(\mu_{\boldsymbol{\theta}}(\boldsymbol{h}^L),\Sigma_\theta(\boldsymbol{h}^L))$$

Gaussians, means/covariances functions (e.g. one-layer neural net) of previous layer and model parameters θ.

Easy to sample!

Where does an "encoder" come in?

"*Decoder/generator*": directed Bayesian network with Gaussian layers

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{\mathbf{h}^1,\ldots,\mathbf{h}^L} p(\mathbf{h}^L|\boldsymbol{\theta})p(\mathbf{h}^{L-1}|\mathbf{h}^L,\boldsymbol{\theta})\cdots p(\mathbf{x}|\mathbf{h}^1,\boldsymbol{\theta})$$

Recall *learning via variational inference*:

ELBO: $$\log\ p(x) = \max_{q(h^L|x)} H\big(q(h^L|x)\big) + \mathbb{E}_{q(h^L|x)}[\log p(x,h^L)]$$

Max-likelihood can be written as:

$$\max_{\theta\in\Theta} \max_{\{q(h^L|x)\}} \sum_{i=1}^{n} H(q(h^L|x)) + \mathbb{E}_{q(h^L|x)}[\log p(x,h^L)]$$

"*Encoder:*" a directed Bayesian network approximating $q(h^L|x)$

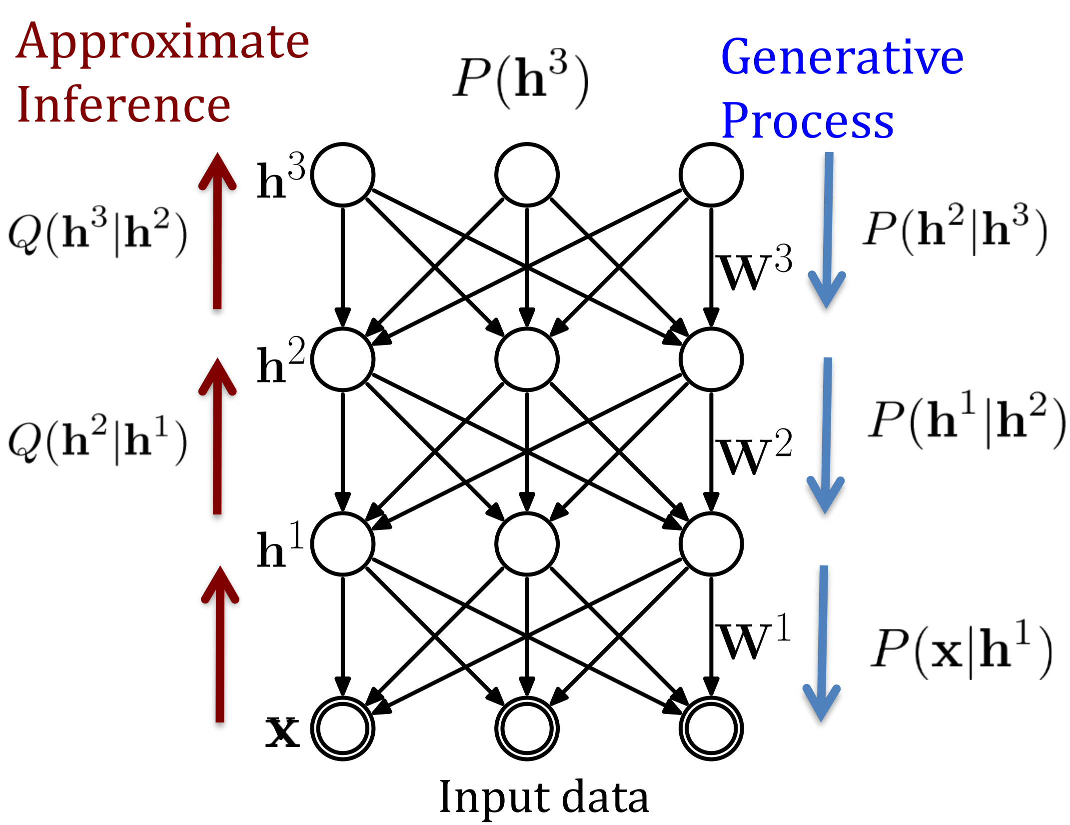
It will be a directed Bayesian network in the "reverse" direction.

Encoder: A "recognition network"

The encoder is defined in terms of an analogous factorization:

$$q(\boldsymbol{h}|\boldsymbol{x},\boldsymbol{\theta}) = q(\boldsymbol{h}^1|\boldsymbol{x},\boldsymbol{\theta})q(\boldsymbol{h}^2|\boldsymbol{h}^1,\boldsymbol{\theta}) \dots q(\boldsymbol{h}^L|\boldsymbol{h}^{L-1},\boldsymbol{\theta})$$

Each term may denote a complicated nonlinear relationship



Typically, directed layers are parametrized as:

$$q\left(\boldsymbol{h}^l|\boldsymbol{h}^{l-1},\boldsymbol{\theta}\right) = \mathcal{N}\left(\mu_{\boldsymbol{\theta}}\left(\boldsymbol{h}^{l-1}\right),\Sigma_{\boldsymbol{\theta}}\left(\boldsymbol{h}^{l-1}\right)\right)$$

Means/covariances fns (e.g. one-layer neural net) of previous layer and parameters $\boldsymbol{\theta}$.

Why is this called an "encoder"?

Max-likelihood can be written as:

$$\max_{\theta \in \Theta} \max_{\{q(h^L|x)\}} \sum_{i=1}^{n} H(q(h^L|x)) + \mathbb{E}_{q(h^L|x)}[\log p(x, h^L)]$$

Let's rewrite the ELBO a bit:

$$H\big(q(h^L|x)\big) + \mathbb{E}_{q(h^L|x)}[\log p(x, h^L)] = \mathbb{E}_{q(h^L|x)}[\log p(x, h^L) - \log q(h^L|x)]$$

$$= \mathbb{E}_{q(h^L|x)}[\log p(h^L) + \log p(x|h^L) - \log q(h^L|x)]$$

$$= \mathbb{E}_{q(h^L|x)} \log p(x|h^L) - \mathbb{E}_{q(h^L|x)} \log \frac{q\big(h^L|x\big)}{p(h^L)}$$

$$= \underbrace{\mathbb{E}_{q(h^L|x)} \log p(x|h^L)}_{\text{"Reconstruction" error}} - \underbrace{KL\big(q(h^L|x)\big\|p(h^L)\big)}_{\text{"Regularization towards prior"}}$$

"Reconstruction" error
Use q as a "probabilistic" encoder,
Use p as a "probabilistic" decoder,

$x \rightarrow h^L \rightarrow x$

"Regularization
towards prior"

How to train?

Max-likelihood can be written as:

$$\max_{\theta \in \Theta} \max_{\{q_\theta(h^L|x)\}} \sum_x \mathbb{E}_{q_\theta(h^L|x)} \log \frac{p_\theta(x, h^L)}{q_\theta(h^L|x)}$$

As usual: we need to be able to take gradients in θ

The problem: the expectation is with respect to $q_\theta(h^L|x)$, which depends on the variables we are taking a derivative with respect to.

Observation*:* a derivative of the type $\nabla_\theta \mathbb{E}_p f(\theta)$ is easy to approximate if p does not depend on θ:

$$\nabla_\theta \mathbb{E}_p f(\theta) = \mathbb{E}_p \nabla_\theta f(\theta)$$

θ_i: iid samples from p

Exchange only works if p doesn't dep on θ

$$\approx \frac{1}{N} \sum_i \nabla_{\theta_i} f(\theta_i)$$

How to train?

Max-likelihood can be written as:

$$\max_{\theta\in\Theta} \max_{\{q_\theta(h^L|x)\}} \sum_x \mathbb{E}_{q_\theta(h^L|x)} \log \frac{p_\theta(x,h^L)}{q_\theta(h^L|x)}$$

As usual: we need to be able to take gradients in θ

***Remark**:* there is a common Monte Carlo estimator of $\nabla_\theta \mathbb{E}_{p(\theta)} f(\theta)$ as well but it typically has high variance:

$$\nabla_\theta \mathbb{E}_{p(\theta)} f(\theta) = \int \nabla_\theta f(\theta)\, p(\theta) d\theta + \int f(\theta)\, \nabla_\theta p(\theta) d\theta$$

$$= \int \nabla_\theta f(\theta)\, p(\theta) d\theta + \int f(\theta) \frac{p(\theta)}{p(\theta)}\, \nabla_\theta p(\theta) d\theta$$

$$= \int \nabla_\theta f(\theta)\, p(\theta) d\theta + \int f(\theta) \nabla_\theta \log p(\theta)\, p(\theta) d\theta$$

$$= \mathbb{E}_{p(\theta)}[\nabla_\theta f(\theta) + \ f(\theta) \nabla_\theta \log p(\theta)]$$

This term typically is high var.,

How to train?

Max-likelihood can be written as:

$$\max_{\theta \in \Theta} \max_{\{q_\theta(h^L|x)\}} \sum_x \mathbb{E}_{q_\theta(h^L|x)} \log \frac{p_\theta(x, h^L)}{q_\theta(h^L|x)}$$

As usual: we need to be able to take gradients in θ

The solution: write the expectation $\mathbb{E}_{q_\theta(h^L|x)} \log \frac{p_\theta(x,h^L)}{q_\theta(h^L|x)}$ as an expectation over a distribution not dependent on θ.

Kingma-Welling '13: reparametrization trick!

Main idea: a sample from $\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ can be generated as follows

Sample $\mathbf{x} \sim \mathcal{N}(0, \mathrm{I})$.

Output $\boldsymbol{y} = \boldsymbol{\mu} + \boldsymbol{\Sigma}^{1/2}\boldsymbol{x}$.

How to train?

Max-likelihood can be written as:

$$\max_{\theta \in \Theta} \max_{\{q_\theta(h^L|x)\}} \sum_x \mathbb{E}_{q_\theta(h^L|x)} \log \frac{p_\theta(x, h^L)}{q_\theta(h^L|x)}$$

As usual: we need to be able to take gradients in θ

Recall that $q(\boldsymbol{h}|\boldsymbol{x}, \boldsymbol{\theta}) = q(\boldsymbol{h}^1|\boldsymbol{x}, \boldsymbol{\theta})q(\boldsymbol{h}^2|\boldsymbol{h}^1, \boldsymbol{\theta}) \dots q(\boldsymbol{h}^L|\boldsymbol{h}^{L-1}, \boldsymbol{\theta})$

where $q(\boldsymbol{h}^l|\boldsymbol{h}^{l-1}, \boldsymbol{\theta}) = \mathcal{N}(\mu_{\boldsymbol{\theta}}(\boldsymbol{h}^{l-1}), \Sigma_{\boldsymbol{\theta}}(\boldsymbol{h}^{l-1}))$

To produce a sample from $q(\boldsymbol{h}|\boldsymbol{x}, \boldsymbol{\theta})$, sample iid standard Gaussians $\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2, \dots, \boldsymbol{\epsilon}_L$. Set

$$\mathbf{h}^\ell\left(\boldsymbol{\epsilon}^\ell, \mathbf{h}^{\ell-1}, \boldsymbol{\theta}\right) = \boldsymbol{\Sigma}(\mathbf{h}^{\ell-1}, \boldsymbol{\theta})^{1/2}\boldsymbol{\epsilon}^\ell + \boldsymbol{\mu}(\mathbf{h}^{\ell-1}, \boldsymbol{\theta})$$

Using the reparametrization trick

Max-likelihood can be written as:

$$\max_{\theta\in\Theta} \max_{\{q_\theta(h^L|x)\}} \sum_x \mathbb{E}_{q_\theta(h^L|x)} \log \frac{p_\theta(x,h^L)}{q_\theta(h^L|x)}$$

We can hence write the gradient wrt to θ :

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{h}\sim q(\mathbf{h}|\mathbf{x},\boldsymbol{\theta})} \left[\log \frac{p(\mathbf{x},\mathbf{h}|\boldsymbol{\theta})}{q(\mathbf{h}|\mathbf{x},\boldsymbol{\theta})} \right]$$

$$= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\epsilon}^1,\ldots,\boldsymbol{\epsilon}^L\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})} \left[\log \frac{p(\mathbf{x},\mathbf{h}(\boldsymbol{\epsilon},\mathbf{x},\boldsymbol{\theta})|\boldsymbol{\theta})}{q(\mathbf{h}(\boldsymbol{\epsilon},\mathbf{x},\boldsymbol{\theta})|\mathbf{x},\boldsymbol{\theta})} \right]$$

$$= \mathbb{E}_{\boldsymbol{\epsilon}^1,\ldots,\boldsymbol{\epsilon}^L\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})} \left[\nabla_{\boldsymbol{\theta}} \log \frac{p(\mathbf{x},\mathbf{h}(\boldsymbol{\epsilon},\mathbf{x},\boldsymbol{\theta})|\boldsymbol{\theta})}{q(\mathbf{h}(\boldsymbol{\epsilon},\mathbf{x},\boldsymbol{\theta})|\mathbf{x},\boldsymbol{\theta})} \right]$$

Using the reparametrization trick

Max-likelihood can be written as:

$$\max_{\theta\in\Theta} \max_{\{q_\theta(h^L|x)\}} \sum_x \mathbb{E}_{q_\theta(h^L|x)} \log \frac{p_\theta(x, h^L)}{q_\theta(h^L|x)}$$

We can hence write the gradient wrt to θ :

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{h}\sim q(\mathbf{h}|\mathbf{x},\boldsymbol{\theta})} \left[\log \frac{p(\mathbf{x},\mathbf{h}|\boldsymbol{\theta})}{q(\mathbf{h}|\mathbf{x},\boldsymbol{\theta})}\right] = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\epsilon}^1,\ldots,\boldsymbol{\epsilon}^L\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})} \left[\log \frac{p(\mathbf{x},\mathbf{h}(\boldsymbol{\epsilon},\mathbf{x},\boldsymbol{\theta})|\boldsymbol{\theta})}{q(\mathbf{h}(\boldsymbol{\epsilon},\mathbf{x},\boldsymbol{\theta})|\mathbf{x},\boldsymbol{\theta})}\right]$$

$$= \mathbb{E}_{\boldsymbol{\epsilon}^1,\ldots,\boldsymbol{\epsilon}^L\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})} \left[\nabla_{\boldsymbol{\theta}} \log \frac{p(\mathbf{x},\mathbf{h}(\boldsymbol{\epsilon},\mathbf{x},\boldsymbol{\theta})|\boldsymbol{\theta})}{q(\mathbf{h}(\boldsymbol{\epsilon},\mathbf{x},\boldsymbol{\theta})|\mathbf{x},\boldsymbol{\theta})}\right]$$

We can approximate the expectation by an empirical average as before.

For **fixed** $\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2, \ldots, \boldsymbol{\epsilon}_L$: log p and log q are easy to take gradients of via backpropagating.

It's common to have **diagonal covariance mxs** for training efficiency.

Part II: Evaluating representations

Desiderata for representations

What do we want out a representation?

Many possible answers here. First, a few uncontroversial desiderata:

Interpretability: if the derived features are semantically meaningful, and interpretable by a human, they can be easily evaluated.
(e.g. noisy-OR: "features" are diseases a patient has)

Sparsity of a representation is an important subcase: "explanatory" features for sample can be examined if there are a small number of them.

Downstream usability: the features are "useful" for downstream tasks. Some examples:

Improving label efficiency: if, for a task, a linear (or otherwise "simple") classifier can be trained on features and it works well, smaller # of labeled samples are needed.

Desiderata for representations

Obvious issue: interpretability and "usefulness" are not easily mathematically expressed. We need some "proxies" that induce such properties.

This is a lot more contraversial – here we survey some general desiderata, proposed as early as *Bengio-Courville-Vincent '14:*

Hierarchy/compositionality: video/images/text/ are expected to have hierarchical structure – depth helps induce such structure.

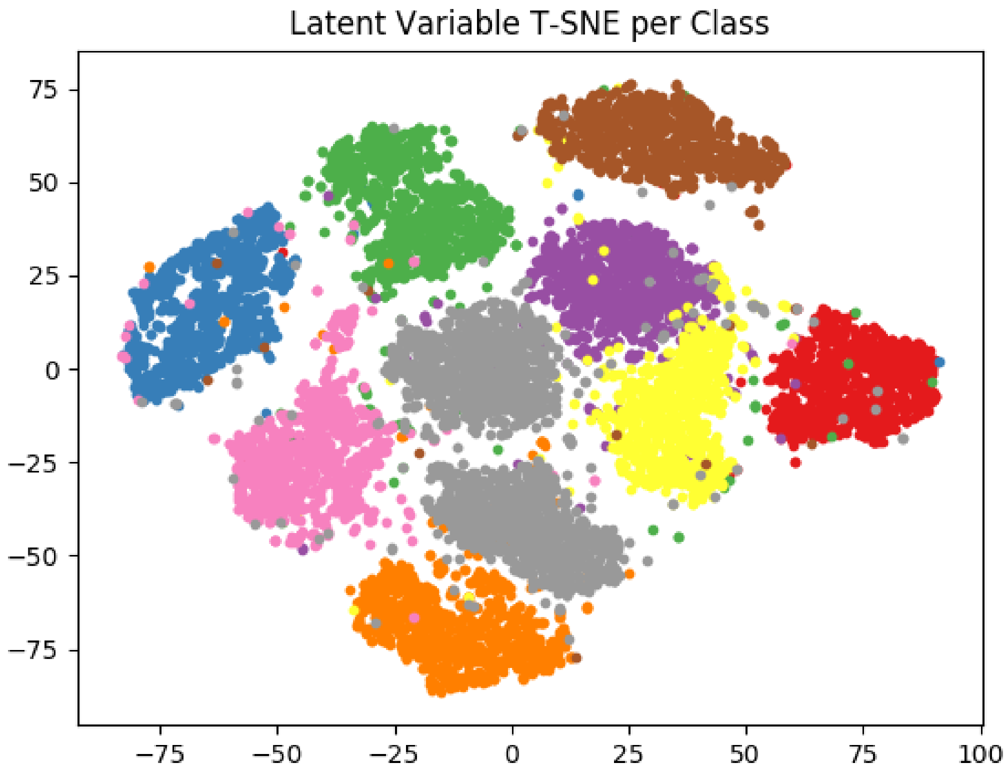
Semantic clusterability: features of the same "semantic class" (e.g. images in the same category) are clustered.

Linear interpolation: in representation space, linear interpolations produce meaningful data points (i.e. "latent space is convex"). Sometimes called *manifold flattening*.

Disentangling: features capture "independent factors of variation" of data. (*Bengio-Courville-Vincent '14).* Has been very popular in modern unsupervised learning, though many potential issues with it.

Semantic clustering

Semantic clusterability: features of the same “semantic class” (e.g. images in the same category) are clustered together.



The intuition:

If semantic classes are linearly (or other simple function) separable, and labels on downstream tasks depend linearly on semantic classes – can afford to learn a simple classifier !!

t-SNE projection of VAE-learned features of the 10 MNIST classes.
Image from https://pyro.ai/examples/vae.html

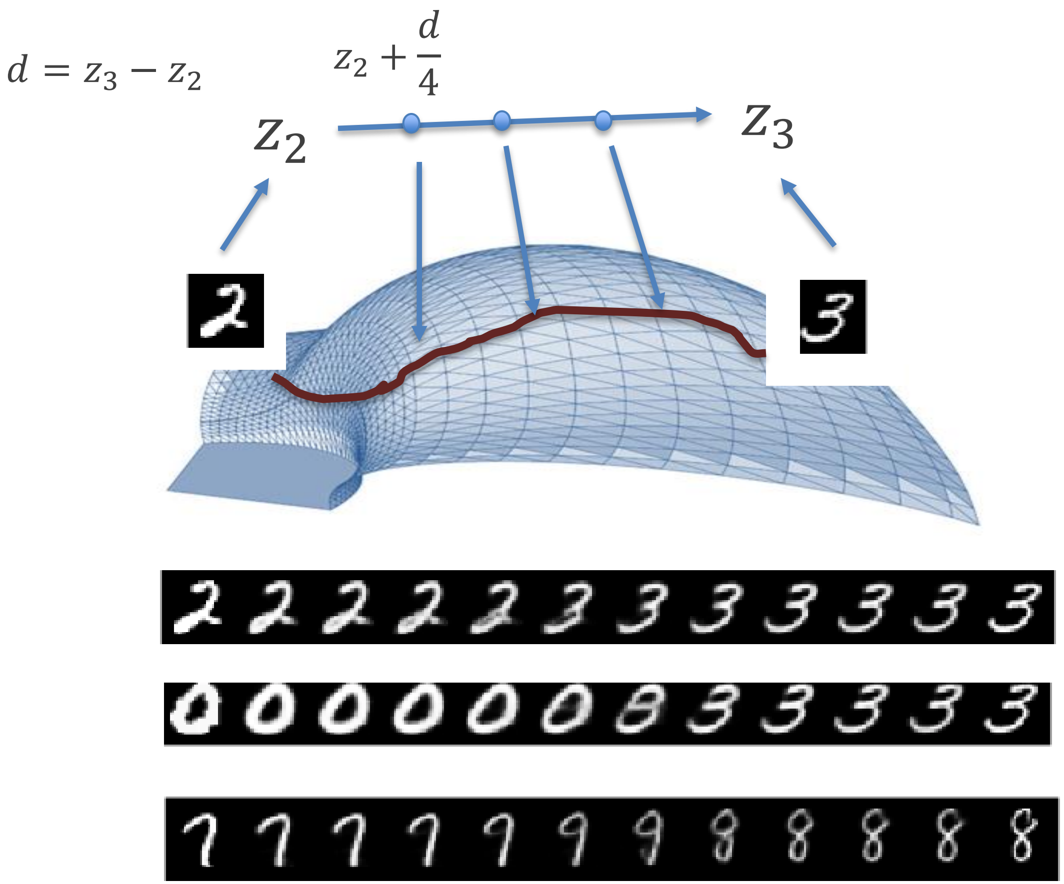
Semantic clustering

Semantic clusterability: features of the same "semantic class" (e.g. images in the same category) are clustered together.

t-SNE projection of word embeddings for artists (clustered by genre). Image from https://medium.com/free-code-camp/learn-tensorflow-the-word2vec-model-and-the-tsne-algorithm-using-rock-bands-97c99b5dcb3a

Linear interpolation

Linear interpolation: in representation space, linear interpolations produce meaningful data points. (i.e. “latent space is convex”)



Interpolations for a VAE trained on MNIST.

The intuition:

The data manifold is complicated/curved.

The latent variable manifold is a convex set – moving in straight lines keeps us on it.

Linear interpolation

Linear interpolation: in representation space, linear interpolations produce meaningful data points. (i.e. "latent space is convex")

Interpolations for a BigGAN, image from
https://thegradient.pub/bigganex-a-dive-into-the-latent-space-of-biggan/

Disentangled representations

Disentangling: features capture "independent factors of variation" of data. (*Bengio-Courville-Vincent '14).* Has been very popular in modern unsupervised learning, though many potential issues with it.

For concreteness, let's assume that we have a latent variable model for data with latent variables $\boldsymbol{z}$, observables $\boldsymbol{x}$, and joint distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})$

There are (at least) two ways to formalize this (literature is not always clear on which one is aimed for!):

Prior disentangling: $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ is a product distribution, i.e. $p_{\boldsymbol{\theta}}(\boldsymbol{z}) = \Pi_i p_{\boldsymbol{\theta}}(\boldsymbol{z}_{\boldsymbol{i}})$

Classical example: ICA (independent component analysis)

Posterior disentangling: fit a variational posterior $q_{\boldsymbol{\theta}}$ s.t. $q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ is (on average over **x**) a product distribution

In other words, $\int_x q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\, p(\boldsymbol{x}) d\boldsymbol{x}$ – usually called the *aggregate posterior* – is close to a product distribution.

Disentangled representations

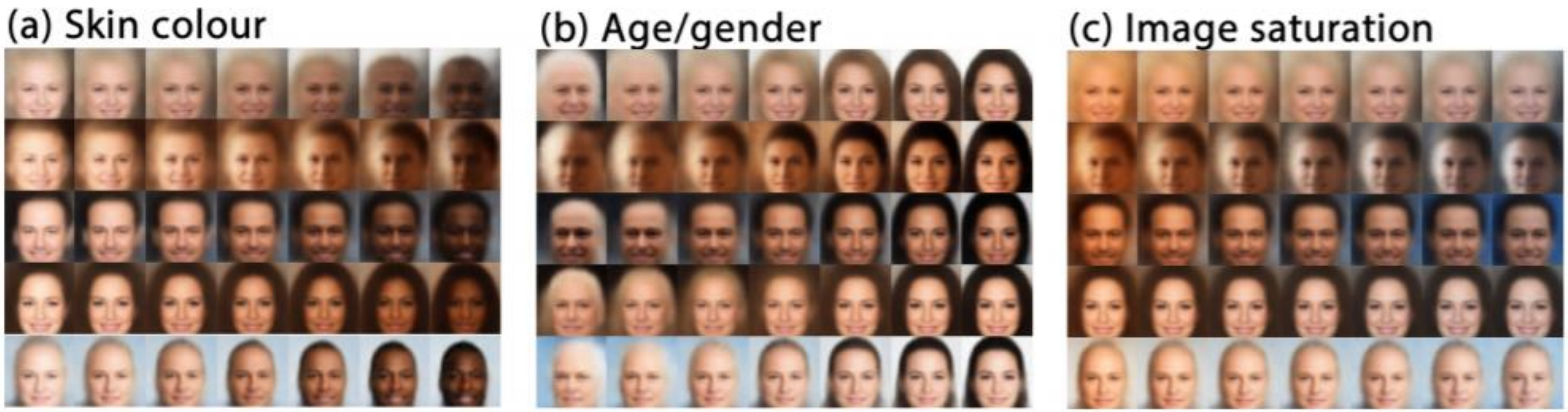


Figure 4: **Latent factors learnt by β-VAE on celebA:** traversal of individual latents demonstrates that β-VAE discovered in an unsupervised manner factors that encode skin colour, transition from an elderly male to younger female, and image saturation.

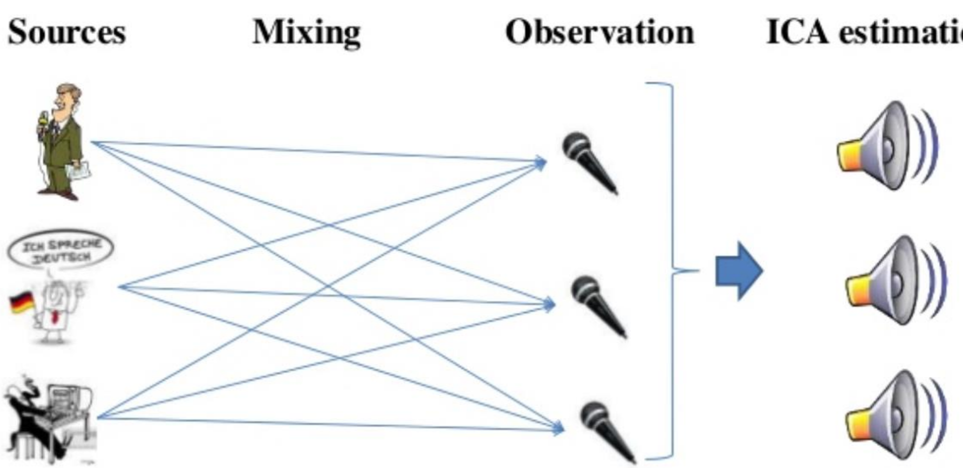
Posterior disentangling in β −VAE. To produce plots, infer latent variable for an image, then change a single latent variable gradually.
Image from Higgins et al. '17.

Prior disentangling

Prior disentangling: $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ is a product distribution, i.e. $p_{\boldsymbol{\theta}}(\boldsymbol{z}) = \Pi_i p_{\boldsymbol{\theta}}(\boldsymbol{z}_{\boldsymbol{i}})$

Classical example: ICA (independent component analysis), also called the "cocktail party problem".

Assume data is generated as $x = Az,\ z \in \mathbb{R}^d, A \in \mathbb{R}^{d \times d}$



If z has an independent, *non-Gaussian* prior, model is identifiable and efficiently learnable.
(See, e.g. *Frieze-Jerum-Kannan '96, Anandkumar et al '12*)

Other examples: noisy-OR networks (diseases are independent), general Bayesian nets, viewing top variables as z's, GANs, …

Posterior disentanglement in VAEs

Recall the "regularization" view of the VAEs objective:

$$\underbrace{\sum_x \mathbb{E}_{q(h^L|x)} \log p(x|h^L)}_{\text{"Reconstruction" error}} - \underbrace{KL(q(h^L|x)||p(h^L))}_{\text{"Regularization towards prior"}}$$

Consider a prior which is a product distribution (e.g. standard Gaussian):

The KL term implicitly penalizes distributions for which

$$\sum_x KL(q(h^L|x)||p(h^L)) \approx \mathbb{E}_{x \sim p^*} KL(q(h^L|x)||p(h^L))$$

is large – i.e. the aggregated posterior is far from a product distribution.

Posterior disentanglement in VAEs

Recall the "regularization" view of the VAEs objective:

$$\underbrace{\sum_x \mathbb{E}_{q(h^L|x)} \log p(x|h^L)}_{\text{"Reconstruction" error}} - \underbrace{KL(q(h^L|x)||p(h^L))}_{\text{"Regularization towards prior"}}$$

The KL term implicitly penalizes distributions for which

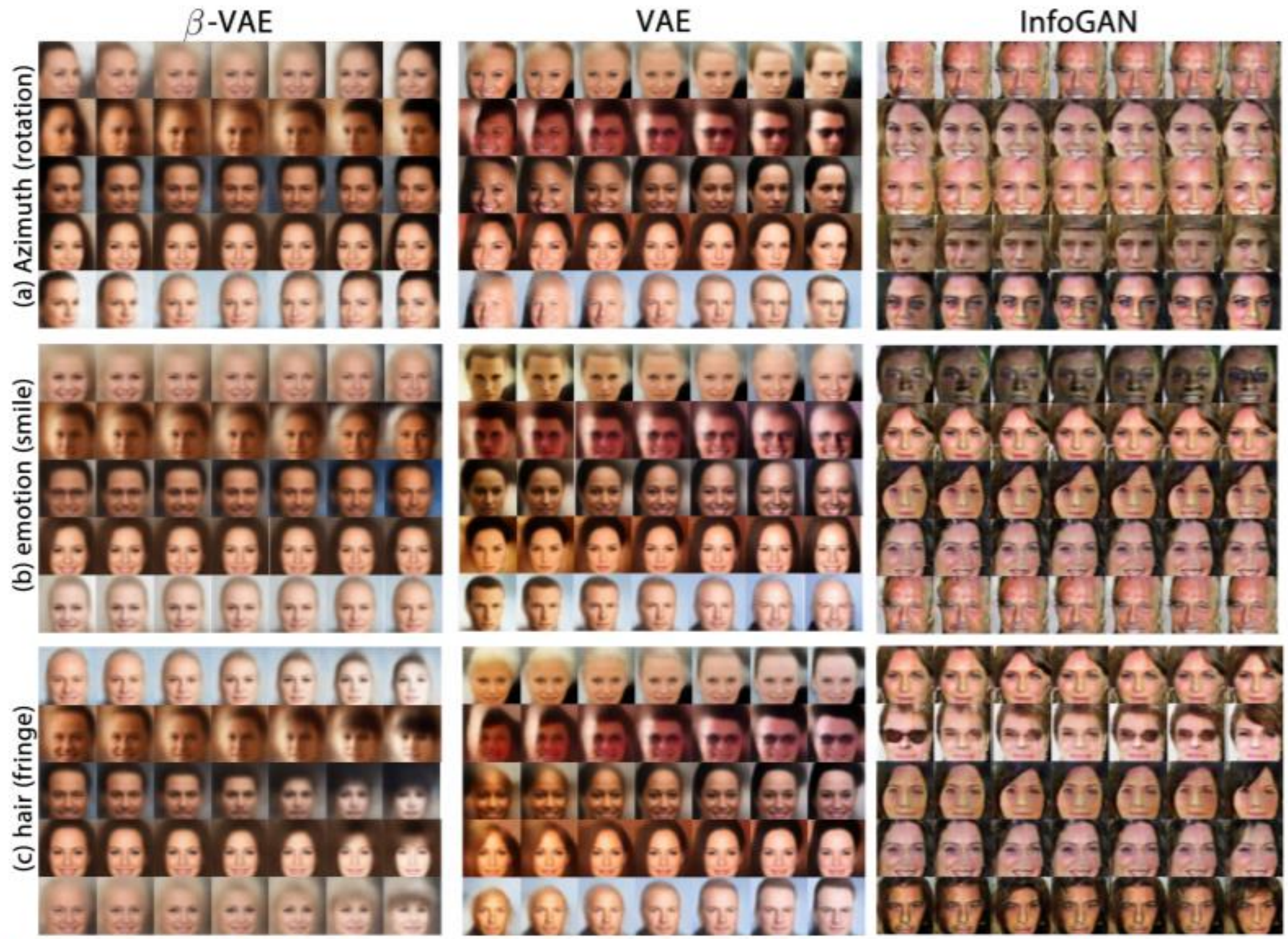
$$\sum_x KL(q(h^L|x)||p(h^L)) \approx \mathbb{E}_{x\sim p^*} KL(q(h^L|x)||p(h^L))$$

The idea of *Higgins et al '17*: introduce a "weighting" factor to put more weight on reconstruction or disentanglement:

β −VAE objective: $\sum_x \mathbb{E}_{q(h^L|x)} \log p(x|h^L) - \beta\, KL(q(h^L|x)||p(h^L))$

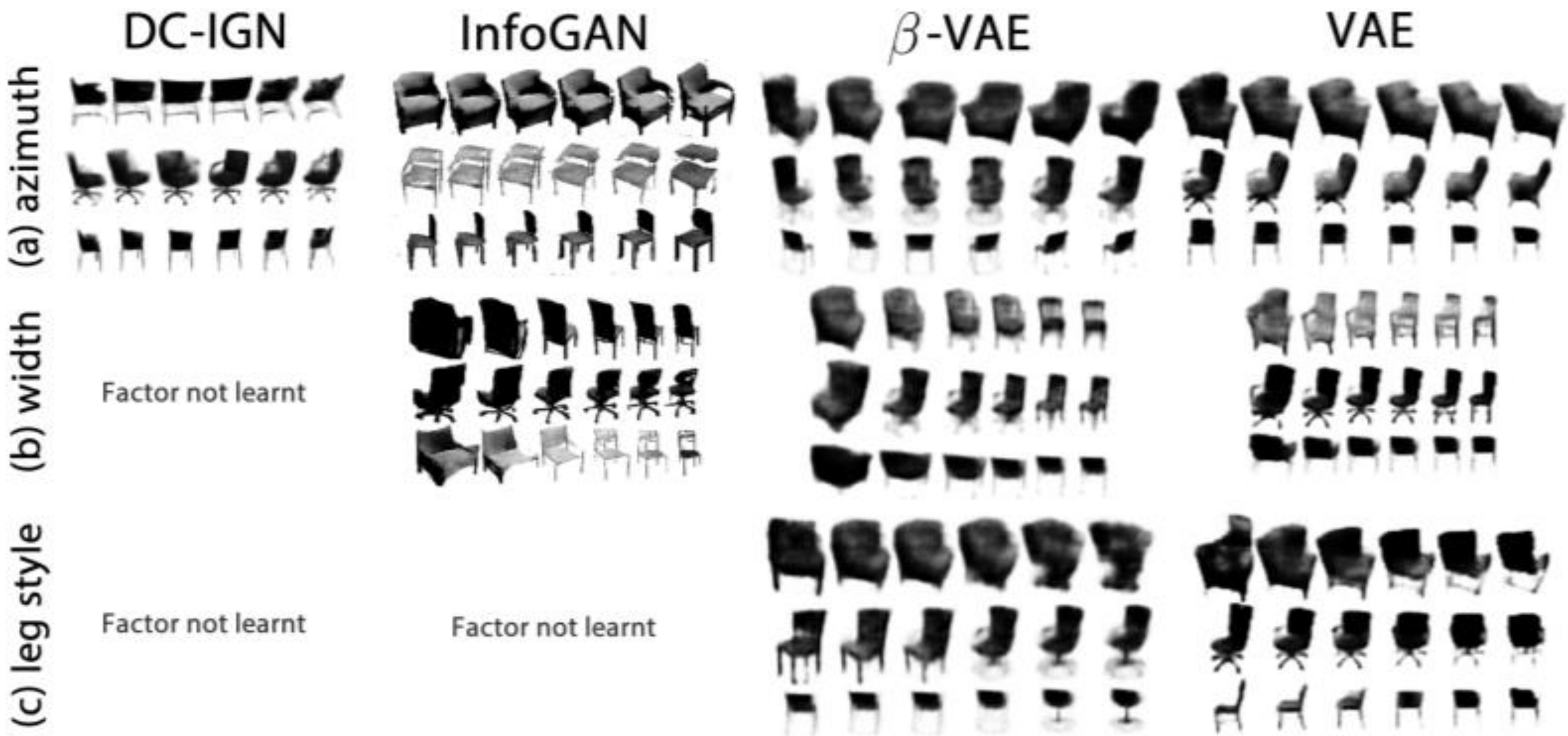
β large: more weight on disentanglement

Posterior disentanglement in VAEs



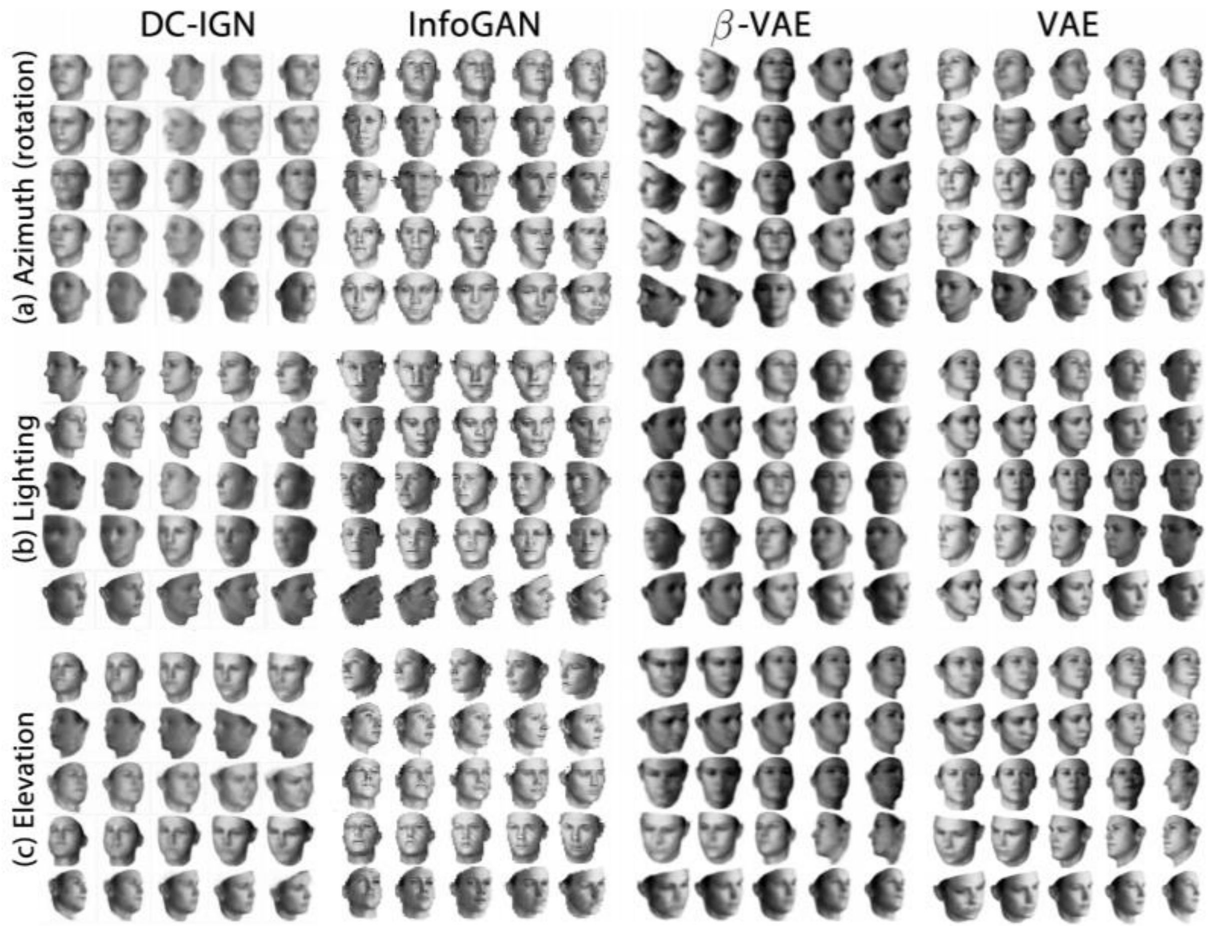
Comparing disentangling of different types of generative models.
Image from Higgins et al. '17.

Posterior disentanglement in VAEs



Comparing disentangling of different types of generative models.
Image from Higgins et al. '17.

Posterior disentanglement in VAEs



Comparing disentangling of different types of generative models.
Image from Higgins et al. '17.

Measuring disentanglement

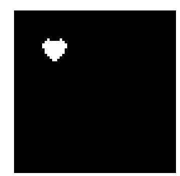
Metrics are typically defined *assuming access* to a dataset with “ground-truth” variation factors. Example: dSprites dataset

dSprites is a dataset of 2D shapes procedurally generated from 6 ground truth independent latent factors. These factors are *color*, *shape*, *scale*, *rotation*, *x* and *y* positions of a sprite.

All possible combinations of these latents are present exactly once, generating N = 737280 total images.

Latent factor values

- Color: white
- Shape: square, ellipse, heart
- Scale: 6 values linearly spaced in [0.5, 1]
- Orientation: 40 values in [0, 2 pi]
- Position X: 32 values in [0, 1]
- Position Y: 32 values in [0, 1]



Measuring disentanglement

Metrics are typically defined *assuming access* to a dataset with K "ground-truth" variation factors.

BetaVAE metric: based on "linear separability" of factors

Generate a **training set** of samples as follows:

Sample a **batch** of B samples as follows:

Pick a **ground-truth variation factor k** uniformly at random from [K].

Generate two sets of "ground truth" latent factors $\boldsymbol{v_1}, \boldsymbol{v_2} \in \mathbb{R}^K$, s.t. $(\boldsymbol{v_1})_\mathrm{k} = (\boldsymbol{v_2})_\mathrm{k}$, and other coords are independently, randomly sampled.

Generate **images** $\boldsymbol{x_1}, \boldsymbol{x_2}$ from $\boldsymbol{v_1}, \boldsymbol{v_2}$.

Infer latent vars $\boldsymbol{z_1}, \boldsymbol{z_2}$ using model we are evaluating. (e.g. encoder in VAE)

Calculate average $\boldsymbol{z_{avg}}$ of $|\boldsymbol{z_1} - \boldsymbol{z_2}|$ in batch, add $(\boldsymbol{z_{avg}}, \mathrm{k})$ to training set.

Train linear predictor on training set, evaluate it's test performance.

Measuring disentanglement

Metrics are typically defined *assuming access* to a dataset with K "ground-truth" variation factors.

BetaVAE metric: based on "linear separability" of factors

Generate a training set of samples as follows:

- Sample a batch of B samples as follows:
  - Pick a ground-truth variation factor k uniformly at random from [K].
  - Generate two sets of "ground truth" latent factors $\boldsymbol{v_1}, \boldsymbol{v_2} \in \mathbb{R}^K$, s.t. $(\boldsymbol{v_1})_{\mathrm{k}} = (\boldsymbol{v_2})_{\mathrm{k}}$, and other coordinates are independently, randomly sampled.
  - Generate images $\boldsymbol{x_1}, \boldsymbol{x_2}$ from $\boldsymbol{v_1}, \boldsymbol{v_2}$.
  - Infer latent variables $\boldsymbol{z_1}, \boldsymbol{z_2}$ using model we are evaluating.(E.g. encoder in VAE)
- Calculate average $\boldsymbol{z_{avg}}$ of $|\boldsymbol{z_1} - \boldsymbol{z_2}|$ and add $(\boldsymbol{z_{avg}}, \mathrm{k})$ to training set.

Train linear predictor on above training set, and evaluate it's test performance.

Intuition: averaging should make coords in $\boldsymbol{z_{avg}}$ different from k smaller, thus linear classifier should "focus" on k.

Many variants of this exist. (e.g. FactorVAE, mutual information gap, etc.)

Measuring disentanglement

Locatello et al '19, "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations" (Best paper award at ICML '19): A large-scale study of disentanglement measures, as well as gen. models.

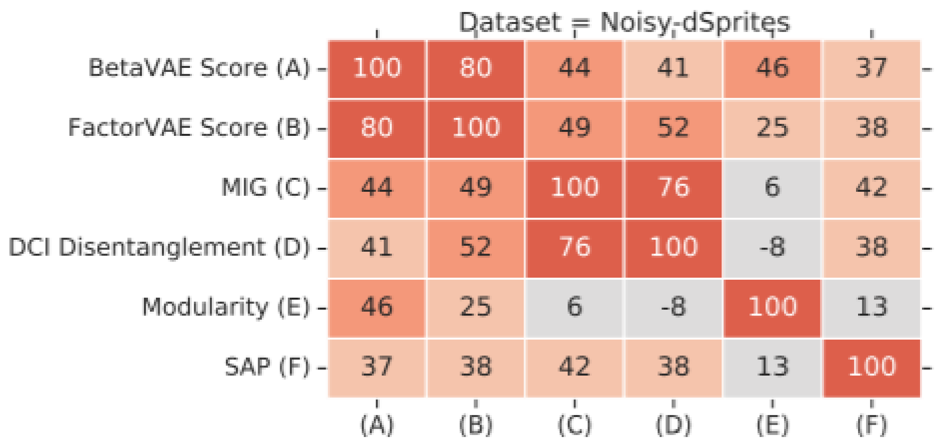


Figure 2. Rank correlation of different metrics on Noisy-dSprites. Overall, we observe that all metrics except Modularity seem mildly correlated with the pairs BetaVAE and FactorVAE, and MIG and DCI Disentanglement strongly correlated with each other.

Usefulness of disentanglement?

Locatello et al '19, "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations" (Best paper award at ICML '19): A large-scale study of disentanglement measures, as well as gen. models.

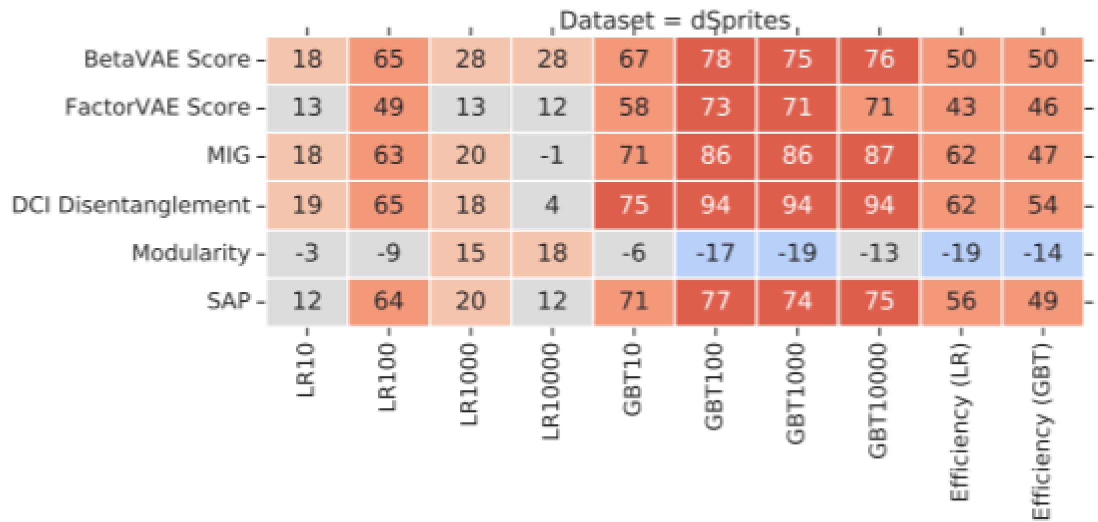


Figure 5. Rank correlations between disentanglement metrics and downstream performance (accuracy and efficiency) on dSprites.

Downstream classification task: predict **true** ground-truth factors (w/ multiclass logistic regression)

Carefull to extrapolate too much – task/setup is a little contrived.

Usefulness of disentanglement?

Locatello et al '19, "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations" (Best paper award at ICML '19): A large-scale study of disentanglement measures, as well as gen. models.

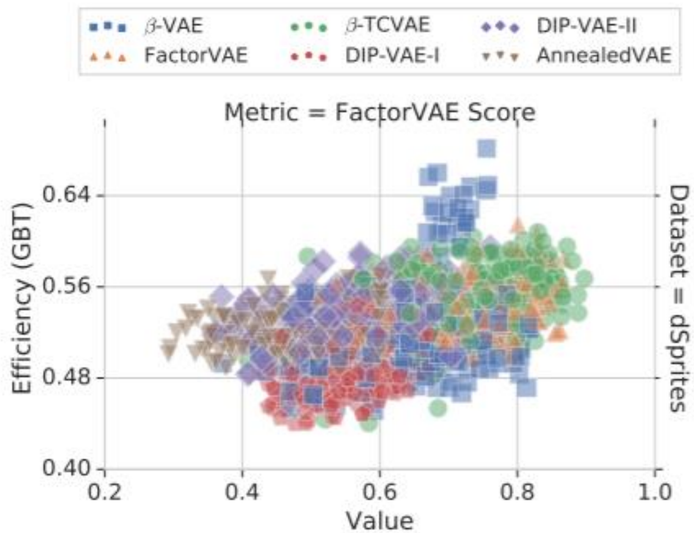


Figure 6. Statistical efficiency of the FactorVAE Score for learning a GBT downstream task on dSprites.

Statistical efficiency measure: average accuracy based on 100 samples divided by the average accuracy based on 10 000 samples

Issue of ill-posedness?

Locatello et al '19, "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations" (Best paper award at ICML '19):

Similar issues plague disentangling that do "flat minima": a model can be re-parametrized, s.t. the distribution over the data is unchanged, but it can be arbitrarily more "entangled".

Thus, some kind of inductive bias both on model class and data seems necessary.

As a simple example: consider $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$, let $\boldsymbol{z}' = \boldsymbol{U}\boldsymbol{z}$, for any non-identity orthogonal matrix U.

Then, under any "intuitive" understanding of entangling, $\boldsymbol{z}'$ seems entangled with $\boldsymbol{z}$ – small changes of coordinates of z cause global changes in $\boldsymbol{z}'$.

Issue of ill-posedness?

Locatello et al '19, "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations" (Best paper award at ICML '19):

Similar issues plague disentangling that do "flat minima": a model can be re-parametrized, s.t. the distribution over the data is unchanged, but it can be arbitrarily more "entangled".

Theorem 1. *For $d > 1$, let $\mathbf{z} \sim P$ denote any distribution which admits a density $p(\mathbf{z}) = \prod_{i=1}^{d} p(\mathbf{z}_i)$. Then, there exists an infinite family of bijective functions $f : \mathrm{supp}(\mathbf{z}) \to \mathrm{supp}(\mathbf{z})$ such that $\frac{\partial f_i(\mathbf{u})}{\partial u_j} \neq 0$ almost everywhere for all i and j (i.e., $\mathbf{z}$ and $f(\mathbf{z})$ are completely entangled) and $P(\mathbf{z} \leq \mathbf{u}) = P(f(\mathbf{z}) \leq \mathbf{u})$ for all $\mathbf{u} \in \mathrm{supp}(\mathbf{z})$ (i.e., they have the same marginal distribution).*

***Reparametrization**:*
z'=f(z) is "entangled" wrt to z