Autoencoder and Embedding
Figure 1: Fifteen consecutive frames from the video. The experimental setup involved: a transparent bucket of water, the cover of a vision textbook “Computer Vision/A Modern Approach”.

Figure 2: Ground truth image and reconstruction results using mean and median
Figure 7: **Top row:** sample patches (two different locations) from 800 frames, **Middle row:** Convex flow embedding, showing the transition paths. **Bottom row:** corresponding patches (A, B, C, A1, A2, B1, B2, C1, C2) and the morphing of them to the centers F, FA, FB, FC respectively.
IsoMap

Find estimated geodesic distances between all pairs in $X$:
Figure 4: The leakage problem. **Left:** Equivalence of shortest path leakage and uncapacitated flow leakage problem. **Bottom-middle:** After the erroneous edge is inserted, the shortest paths from the top of the triangle to vertex $v$ go through this edge. **Boxes A-C:** Alternatives for charging a unit of flow between nodes $u$ and $w$. The horizontal axis of the plots is the amount of flow and the vertical axis is the cost. **Box A:** Linear flow. The cost of a unit of flow is $d_1$. **Box B:** Convex flow. Multiple edges are introduced between two nodes, with fixed capacity, and convexly increasing costs. The cost of a unit of flow increases from $d_1$ to $d_2$ and then to $d_3$ as the amount of flow from $u$ to $w$ increases. **Box C:** Linear flow with capacity. The cost is $d_1$ until a capacity of $c_1$ is achieved and becomes infinite afterwards.
Figure 5: **Top row:** Ground truth. After sampling points from a triangular disk, a kNN graph is constructed to provide a local measure for the embedding (left). Additional erroneous edges $AC$ and $CB$ are added to perturb the local measure (middle, right). **Middle row:** Isomap embedding. Isomap recovers the manifold for the error-free cases (left). However, all-pairs shortest path can “leak” through $AC$ and $CB$, resulting in a significant change in the embedding. **Bottom row:** Convex flow embedding. Convex flow penalized too many paths going through the same edge – correcting the leakage problem. The resulting embedding is more resistant to perturbations in the kNN graph.
Figure 6: Comparison of reconstruction results of different methods using the first 800 frames, top: patches stitched together which are closest to mean (left) and median (right), bottom: our results using a single (left) and three (right) centers
This PCA visualization is terrible
Dimensionality Reduction contd ...

Aarti Singh

Machine Learning 10-601
Nov 10, 2011

Slides Courtesy: Tom Mitchell, Eric Xing, Lawrence Saul
Principal Components are the eigenvectors of the matrix of sample correlations $XX^T$ of the data.

New set of axes $V = [v_1, v_2, \ldots, v_D]$ where $XX^T = V\Lambda V^T$

- Geometrically: centering followed by rotation
  - Linear transformation

Original representation of data points $x_i = [x_i^1, x_i^2, \ldots, x_i^D]$

$x_i^j = e_j^T x_i$ where $e_j = [0 \ldots 0 1 0 \ldots 0]$

$j^{th}$ coordinate

Transformed representation of data points $[v_1^T x_i, v_2^T x_i, \ldots, v_D^T x_i]$
Original Representation \([x_i^1, x_i^2, \ldots, x_i^D]\) (D-dimensional vector)

\[
    x_i = \sum_{j=1}^{D} x_i^j e_j = \sum_{j=1}^{D} (e_j^T x_i) e_j
\]

\((x_i^j)^2 = (e_j^T x_i)^2 = \text{energy/variance of data point } i \text{ along coordinate } j\)

Transformed representation \([v_1^T x_i, v_2^T x_i, \ldots, v_D^T x_i]\) (D-dimensional vector)

\[
    x_i = \sum_{j=1}^{D} (v_j^T x_i) v_j
\]

\((v_j^T x_i)^2 = \text{energy/variance of data point } i \text{ along principal component } v_j\)

\[
    \lambda_j = \sum_{i=1}^{n} (v_j^T x_i)^2 = \text{energy/variance of all points along } v_j
\]

Dimensionality reduction \([v_1^T x_i, v_2^T x_i, \ldots, v_d^T x_i]\) (d-dimensional vector)

\[
    \mathbf{x}_i = \sum_{j=1}^{d} (v_j^T x_i) v_j
\]

Only keep data projections onto principal components which capture enough energy/variance of the data \(\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_D\)
Maximum Variance Subspace: PCA finds vectors $v$ such that projections on to the vectors capture maximum variance in the data

$$
\sum_{i=1}^{n} (v^T x_i)^2 = v^T X X^T v
$$

Minimum Reconstruction Error: PCA finds vectors $v$ such that projection on to the vectors yields minimum MSE reconstruction

$$
\sum_{i=1}^{n} \|x_i - (v^T x_i)v\|^2
$$

One direction approximation

Recall: $x_i = \sum_{k} (v_k^T x_i) \cdot v_k$
**Principal Components** – Eigenvectors of $X X^T$ (D x D matrix)

Problematic for high-dimensional datasets!

Another way to compute PCs: **Singular Vector Decomposition (SVD)**

$$X = \tilde{V} S U^T \Rightarrow X X^T = \tilde{V} S U^T U S \tilde{V}^T = \tilde{V} S^2 \tilde{V}^T = \tilde{V} \Lambda \tilde{V}^T$$

**Factorized embedding**

$x_i = \begin{pmatrix} x_i^1 \\ x_i^2 \\ \vdots \\ x_i^D \end{pmatrix}$

$X = \begin{pmatrix} D \times n \\ \end{pmatrix}$

$\tilde{V} = \begin{pmatrix} \end{pmatrix}$

$S = \begin{pmatrix} n \times n \\ \end{pmatrix}$

$U^T = \begin{pmatrix} n \times n \\ \end{pmatrix}$

Singular values = $\sqrt{\text{eigenvalues}}$

Left singular vectors

$\nu_j = \text{Principal Components}$

Right singular vectors
Factorized embedding

\[ X = \tilde{V}SU^T \quad \Rightarrow \quad XX^T = \tilde{V}SU^TUS\tilde{V}^T = \tilde{V}S^2\tilde{V}^T = \tilde{V}\Lambda\tilde{V}^T \]

\[ x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{iD} \end{bmatrix} \quad \text{D x n} \]

\[ \tilde{V} \quad \text{Singular values} \quad \text{= \sqrt{eigenvalues}} \quad \text{Right singular vectors} \]

\[ U^T \quad \text{Left singular vectors} \quad v_j = \text{Principal Components} \]
Factorized embedding

\[ \text{data } X \approx \text{dictionary } W \approx \text{activations } H \]
- PCA seeks “orthogonal” directions that capture maximum variance in data, or that minimize squared reconstruction error.

- ICA seeks “statistically independent” directions in the data.
Nonnegative matrix factorisation

- data $V$ and factors $W$, $H$ have nonnegative entries.
- nonnegativity of $W$ ensures interpretability of the dictionary, because patterns $w_k$ and samples $v_n$ belong to the same space.
- nonnegativity of $H$ tends to produce part-based representations, because subtractive combinations are forbidden.

PCA dictionary with $K = 25$

red pixels indicate negative values
NMF dictionary with $K = 25$

experiment reproduced from (Lee and Seung, 1999)
Dimensionality Reduction

“Unrolling the swiss roll”
Laplacian Eigenmaps

Linear methods – Lower-dimensional linear projection that preserves distances between all points

Laplacian Eigenmaps (key idea) – preserve local information only

Construct graph from data points (capture local information)

Project points into a low-dim space using “eigenvectors of the graph”
Nonlinear Embedding Results
Step 1 - Graph Construction

Similarity Graphs: Model local neighborhood relations between data points

\[ G(V,E,W) \]

V – Vertices (Data points)

E – Edges

1. \( E \) – Edge if \( ||x_i - x_j|| \leq \varepsilon \) \((\varepsilon \text{ – neighborhood graph})\)

2. \( E \) – Edge if k-nearest neighbor (k-NN graph)
Step 1 - Graph Construction

Similarity Graphs: Model local neighborhood relations between data points

(2) \( E \) – Edge if k-nearest neighbor (k-NN)

yields directed graph

connect A with B if \( A \rightarrow B \) OR \( A \leftarrow B \) (symmetric kNN graph)
connect A with B if \( A \rightarrow B \) AND \( A \leftarrow B \) (mutual kNN graph)
Step 1 - Graph Construction

Similarity Graphs: Model local neighborhood relations between data points

$G(V,E,W)$

$V$ – Vertices (Data points)

$E$ – Edges

$W$ – Weights

(1) $W - W_{ij} = 1$ if edge present, 0 otherwise

(2) $W_{ij} = e^{-\frac{||x_i - x_j||^2}{2\sigma^2}}$  Gaussian kernel similarity function (aka Heat kernel)
Original Representation data point
$x_i$ → Transformed representation projections
(D-dimensional vector)               (d-dimensional vector)

Basic Idea: Find vector $f$ such that, if $x_i$ is close to $x_j$ in the graph (i.e. $W_{ij}$ is large), then projections of the points $f(i)$ and $f(j)$ are close

$$\min_f \sum_{ij} W_{ij} (f_i - f_j)^2$$
Step 2 – Projection using Graph

- Graph Laplacian (unnormalized version)
  \[ L = D - W \] (n x n matrix)

  \( W \) – Weight matrix

  \( D \) – Degree matrix = \( \text{diag}(d_1, \ldots, d_n) \)

  \[ d_i = \sum_j w_{ij} \] degree of a vertex

**Note:** If graph is connected, 1 is an eigenvector with 0 as eigenvalue.
Step 2 – Projection using Graph

- Justification – points connected on the graph stay as close as possible after embedding

\[
\min_f \sum_{i,j} W_{ij} (f_i - f_j)^2 \equiv \min_f f^T L f
\]

\[
\text{RHS} = f^T (D-W) f = f^T D f - f^T W f = \sum_i d_i f_i^2 - \sum_{i,j} f_i f_j w_{ij}
\]

\[
= \frac{1}{2} \left( \sum_i (\sum_j w_{ij} f_i^2) - 2 \sum_i f_i f_j w_{ij} + \sum_j (\sum_i w_{ij} f_j^2) \right)
\]

\[
= \frac{1}{2} \sum_{i,j} w_{ij} (f_i - f_j)^2 = \text{LHS}
\]
Step 2 – Projection using Graph

Justification – points connected on the graph stay as close as possible after embedding

$$\min_f \sum_{i,j} W_{ij}(f_i - f_j)^2 \equiv \min_f f^T L f \quad s.t. \ f^T f = 1$$

Similar to PCA with $XX^T$ replaced by $L$

Lagrangian: $$\min_f f^T L f - \lambda f^T f$$

Wrap constraint into the objective function

$$\frac{\partial}{\partial f} = 0 \quad (L - \lambda I) f = 0$$

$$Lf = \lambda f$$
Step 2 – Projection using Graph Laplacian

- Graph Laplacian (unnormalized version)
  \[ L = D - W \]  \((n \times n \text{ matrix})\)

Find eigenvectors of the graph Laplacian \[ Lf = \lambda f \]

Ordered eigenvalues \(0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \ldots \leq \lambda_n\)

To embed data points in \(d\)-dim space, project data points onto eigenvectors associated with \(\lambda_1, \lambda_2, \ldots, \lambda_d\)

<table>
<thead>
<tr>
<th>Original Representation</th>
<th>Transformed representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>data point (x_i)</td>
<td>projections ((f_1(i), \ldots, f_d(i)))</td>
</tr>
<tr>
<td>(D-dimensional vector)</td>
<td>(d-dimensional vector)</td>
</tr>
</tbody>
</table>
Unsupervised embedding

Unrolling the swiss roll

\[ f_2 \]

\[ f_3 \]

\[ N = 5 \quad t = 5.0 \]

\[ N = 10 \quad t = 5.0 \]

\[ N = 15 \quad t = 5.0 \]

\[ N = 5 \quad t = 25.0 \]

\[ N = 10 \quad t = 25.0 \]

\[ N = 15 \quad t = 25.0 \]

\[ N = 5 \quad t = \infty \]

\[ N = 10 \quad t = \infty \]

\[ N = 15 \quad t = \infty \]

N=number of nearest neighbors, \( t \) = the heat kernel parameter (Belkin & Niyogi’03)
unsupervised embedding

**Dimensionality Reduction Methods**

- **Feature Selection** - Only a few features are relevant to the learning task
  
  Score features (mutual information, prediction accuracy, domain knowledge)
  
  Regularization

- **Latent features** – Some linear/nonlinear combination of features provides a more efficient representation than observed feature
  
  **Linear:** Low-dimensional linear subspace projection
  
  PCA (Principal Component Analysis),
  
  MDS (Multi Dimensional Scaling),
  
  Factor Analysis, ICA (Independent Component Analysis)

  **Nonlinear:** Low-dimensional nonlinear projection that preserves local information along the manifold
  
  Laplacian Eigenmaps
  
  ISOMAP, Kernel PCA, LLE (Local Linear Embedding),
  
  Many, many more ...
unsupervised embedding

One the Role and Impact of the Metaparameters in $t$-distributed Stochastic Neighbor Embedding

John A. Lee and Michel Verleysen
Machine Learning Group
Université catholique de Louvain
Louvain-la-Neuve, Belgium
michel.verleysen@uclouvain.be

---

t-SNE Method for High-Dimensional Data Visualization

David Khosid

2017
unsupervised embedding

Results with Euclidean distances

On the role and impact of the metaparameters in t-distributed SNE
unsupervised embedding

SNE and t-SNE

- In the original space, the similarity between $y_i$ and $y_j$ is defined as

$$p_{j|i}(\lambda_i) = \begin{cases} 0 & \text{if } i = j \\ \frac{g(\delta_{ij}/\lambda_i)}{\sum_{k \neq i} g(\delta_{ik}/\lambda_i)} & \text{otherwise} \end{cases}$$

$$g(u) = \exp\left(\frac{-u^2}{2}\right)$$

- Similarities are not symmetric (individual widths)!
- $p_{j|i}$ is the empirical probability of $y_j$ to be a neighbor of $y_i$
unsupervised embedding

Similarity of datapoints \((x_i)\) in data space \(R^D\)

\[
p_{j|i} = \frac{\exp\left(- \frac{\|x_j-x_i\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq m} \exp\left(- \frac{\|x_k-x_m\|^2}{2\sigma_i^2}\right)}
\]

\(p_{j|i}\) measures how close \(x_j\) is from \(x_i\), considering Gaussian distribution around \(x_i\) with a given variance \(\sigma_i^2\).

Similarity of datapoints \((x_i)\) in data space \(R^D\)

\[
p_{j|i} = \frac{\exp\left(- \frac{\|x_j-x_i\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq m} \exp\left(- \frac{\|x_k-x_m\|^2}{2\sigma_i^2}\right)}
\]

Make the similarity metric \(p_{ij}\) symmetric. The main advantage of symmetry is simplifying the gradient (learning stage):

\[
p_{ij} = \frac{p_{ij} + p_{ji}}{2N}
\]

- we set \(p_{ii} = 0\), as we interested in pairwise similarities
- \(\sigma_i\) is chosen such that the data point has a fixed perplexity (effective number of neighbors).
unsupervised embedding

SNE and t-SNE

- In the embedding space, the similarity between \( x_i \) and \( x_j \) is defined as
  
  \[
  q_{ij}(n) = \begin{cases} 
  0 & \text{if } i = j \\
  \frac{t(d_{ij}, n)}{\sum_{k \neq l} t(d_{kl}, n)} & \text{otherwise}
  \end{cases}
  \]

  \[
  t(u, n) = \left(1 + \frac{u^2}{n}\right)^{-\frac{n+1}{2}}
  \]

- Similarities are symmetric
- \( t(u, n) \) is proportional to a Student t with \( n \) degrees of freedom ( \( n \) controls the thickness of the tail)
- SNE: \( n \to \infty \)    t-SNE: \( n = 1 \)
unsupervised embedding

Student t-distribution with 1DoF (same as Cauchy distribution)

\[ q_{ij} = \frac{\frac{1}{1+\|y_i-y_j\|^2}}{\sum_{k \neq m} \frac{1}{1+\|y_k-y_m\|^2}} \]  

- we set \( q_{ii} = 0 \), as we interested in pairwise similarities
- heavy-tail (will be discussed later)
- still closely related to the Gaussian
- computationally convenient (no exponent)

\[ C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \]

**KL divergence:**
- is not a distance, since it is asymmetric
- large \( p_{ij} \) modelled by small \( q_{ij} \) → large penalty
- Small \( p_{ij} \) modelled by large \( q_{ij} \) → small penalty
- **KL divergence** meaning: cross-entropy
unsupervised embedding

t-SNE algorithm minimizes KL divergence between $P$ and $Q$ distributions.

\[
\frac{\partial C}{\partial y_i} = 4 \sum_{i \neq j} (p_{ij} - q_{ij}) \frac{y_i - y_j}{1 + \|y_i - y_j\|^2}
\]

- positive $\rightarrow$ attraction
- negative $\rightarrow$ repulsion
  (dissimilar DPs, similar MPs) $\rightarrow$ repulsion

\[
Y^{(t)} = Y^{(t-1)} + \eta \frac{\partial C}{\partial Y} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)})
\]

Spring Analogy: $F = -k \times (y_i - y_j)$, attraction/repulsion
Supervised vs. unsupervised embedding

Toy Example of MNIST Dataset
Learning Features for discriminative embedding

\[ d_g(a, b, c) = \mathbb{1}(c = g)d(a, b) + \mathbb{1}(c \neq g) \max(\delta - d(a, b), 0) \]
Features as embedding
Autoencoders

• Generated images need to be sharp
  • Generated images from auto-encoder tend to be blurry.
Autoencoders

Image $x$ → Deep Neural Network → Deep Feature $\phi$

Representation of Image $x$ in a Condensed Version
Autoencoders

Supervised scenario: label y to train the network to get the feature

- **image x**
- **deep neural network**
- **label y**

**deep feature $\phi$**
- Representation of image $x$
- Less spatial resolution, more feature descriptions
Autoencoders

Unsupervised scenario: if we don’t have any label, how can we train it?

image x → deep neural network → deep feature $\phi$

Representation of image $x$
Less spatial resolution, more feature descriptions
Autoencoders

Autoencoders: $||x-x'||$ as loss, use back propagation to train the network

representation of image $x$ 

Space vs features: Less spatial resolution, more feature descriptions
Autoencoders

Autoencoders: $||x-x'||$ as loss, use back propagation to train the network

- **image x**
- **encoder**
- **decoder**
- **reconstructed image x’**

Deep feature $\phi$

Representation of image x

Less spatial resolution, more feature descriptions
Autoencoders

The encoder maps image space to feature space (self-learned)

image x $\xrightarrow{\text{encoder}}$ deep feature $\phi$ $\xrightarrow{\text{decoder}}$ reconstructed image $x'$

Less spatial resolution, more feature descriptions
Autoencoders

The decoder maps feature space to image space

image $x$ \rightarrow \text{encoder} \rightarrow \text{deep feature } \phi \rightarrow \text{decoder} \rightarrow \text{reconstructed image } x'$

Representation of image $x$
Less spatial resolution, more feature descriptions
Autoencoders

Can this process be generalized?
Goal of generative model: Generate realistic image from a random vector.
Problem: How to train it?

random vector $z$ from a predefined distribution (e.g. Gaussian) -> decoder -> generated image $x$
Autoencoders

Can this process be generalized?
Goal of generative model: Generate realistic image from a random vector.
Problem: How to train it?

random vector $z$ from a predefined distribution (e.g. Gaussian)

Latent variable: attributes of the image
1. Variational Autoencoders

- Variational Autoencoders
- Generative Adversarial Network

**Models**

**1. Variational Autoencoders**

- **Generative model**
- Discriminative model

**Sample z from**

\[ z|x \sim N(\mu_{z|x}, \Sigma_{z|x}) \]

**Sample x from**

\[ x|z \sim N(\mu_{x|z}, \Sigma_{x|z}) \]

**2. Generative Adversarial Network**

- Generative model
- Discriminative model

**Model sample**

\[ G(z) \]

**Data sample**

\[ x \]
Variational Autoencoders

if we can train a model to approximate $p(x|z)$, then we can generate the image by:
(1) Draw a latent variable $z \sim p(z)$
(2) Draw a data point $x \sim p(x|z)$
Variational Autoencoders

if we can train a model to approximate $p(x|z)$, then we can generate the image by:

1. Draw a latent variable $z \sim p(z)$
2. Draw a data point $x \sim p(x|z)$

$z \sim p(z)$
Simple distribution
(e.g. Multivariate Gaussian)

$x \sim p(x)$
Variational Autoencoders

if we can train a model to approximate $p(x|z)$, then we can generate the image by:

1. Draw a latent variable $z \sim p(z)$
2. Draw a data point $x \sim p(x|z)$

Complex model, use deep neural network
Variational Autoencoders

Training strategy:
Given a set of training data (images) $x_1, \ldots, x_n$, train the network to maximize the likelihood of training data $p(x)$
Variational Autoencoders

\[ p(x) = \int p(z)p(x|z)dz \]

\( z \) is continuous, intractable to compute \( p(x|z) \) for every \( z \)

\( z \sim p(z) \quad \Rightarrow \quad p(x|z) \quad \Rightarrow \quad x \sim p(x) \)
Training as an autoencoder

Training use maximum likelihood of $p(x)$ given the training data

$z \sim p(z)$ multivariate Gaussian
$x \mid z \sim p_\theta(x \mid z)$

Problem:

$p_\theta(z \mid x)$

Cannot be calculated:

Solution:
- MCMC (too costly)
- Approximate $p(z \mid x)$ with $q(z \mid x)$
Variational Autoencoders

1. Introduce an encoder network to model $q(z|x) \approx p(z|x)$
2. Introduce Gaussian distribution to represent the output of the encoder and decoder

We will show later that, this gives tractable lower bound of $p(x)$
Variational Autoencoders

For each image $x$ from training example:

$$L = \log(p(x)) = \sum_z q(z|x) \log(p(x))$$

Multiply with 1

sample $z$ from $z|x \sim N(\mu_{z|x}, \Sigma_{z|x})$

sample $x$ from $x|z \sim N(\mu_{x|z}, \Sigma_{x|z})$
Variational Autoencoders

For each image $x$ from training example:

$$L = \log(p(x))$$

$$= \sum_z q(z|x) \log(p(x))$$

$$= \sum_z q(z|x) \log \left( \frac{p(z,x)}{p(z|x)} \right)$$

$$= \sum_z q(z|x) \log \left( \frac{p(z,x)}{p(z|x) q(z|x)} \right)$$

$$= \sum_z q(z|x) \log \left( \frac{p(z,x)}{q(z|x)} \right) + \sum_z q(z|x) \log \left( \frac{q(z|x)}{p(z|x)} \right)$$

$$= L^v + D_{KL}(q(z|x) || p(z|x))$$
Variational Autoencoders

For each image $x$ from training example:

\[
L = \log(p(x)) = \sum_z q(z|x) \log\left(\frac{p(z,x)}{p(z|x)}\right) = \sum_z q(z|x) \log\left(\frac{p(z,x)q(z|x)}{p(z|x)q(z|x)}\right) = \sum_z q(z|x) \log\left(\frac{p(z,x)}{q(z|x)}\right) + \sum_z q(z|x) \log\left(\frac{q(z|x)}{p(z|x)}\right)
\]

\[
= \underbrace{L^\nu + D_{KL}(q(z|x)\|p(z|x))}_{\text{Variational lower bound of } L}
\]

Multiply with 1
Bayes role
Multiply with 1
Logarithms

KL divergence $\geq 0$
Variational Autoencoders

For each image $x$ from training example:

$$
L^v = \sum_z q(z|x) \log \left( \frac{p(z,x)}{q(z|x)} \right)
= \sum_z q(z|x) \log \left( \frac{p(x|z)p(z)}{q(z|x)} \right)
= \sum_z q(z|x) \log \left( \frac{p(z)}{q(z|x)} \right) + \sum_z q(z|x) \log(p(x|z))
= -D_{KL}(q(z|x) \| p(z)) + \mathbb{E}_{q(z|x)}(\log(p(x|z)))
$$

Bayes role

Logarithms

KL divergence between Gaussian for encoder and prior Gaussian $z$

Close form solution

Also known as regularization term
Variational Autoencoders

For each image $x$ from training example:

$$L^v = \sum_z q(z|x) \log \left( \frac{p(z,x)}{q(z|x)} \right)$$

$$= \sum_z q(z|x) \log \left( \frac{p(x|z)p(z)}{q(z|x)} \right)$$

$$= \sum_z q(z|x) \log \left( \frac{p(z)}{q(z|x)} \right) + \sum_z q(z|x) \log(p(x|z))$$

$$= -D_{KL}(q(z|x)\|p(z)) + \mathbb{E}_{q(z|x)}(\log(p(x|z)))$$

Reconstruction quality from the decoder
Can be estimated by sampling
Variational Autoencoders

\[ L^v = -D_{KL}(q(z|x)\|p(z)) + \mathbb{E}_{q(z|x)}(\log(p(x|z))) \]

Given a minibatch of training images \( x \), pass through the encoder.
Variational Autoencoders

\[ L^v = -D_{KL}(q(z|x)||p(z)) + \mathbb{E}_{q(z|x)}(\log(p(x|z))) \]

KL divergence between two Gaussians, close form solution.
Avoid the encoder from “cheating” by projecting similar images to far latent vectors in latent space.
Variational Autoencoders

\[ L' = -D_{KL}(q(z|x)||p(z)) + \mathbb{E}_{q(z|x)}(\log(p(x|z))) \]

Sample \( z \) from \( z|x \sim N(\mu_{z|x}, \Sigma_{z|x}) \)
Variational Autoencoders

\[ L^v = -D_{KL}(q(z|x) || p(z)) + \mathbb{E}_{q(z|x)}(\log(p(x|z))) \]

Pass z through the decoder.

\[ x \overset{q(x|z)}{\rightarrow} \mu_{z|x}, \Sigma_{z|x} \overset{\text{sample } z \text{ from}}{\rightarrow} z|x \sim N(\mu_{z|x}, \Sigma_{z|x}) \overset{p(z|x)}{\rightarrow} \mu_{x|z}, \Sigma_{x|z} \]
Variational Autoencoders

\[ L' = -D_{KL}(q(z|x) || p(z)) + E_{q(z|x)}(\log(p(x|z))) \]

Approximate by sampling \( z \) from \( q(z|x) \)
Maximize the likelihood of the original input being reconstructed
Sampling to calculate
\[ \mathbb{E}_{q(z|x^{(i)})} \left( \log \left( p(x^{(i)}|z) \right) \right) \]

Approximating \( \mathbb{E}_{q(z|x^{(i)})} \) with sampling form the distribution \( q(z|x^{(i)}) \)

With \( z^{(i,l)} l = 1, 2, \ldots L \) sampled from \( z^{(i,l)} \sim q(z|x^{(i)}) \)
\[ L^y = -D_{KL} \left( q(z|x^{(i)}) \parallel p(z) \right) + \mathbb{E}_{q(z|x^{(i)})} \left( \log \left( p(x^{(i)}|z) \right) \right) \]
\[ L^y \approx -D_{KL} \left( q(z|x^{(i)}) \parallel p(z) \right) + \frac{1}{L} \sum_{i=1}^{L} \log \left( p(x^{(i)}|z^{(i,l)}) \right) \]

Example \( x^{(i)} \)

\[ \log(\mathbb{P}_q(x^{(i)}|z^{(i,l)})) \quad \text{where} \quad z^{(i,l)} \sim \mathcal{N}(\mu^{(i)}, \sigma_{z^{(i,l)}}) \]

\[ \log(\mathbb{P}_q(x^{(i)}|z^{(i,l)})) \quad \text{where} \quad z^{(i)}, \sim \mathcal{N}(\mu^{(i)}, \sigma_{z^{(i,l)}}) \]
Variational Autoencoders

During training, back propagation to maximize the lower bound of $p(x)$

$$L^v = -D_{KL}(q(z|x) || p(z)) + \mathbb{E}_{q(z|x)}(\log(p(x|z)))$$

sample $z$ from $z|x \sim N(\mu_{z|x}, \Sigma_{z|x})$

sample $x$ from $x|z \sim N(\mu_{x|z}, \Sigma_{x|z})$
Variational Autoencoders

During image generation
1. Sample z from prior distribution (usually N(0,1))
2. Pass through the decoder
3. Sample x from $x|z \sim N(\mu_{x|z}, \Sigma_{x|z})$ (usually $\mu_{x|z}$)

sample z from $p(z)$

sample x from $x|z \sim N(\mu_{x|z}, \Sigma_{x|z})$
Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Limit of VAE

Tend to generate blurry images.

Anders Larsen et al. 2017