

Intro al Aprendizaje No-Supervisado

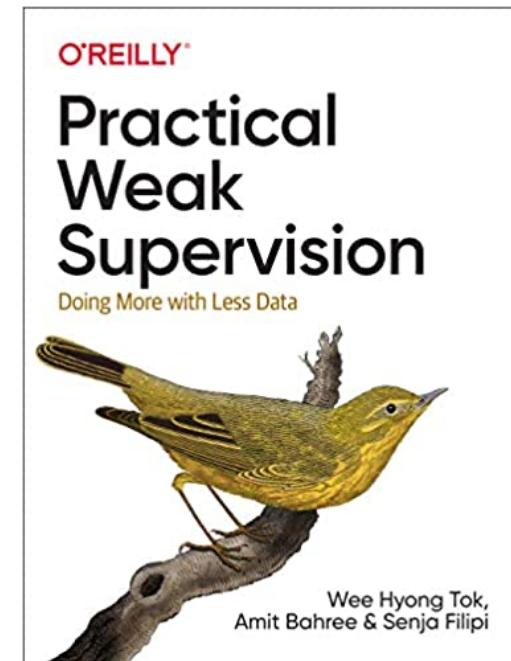
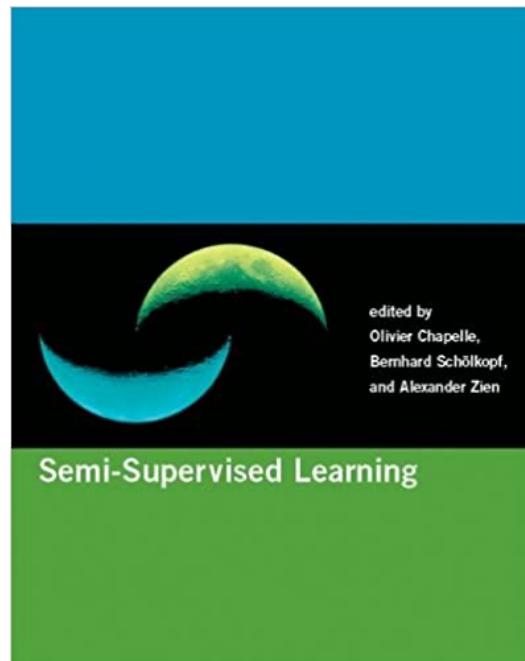
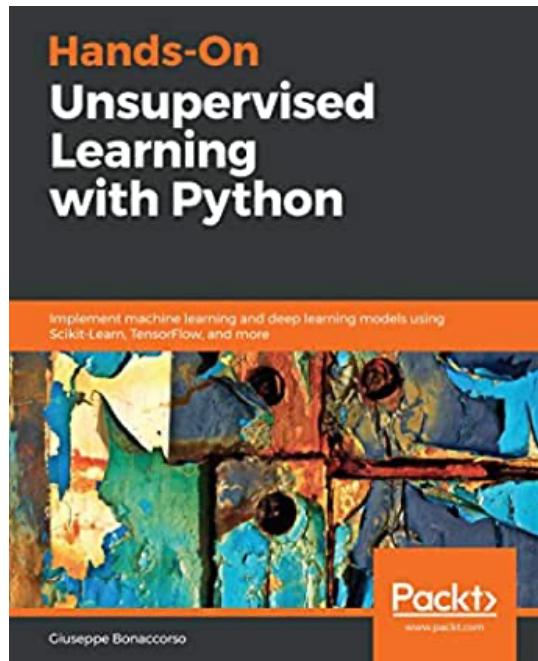
AutoEncoders (Clásicos y Variacionales)



Prof. Ricardo Ñanculef - Departamento de Informática UTFSM 2022

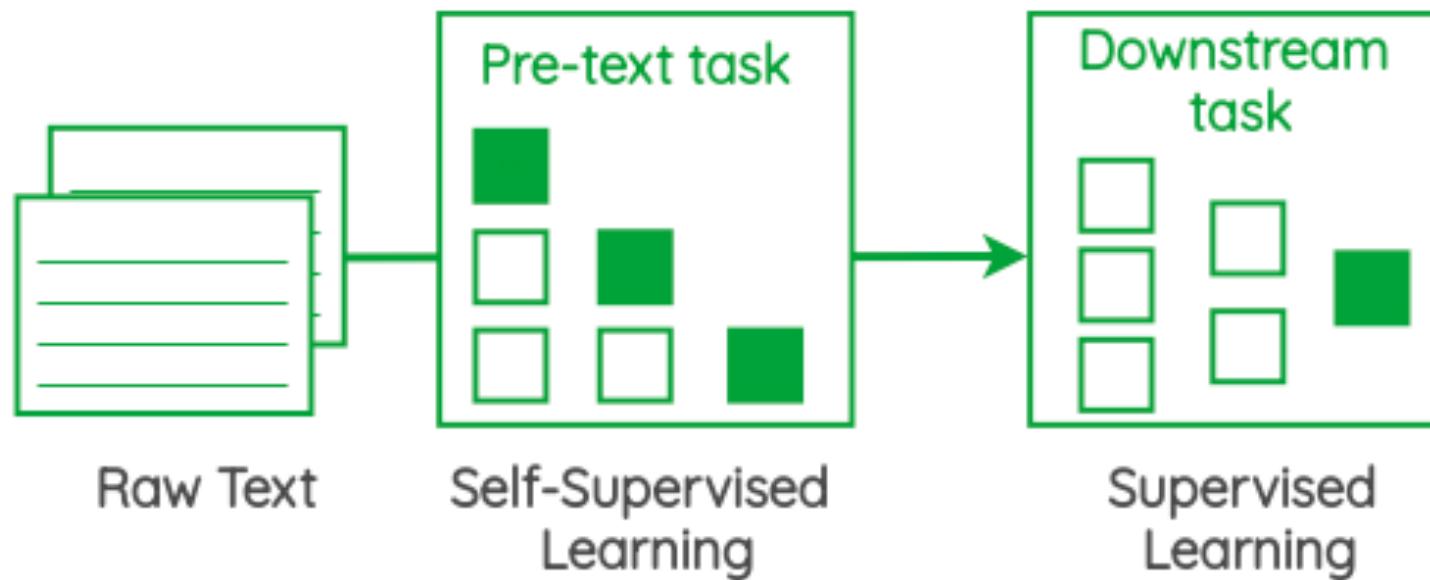
Introducción

- Obtener anotaciones explícitas para guiar el entrenamiento de una red en una aplicación es un proceso **lento, costoso, y difícil de implementar.**
- Esto ha aumentado el interés por métodos no-supervisados, semi-supervisados, auto-supervisados, débilmente supervisados, y técnicas de transferencia de conocimiento (transfer learning).



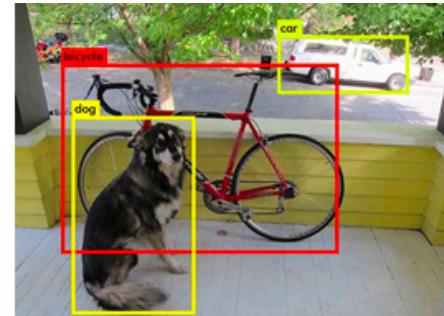
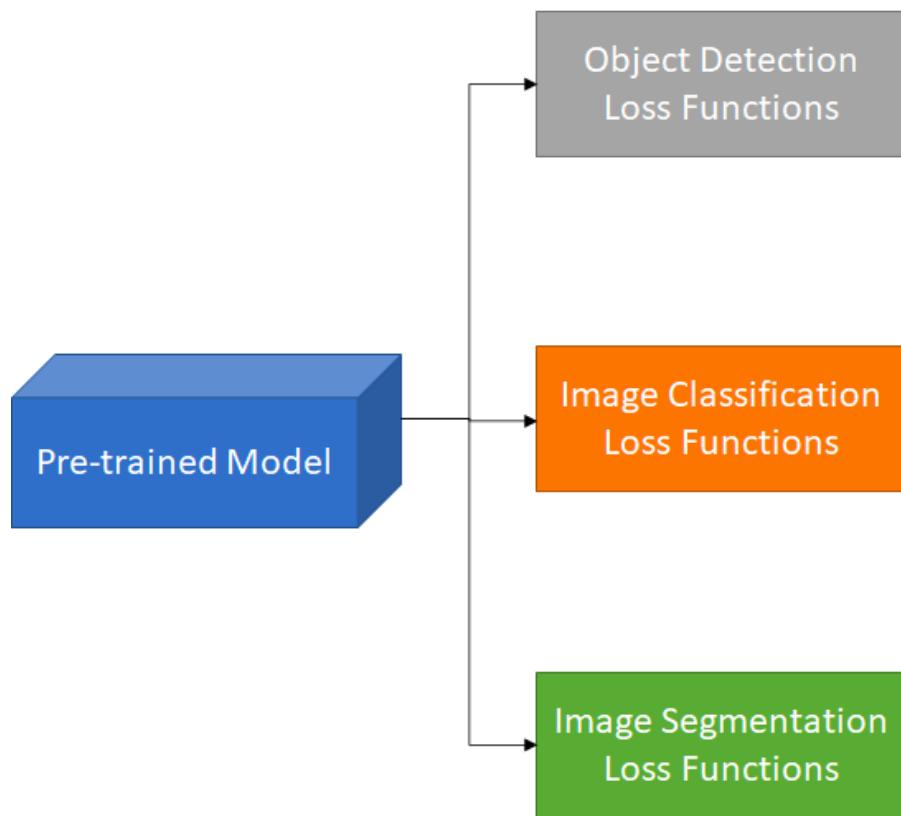
Introducción

- En muchos problemas, estas técnicas se utilizan como complemento a un entrenamiento supervisado estándar, es decir, para **pre-entrenamiento o co-entrenamiento** (función objetivo adicional), reduciendo el impacto de contar con pocas etiquetas.



Introducción

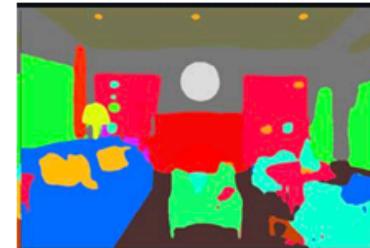
- Pre-entrenamiento



Classification

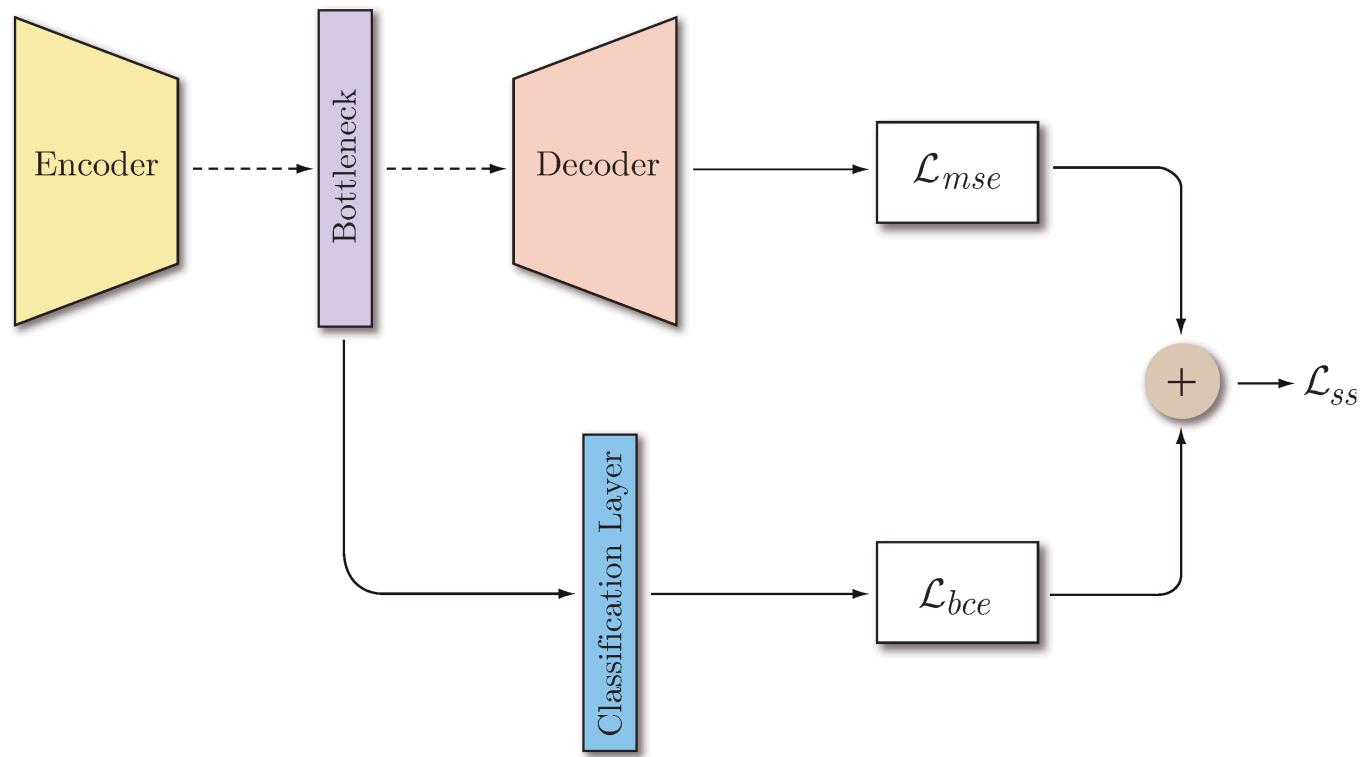


CAT



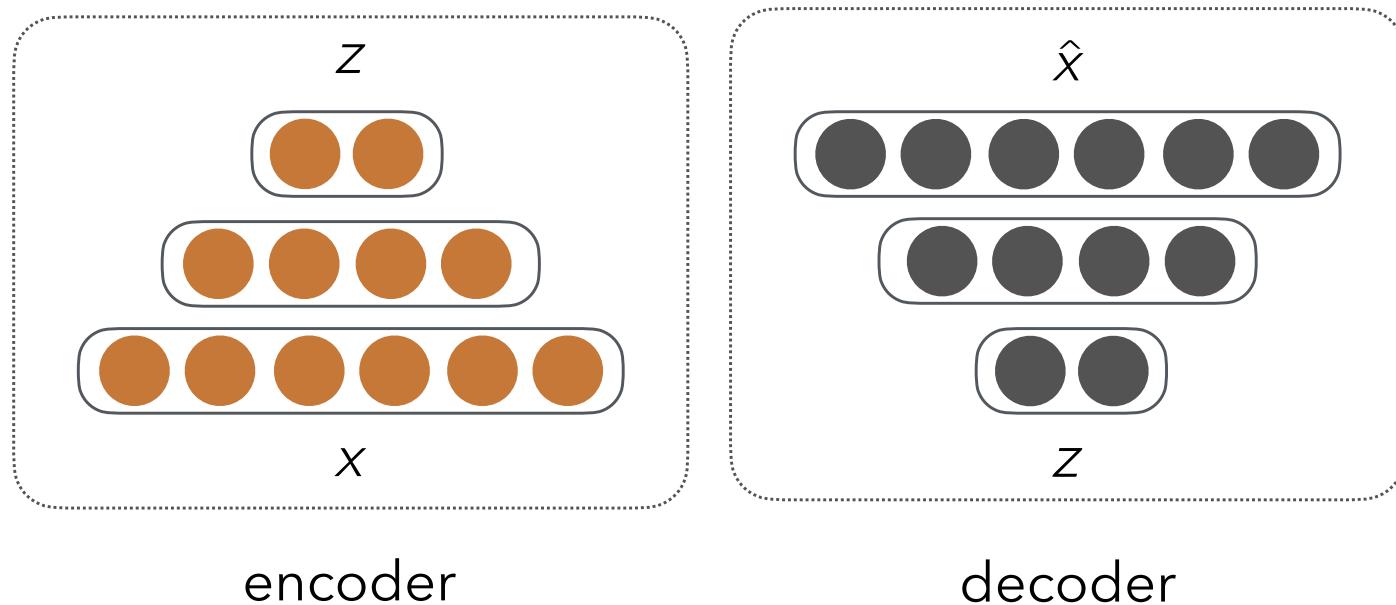
Introducción

- Co-entrenamiento



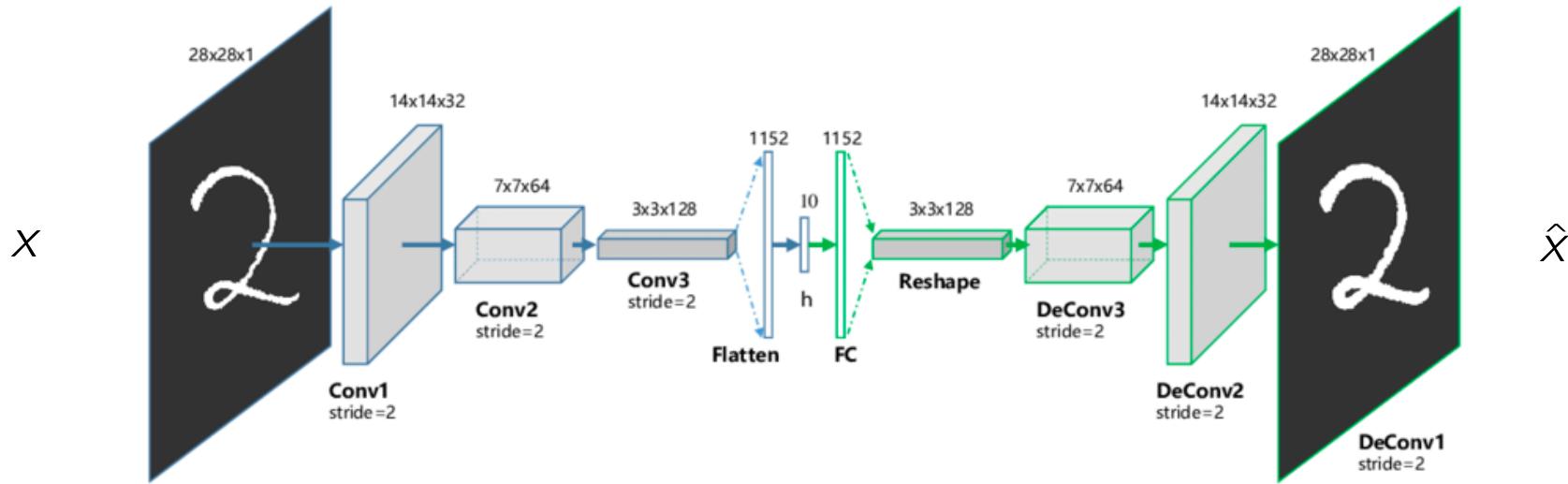
AutoEncoders Clásicos

- Un AE clásico es una red neuronal compuesta de dos subredes:
 - Una denominada **encoder** $z = h(x)$
 - Otra denominada **decoder** $\hat{x} = g(z)$



AutoEncoders Clásicos

- La particularidad de un AE es que **se entrena para reconstruir x .**



- Se trata de hecho de una de las técnicas *auto-supervisadas* más antiguas.
- El método es *no-supervisado* en el sentido de que no necesita etiquetas, solo un conjunto de datos de entrenamiento $\{x^{(\ell)}\}_{\ell=1}^n$

AutoEncoders Clásicos

- Si bien un AE no necesita etiquetas, la maquinaria de entrenamiento utilizada es supervisada: existe una salida deseada (x) para cada x .
- La función objetivo a minimizar se denomina **error de reconstrucción**:

$$\begin{aligned} J(\phi, \theta) &= \mathbb{E}_x L(\hat{x}, x) = \mathbb{E}_x L(g_\theta(h_\phi(x)), x) \\ &\approx \frac{1}{n} \sum_{\ell} L(\hat{x}^{(\ell)}, x^{(\ell)}) = \frac{1}{n} \sum_{\ell} L(g_\theta(h_\phi(x^{(\ell)})), x^{(\ell)}) \end{aligned}$$

- La loss L se debe elegir de modo consistente con la naturaleza de x . Si x es continuo (e.g. $x \in \mathbb{R}^d$) la elección típica es **MSE**:

$$J(\phi, \theta) = \mathbb{E}_x \|\hat{x} - x\|^2 \approx \frac{1}{n} \sum_{\ell} \|\hat{x}^{(\ell)} - x^{(\ell)}\|^2$$

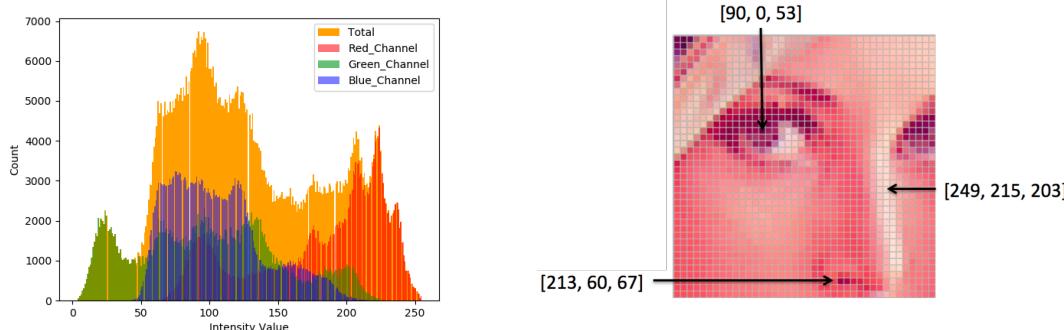


AutoEncoders Clásicos

- Si x es un tensor de valores en $[0,1]$, que podemos interpretar como probabilidades de activación (independientes), la elección típica es una **binary cross-entropy**:

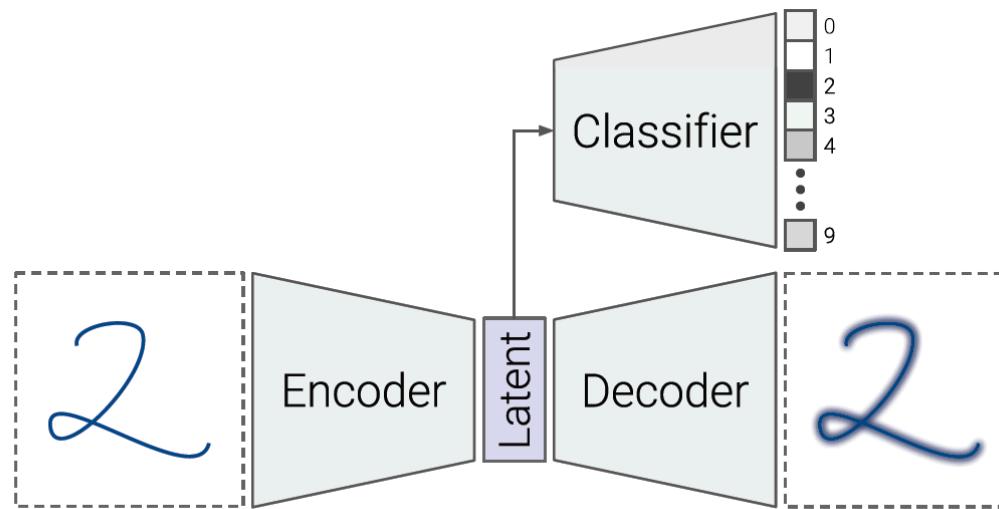
$$\begin{aligned} J(\phi, \theta) &= \mathbb{E}_x \text{CE}(\hat{x}, x) = -\mathbb{E}_x x_k \ln(\hat{x}_k) + (1 - x_k) \ln((1 - \hat{x}_k)) \\ &\approx -\frac{1}{n} \sum_{\ell} \sum_k x_k^{(\ell)} \ln(\hat{x}_k^{(\ell)}) + (1 - x_k^{(\ell)}) \ln(1 - \hat{x}_k^{(\ell)}) \end{aligned}$$

- Esta es una elección frecuente cuando se trabaja con imágenes (que han sido normalizadas de modo que la intensidad de pixel esté en $[0,1]$) o con texto representado como vectores de frecuencia de palabras (TF).



AutoEncoders Clásicos

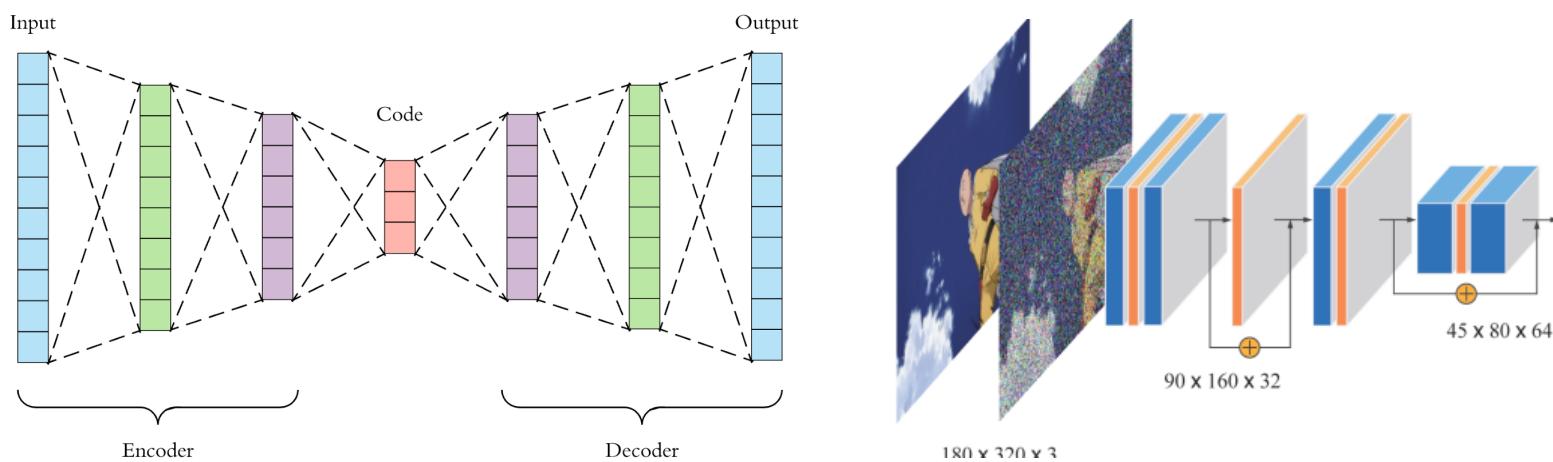
- El objetivo más común al entrenar un AE es obtener un extractor de características que luego es utilizado en alguna otra tarea (downstream task).



- Típicamente esta tarea es una aplicación para la que se cuenta con pocos datos específicos del dominio y/o con muy pocas etiquetas.
- Notar por tanto que producto final más común es el **encoder**. En este tipo de usos, el decoder es sólo una arquitectura auxiliar a la tarea de pretexto.

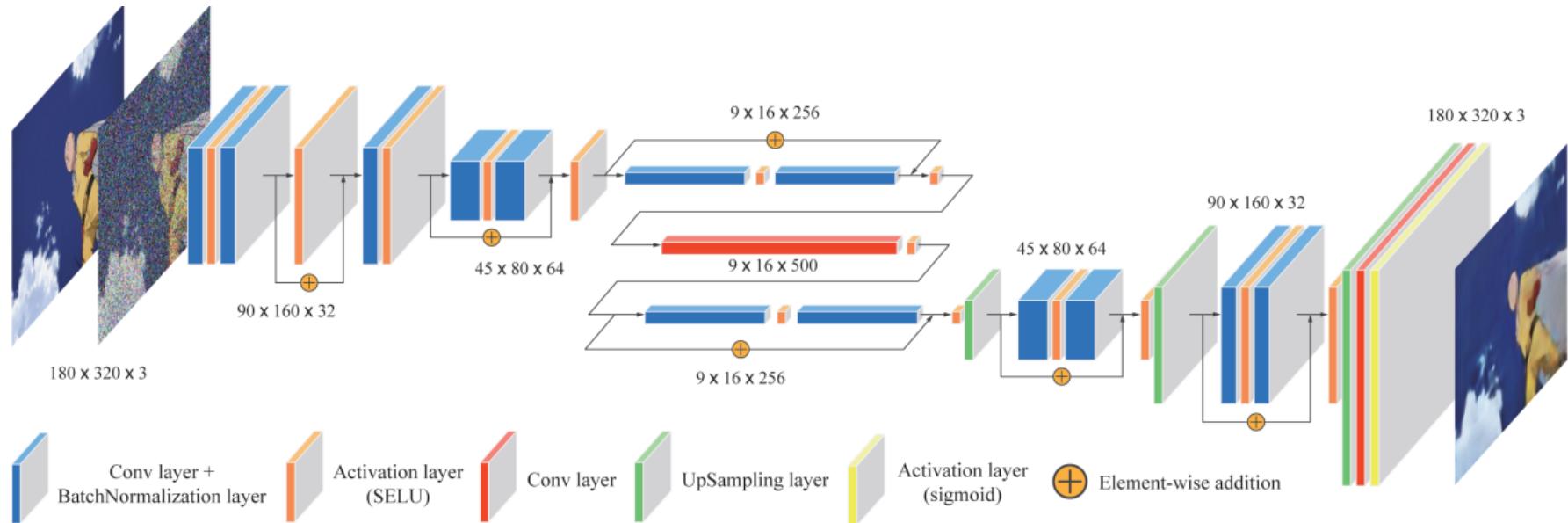
AutoEncoders Clásicos

- La arquitectura del encoder está determinada por la naturaleza de x y posiblemente por el objetivo del entrenamiento (si se está pre-entrenando una red la arquitectura podría estar dada por la aplicación final).
- El decoder suele ser una versión “traspuesta” del encoder: si la k -ésima capa del encoder implementa una transformación de \mathbb{R}^d a \mathbb{R}^k , la k -ésima capa del decoder implementa una transformación de \mathbb{R}^k a \mathbb{R}^d .



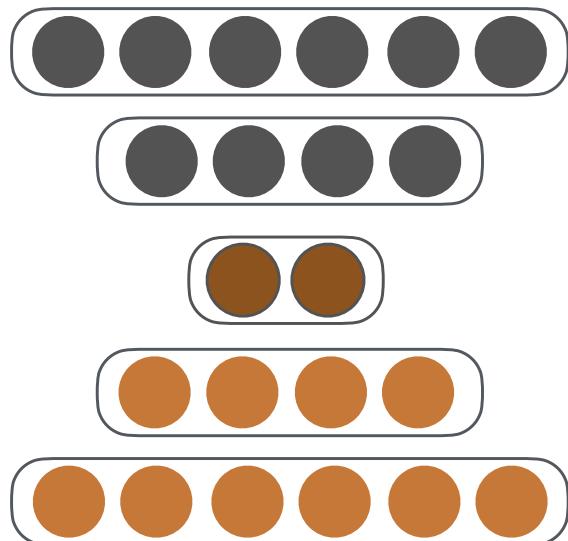
AutoEncoders Clásicos

- En el caso de encoders convolucionales, se necesitan capas que puedan implementar operaciones opuestas (al menos dimensionalmente) a una convolución y un pooling espacial.



Information Bottleneck

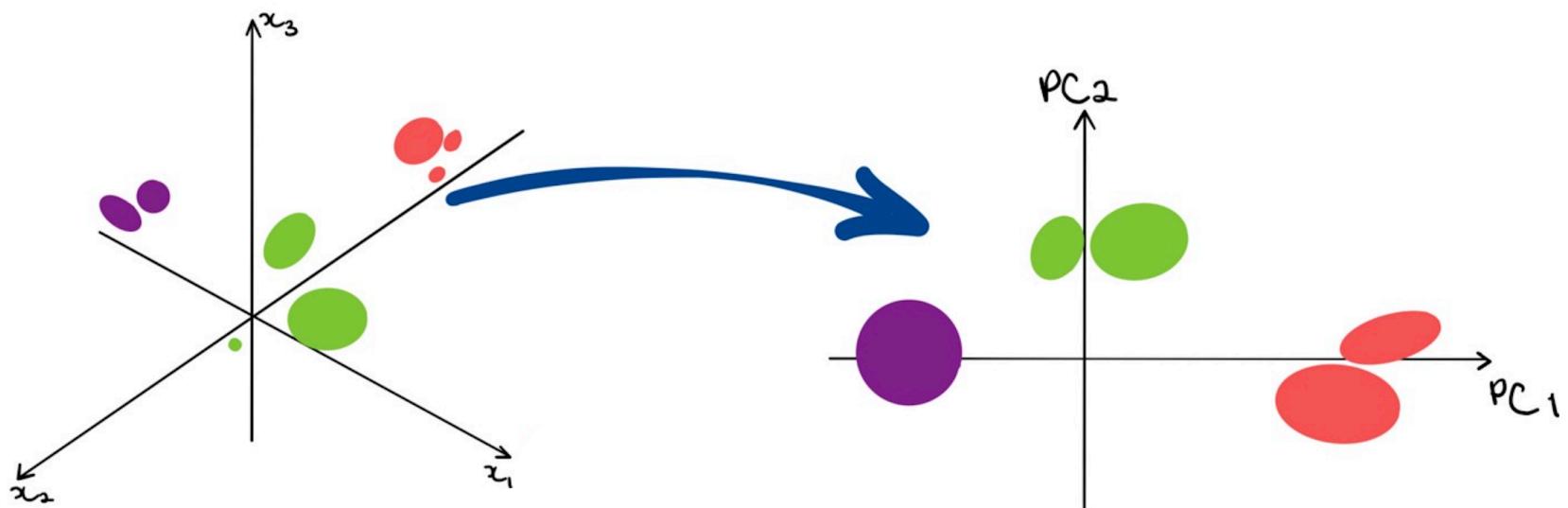
- Como el AE se entrena para aprender la función identidad, es necesario imponer algún tipo de restricción que force al modelo a aprender características relevantes y útiles de los datos de entrenamiento.
- El criterio más simple y antiguo consiste en restringir la dimensionalidad de la representación latente. Si $z \in \mathbb{R}^k$ y $x \in \mathbb{R}^d$, esto consiste en imponer la condición $k \ll d$.



- Esta condición, denominada *cuello de botella informativo*, crea de arquitecturas tipo *reloj de arena* en las que el encoder actúa como un reductor de dimensionalidad.
- Como el modelo debe ser capaz de reconstruir x a partir de z , el cuello de botella obliga al encoder a extraer las características más relevantes de x .

Relación entre un AE y PCA

- Uno de los argumentos clásicos para "demostrar" que el cuello de botella funciona consiste en demostrar que si **el AE es lineal**, minimizar el error de reconstrucción es (esencialmente) equivalente a PCA, un método clásico de reducción de dimensionalidad y arquetipo de modelo no-supervisado en estadística

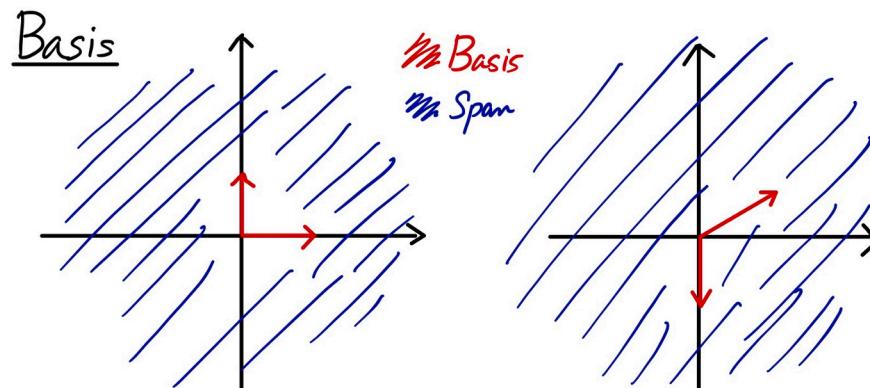


Relación entre un AE y PCA

- Dada una v.a. $x \in \mathbb{R}^d$, PCA aprende una matriz $P \in \mathbb{R}^{k \times d}$ tal que $z = Px$ preserva la mayor parte de la varianza originalmente contenida en x . Si nos aseguramos que $\mathbb{E}(x) = 0$, se cumple que $\mathbb{E}(z) = 0$. Por lo tanto, el objetivo de PCA se puede expresar matemáticamente (como:

$$\max_P \mathbb{E} \|Px\|^2 \text{ s.t. } P^T P = I.$$

donde la condición $P^T P = I$ sólo expresa la preferencia por una base ortogonal del sub-espacio de menor dimensionalidad donde proyectamos x .



Relación entre un AE y PCA

- Aplicando el método de los multiplicadores de Lagrange, uno obtiene

$$\begin{aligned}\mathcal{L}(P, \lambda) &= \mathbb{E} \|Px\|^2 - \sum_i \lambda_i (\|p_i\|^2 - \text{cte}) \\ &= \mathbb{E} (Px)^T (Px) - \sum_i \lambda_i (p_i^T p_i - \text{cte}) \\ &= \mathbb{E} \text{tr}(x^T P^T Px) - \text{tr}(\Lambda (P^T P - \text{cte } I)) \\ &= \mathbb{E} \text{tr}(P x x^T P^T) - \text{tr}(\Lambda P^T P - \text{cte } \Lambda) \\ &= \text{tr}(P \Sigma P^T) - \text{tr}(P \Lambda P^T) - \text{cte}' ,\end{aligned}$$

con $\Sigma = \mathbb{E}(xx^T)$ la matriz de covarianza (se puede estimar como $\frac{1}{n} \sum_{\ell} x^{(\ell)} x^{(\ell)T}$)

- La primera derivada de la Lagrangiana viene dada por

$$\partial \mathcal{L} / \partial A = 2P\Sigma - 2P\Lambda$$

- Por lo tanto, las filas de la matriz P deben ser vectores propios de $\Sigma = \mathbb{E}(xx^T)$

$$\begin{aligned}P\Sigma = P\Lambda &\Leftrightarrow \Sigma P^T = \Lambda P^T \\ &\Leftrightarrow \Sigma p_i = \lambda_i p_i \quad \forall i .\end{aligned}$$



Relación entre un AE y PCA

- Si el encoder tiene una sola capa escondida completamente conectada y el decoder reconstruye x usando una capa completamente conectada, el AE queda definido por las siguientes ecuaciones:

$$\begin{aligned} z = h(x) &= \sigma_h(Wx + b) & W \in \mathbb{R}^{k \times d}, b \in \mathbb{R}^k \\ \hat{x} = g(z) &= \sigma_g(Uz + c) & U \in \mathbb{R}^{d \times k}, c \in \mathbb{R}^d \end{aligned}$$

- Ignorando los biases y eligiendo sólo activaciones lineales

$$\hat{x} = g(h(x)) = UWx$$

- ¿A qué converge nuestro AE si se entrena para minimizar el MSE?

$$J = \mathbb{E}_x \|\hat{x} - x\|^2 = \mathbb{E}_x \|UWx - x\|^2$$

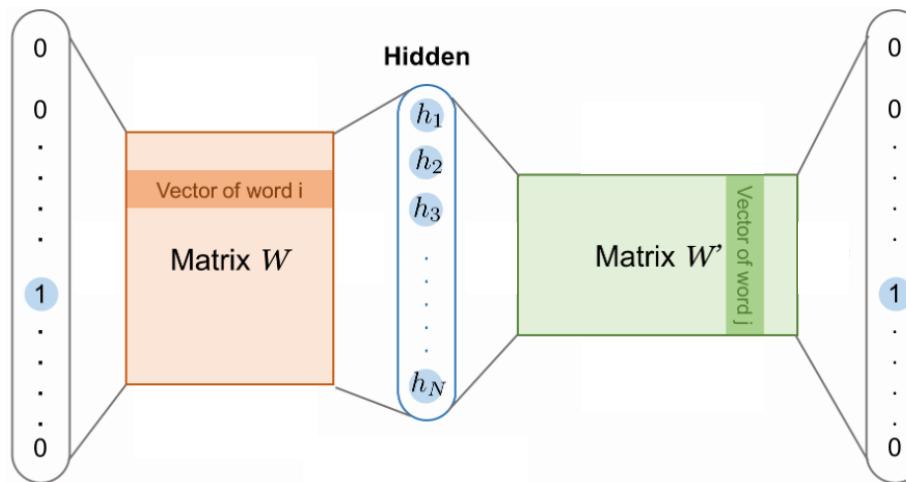
Relación entre un AE y PCA

- Veamos ...
$$\begin{aligned} J &= \mathbb{E}_x \|UWx - x\|^2 = \mathbb{E}_x (UWx - x)^T (UWx - x) \\ &= \mathbb{E}_x (x^T W^T U^T UWx - 2x^T UWx + x^T x) \\ &= \mathbb{E}_x \text{Tr}(x^T W^T U^T UWx - 2x^T UWx) + \text{const} \\ &= \text{Tr}(W^T U^T UW\Sigma - 2UW\Sigma) + \text{const} \end{aligned}$$
- Notemos primero que si optimizamos con respecto a W y recordamos que
$$\frac{\partial \text{tr}(A^T CAB)}{\partial A} = \frac{\partial \text{tr}(ABA^T C)}{\partial A} = CAB + C^T AB^T,$$
obtenemos $\partial J / \partial W = 2U^T UW\Sigma - 2U^T \Sigma$
- Por lo tanto, para un decoder fijo, el encoder óptimo satisface
$$U^T UW\Sigma = U^T \Sigma \quad \Rightarrow \quad W = (U^T U)^{-1} U^T$$
- Si U tiene columnas orto-normales, **el encoder óptimo debe implementar pesos iguales a los del decoder, pero traspuestos:** $W = U^T$.



Relación entre un AE y PCA

- Exactamente lo mismo podemos concluir si invertimos el análisis: dado un encoder fijo (con pesos orto-normales), **el decoder óptimo debe implementar pesos iguales a los del encoder, pero traspuestos:** $U = W^T$.



- Esto es exactamente lo que hace de modo “natural” una arquitectura con pesos no-dirigidos como la RBM. Esto esa también lo que “imponen” muchas arquitecturas auto-supervisadas como word2vec.
- **Como encoder o decoder pueden aprender pesos ortogonales, imponer la compartición de pesos no limita la capacidad del AE de resolver la tarea.**

Relación entre un AE y PCA

- Si ahora examinamos el objetivo como función de sólo 1 matriz (e.g. W)

$$\begin{aligned} J &= \text{Tr} (W^T U^T UW\Sigma - 2UW\Sigma) + \text{const} \\ &= \text{Tr} (W^T WW^T W\Sigma - 2W^T W\Sigma) + \text{const} \end{aligned}$$

- Notemos que si W tiene columnas orto-normales:

$$\begin{aligned} WW^T = I \quad \Rightarrow \quad J &= \text{Tr} (W^T W\Sigma - 2W^T W\Sigma) + \text{const} \\ &= -\text{Tr} (W^T W\Sigma) + \text{const} \\ &= -\mathbb{E}_x (x^T W^T W x) + \text{const} \\ &= -\mathbb{E}_x z^T z + \text{const} \end{aligned}$$

- Por lo tanto, entrenar el AE para minimizar el error de reconstrucción es equivalente a maximizar

$$\max_W \mathbb{E} \|Wx\|^2 \text{ s.t. } W^T W = I .$$

Relación entre un AE y PCA

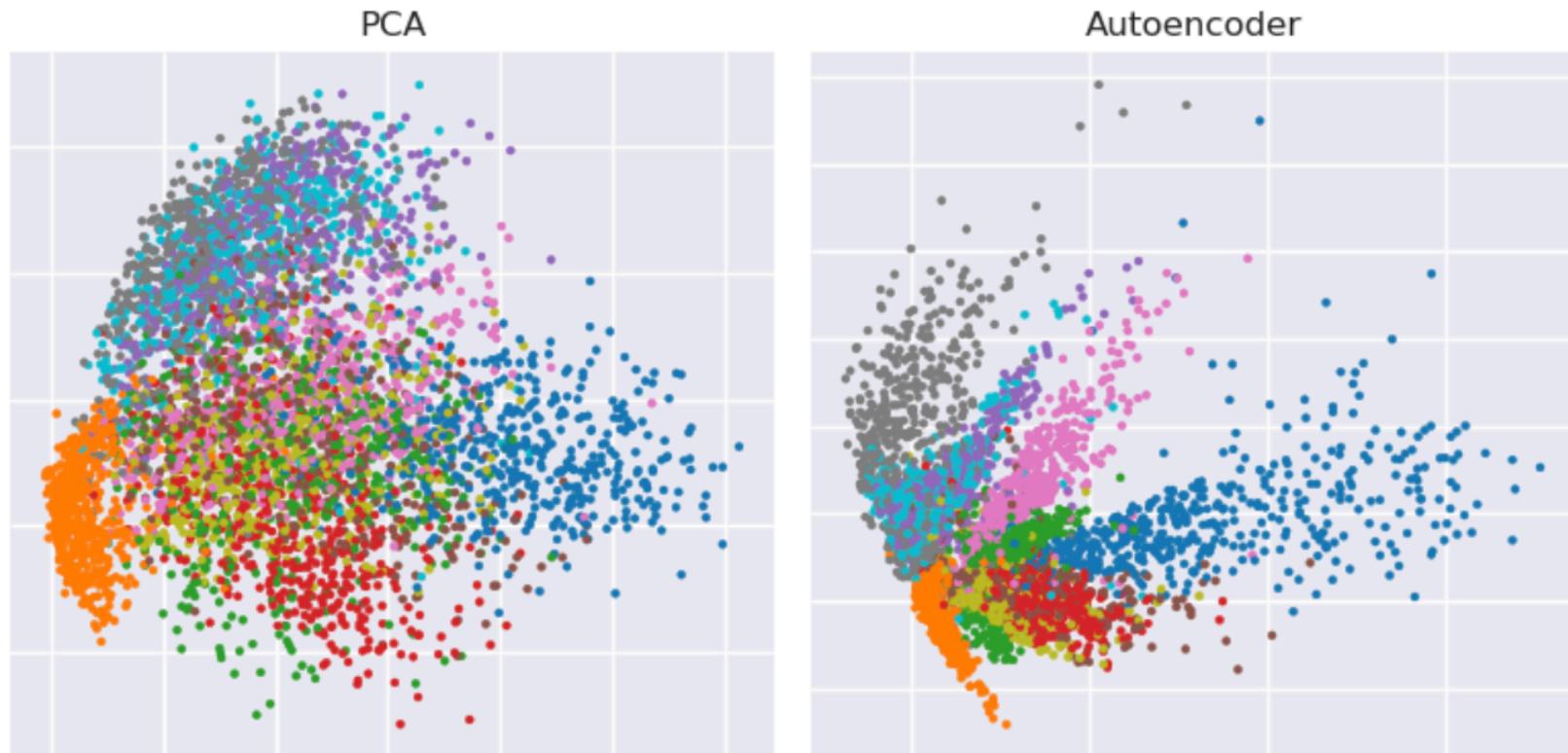
- Comparemos la f.o. del AE con la de PCA

$$\max_W \mathbb{E} \|Wx\|^2 \text{ s.t. } W^T W = I.$$

$$\max_P \mathbb{E} \|Px\|^2 \text{ s.t. } P^T P = I.$$

- **Gran conclusión: Un AE puede perfectamente aprender PCA.** En particular, un AE lineal con pesos compartidos puede implementar PCA.
- Sin embargo, un AE con un mayor número de capas, con funciones de activación no-lineales, y posiblemente con otras funciones objetivos, puede aprender mucho más que PCA. En este sentido **un AE es la generalización profunda, no-lineal, y neuronal de PCA.**

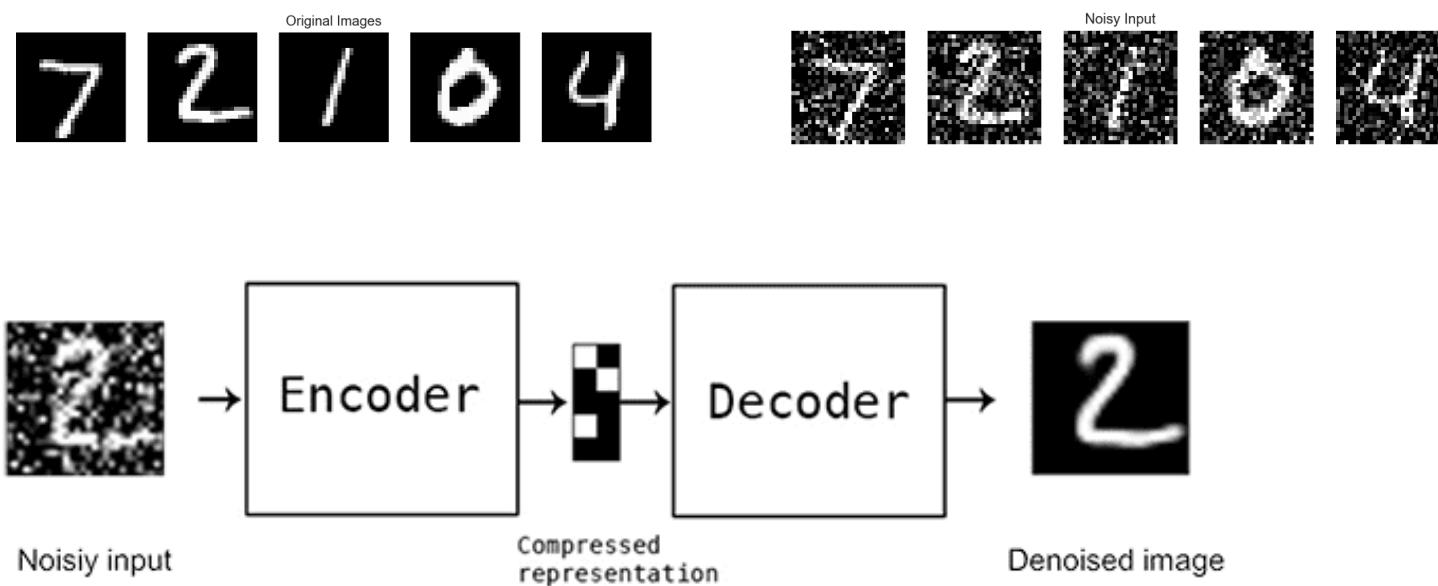
Relación entre un AE y PCA



- Un AE es la generalización profunda, no-lineal, y neuronal de PCA, lo que podría permitirle aprender algo diferente a PCA.

Denoising AutoEncoders

- Otra forma de introducir “presión” sobre el VAE para que aprenda la estructura de los datos consiste en pedirle que reconstruya datos artificialmente corrompidos.



- Proceso clásicos de corrupción: sal y pimienta.

Denoising AutoEncoders

- Miles de papers sequels desde su introducción en 2008.
- Puede verse como un método aproximado para maximizar la información mutua entre la reconstrucción y el dato original (no corrompido).
- Puede verse como un método para detectar las direcciones principales de variación de los datos.

Extracting and Composing Robust Features with Denoising Autoencoders

Pascal Vincent
Hugo Larochelle
Yoshua Bengio
Pierre-Antoine Manzagol

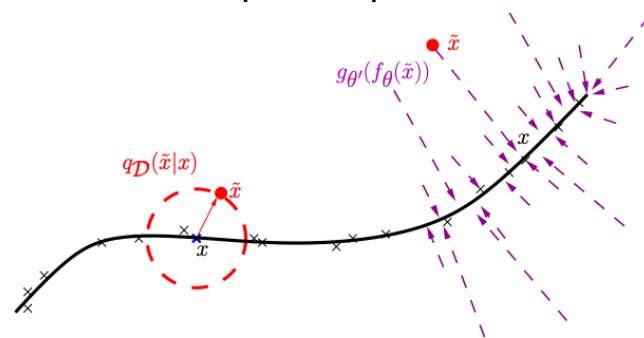
Université de Montréal, Dept. IRO, CP 6128, Succ. Centre-Ville, Montréal, Québec, H3C 3J7, Canada

VINCENTP@IRO.UMONTREAL.CA
LAROCHEH@IRO.UMONTREAL.CA
BENGOY@IRO.UMONTREAL.CA
MANZAGOP@IRO.UMONTREAL.CA

Abstract

Previous work has shown that the difficulties in learning deep generative or discriminative models can be overcome by an initial unsupervised learning step that maps inputs to useful intermediate representations. We introduce and motivate a new training principle for unsupervised learning of a representation based on the idea of making the learned representations robust to partial cor-

to ponder the difficult problem of inference in deep directed graphical models, due to “explaining away”. Also looking back at the history of multi-layer neural networks, their difficult optimization (Bengio et al., 2007; Bengio, 2007) has long prevented reaping the expected benefits of going beyond one or two hidden layers. However this situation has recently changed with the successful approach of (Hinton et al., 2006; Hinton & Salakhutdinov, 2006; Bengio et al., 2007; Ranzato et al., 2007; Lee et al., 2008) for training Deep Belief Networks and stacked autoencoders.



AutoEncoders Regularizados

- Otra forma de introducir “presión” sobre el VAE para que aprenda la estructura de los datos consiste en introducir un regularizador en la función objetivo que pida algo más que “reconstruir”

$$J(\phi, \theta) = \mathbb{E}_x L(g(h(x)), x) + \lambda R(g_\theta, h_\phi)$$

- Por ejemplo, los AE contractivos, fuerzan explícitamente el aprendizaje de una representación robusta, con alto grado de invarianza frente a perturbaciones de los datos

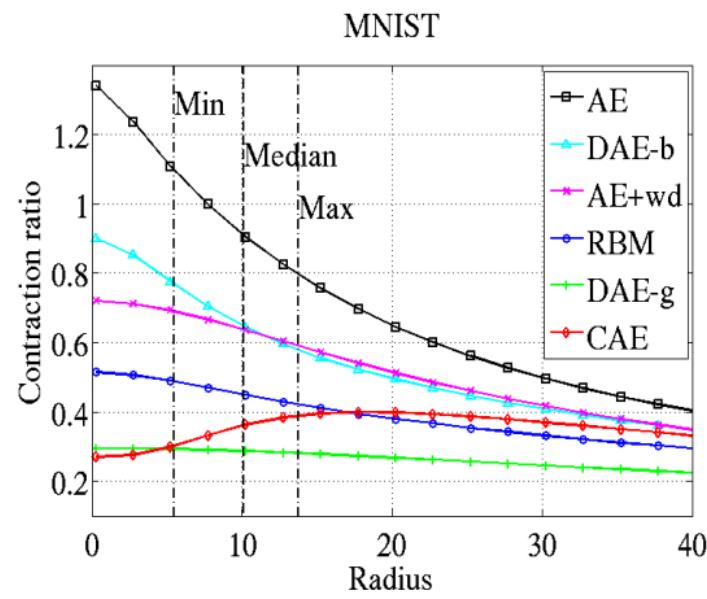
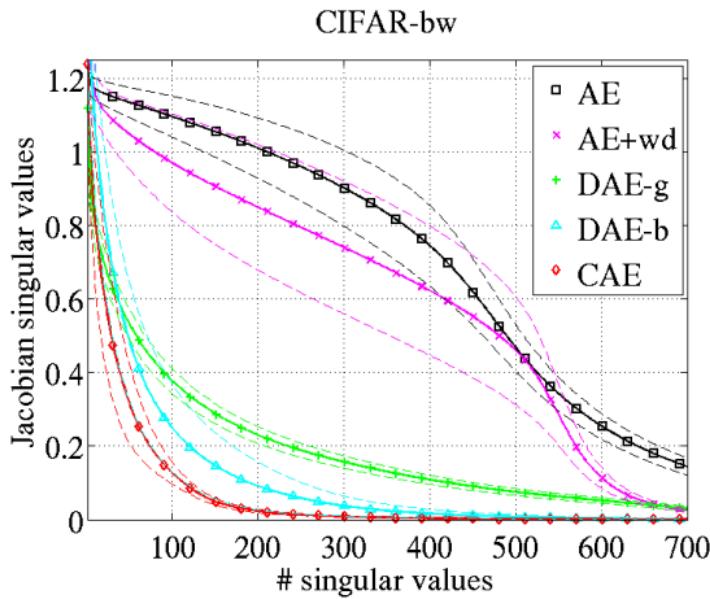
$$R(g_\theta, h_\phi) = \mathbb{E}_x \|\partial h / \partial x\|^2$$



Rifai, S., Vincent, P., Muller, X., Glorot, X., & Bengio, Y. (2011). Contracting auto-encoders. In *International Conference on Machine Learning (ICML)* (p. 95).

AutoEncoders Regularizados

- Izquierda: Valores singulares del Jacobiano (promedio sobre ejemplos). Derecha: Cuociente entre distancias en el espacio original y distancias en el espacio latente.



Rifai, S., Vincent, P., Muller, X., Glorot, X., & Bengio, Y. (2011). Contracting auto-encoders. In *International Conference on Machine Learning (ICML)* (p. 95).

AutoEncoders Contractivos

- Otro ejemplo “famoso” de AE regularizado es el AE disperso (sparse AE).
- La idea es que frente a un determinado input, sólo un subconjunto muy reducido de neuronas latentes se active (esto se puede interpretar la búsqueda de explicaciones sencillas/parsimoniosas de cada dato).

$$R(g_\theta, h_\phi) = \sum_j \text{BCE}(\rho_j \parallel \hat{\rho}_j)$$

con ρ_j el nivel deseado ese actividad para la neurona j (por ejemplo ϵ) y

$$\hat{\rho}_j = \mathbb{E}_x h_j(x)$$

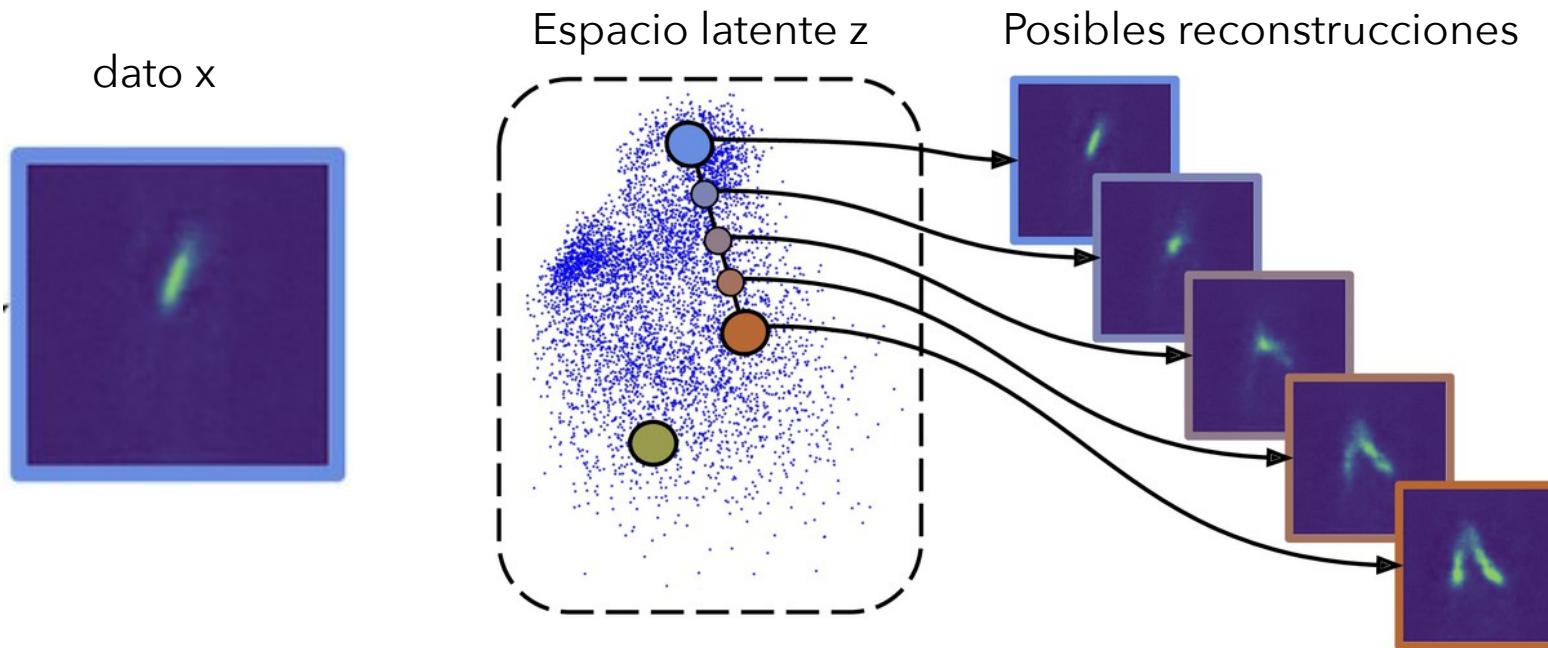
- Se han propuesto muchas versiones alternativas. Por ejemplo Makhzani et al. proponen “apagar” explícitamente las neuronas menos activadas durante un forward pass.



Makhzani, A., & Frey, B. (2013). K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*.

AutoEncoders Variacionales

- **Idea:** ¿Porqué no permitir que el encoder sea aleatorio? ¿Porqué no permitir que el encoder y decoder sean aleatorios?



En este caso $z=h(x)$ es una transformación estocástica con f.d.p. $q(z|x)$.
Análogamente, $x=g(z)$ es una transformación estocástica con f.d.p. $p(x|z)$.
Tal como en una RBM!

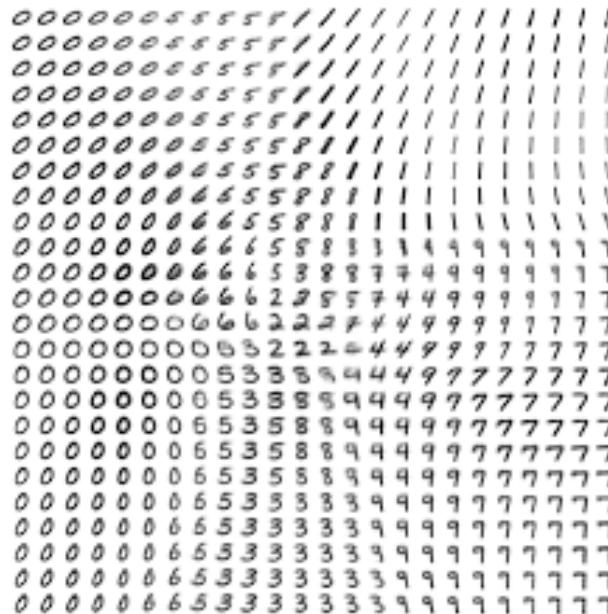
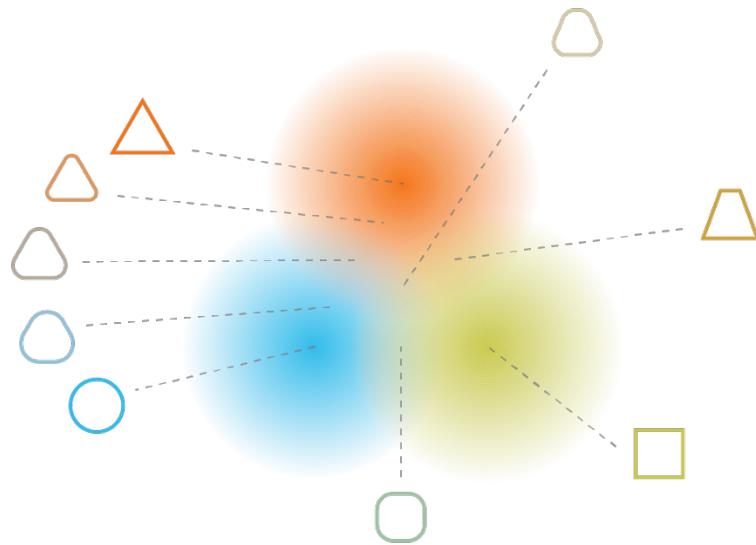
AutoEncoders Variacionales

- Un espacio latente estocástico tiene las siguientes ventajas conceptuales
 - El modelo aprende a estimar la verosimilitud (i.e. la **incerteza**) de sus reconstrucciones $p(x|z)$.
 - El modelo debe aprender a reconstruir x desde representaciones alternativas ligeramente diferentes entre sí lo que mejora su **robustez y capacidad de generalización**.
 - **Interpolación:** Si nos movemos en el espacio latente (por ejemplo, de una moda a otra), los datos generados debiesen cambiar de modo suave y continuo.
 - **Exploración:** Moviéndonos en el espacio latente, podemos manipular la diversidad de los datos generados.



AutoEncoders Variacionales

- Interpolación y Exploración:

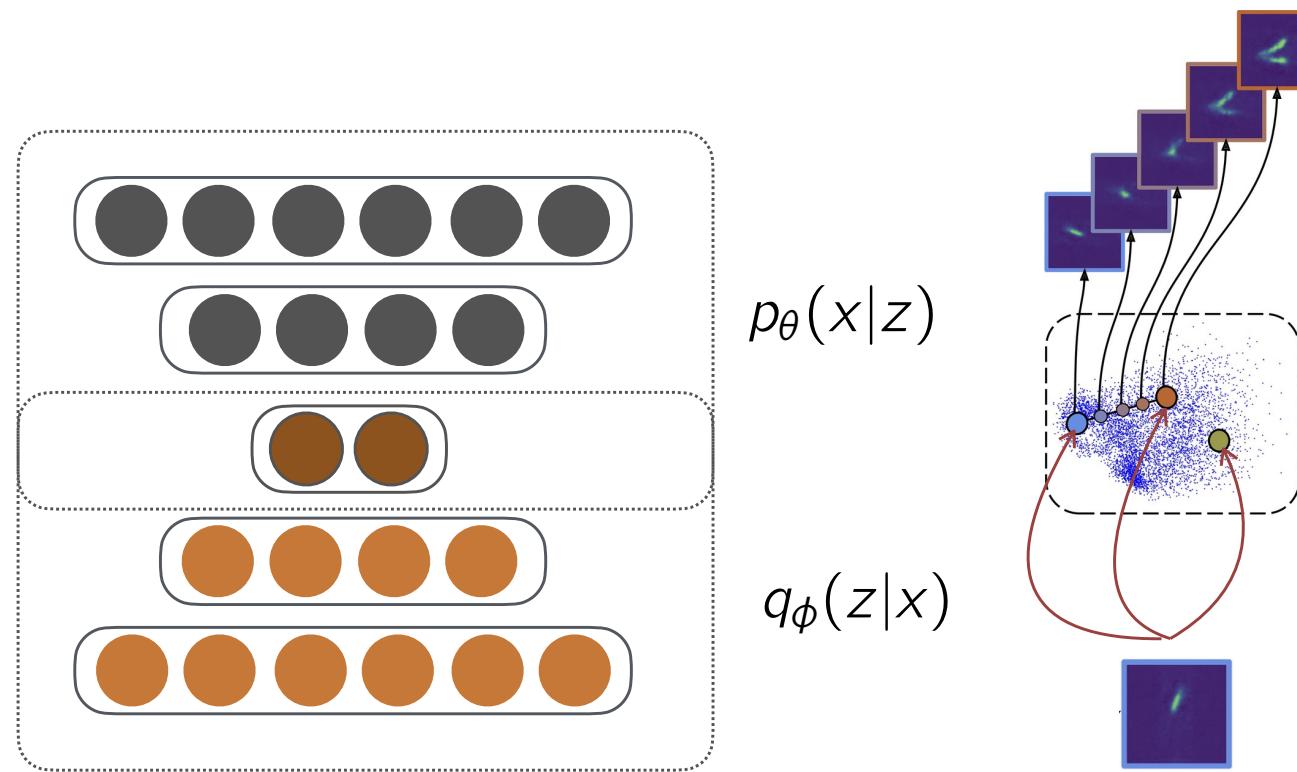


Edit prompt or view more images↓

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., ... & Sutskever, I. (2021, July). Zero-shot text-to-image generation. In *International Conference on Machine Learning* (pp. 8821-8831). PMLR.

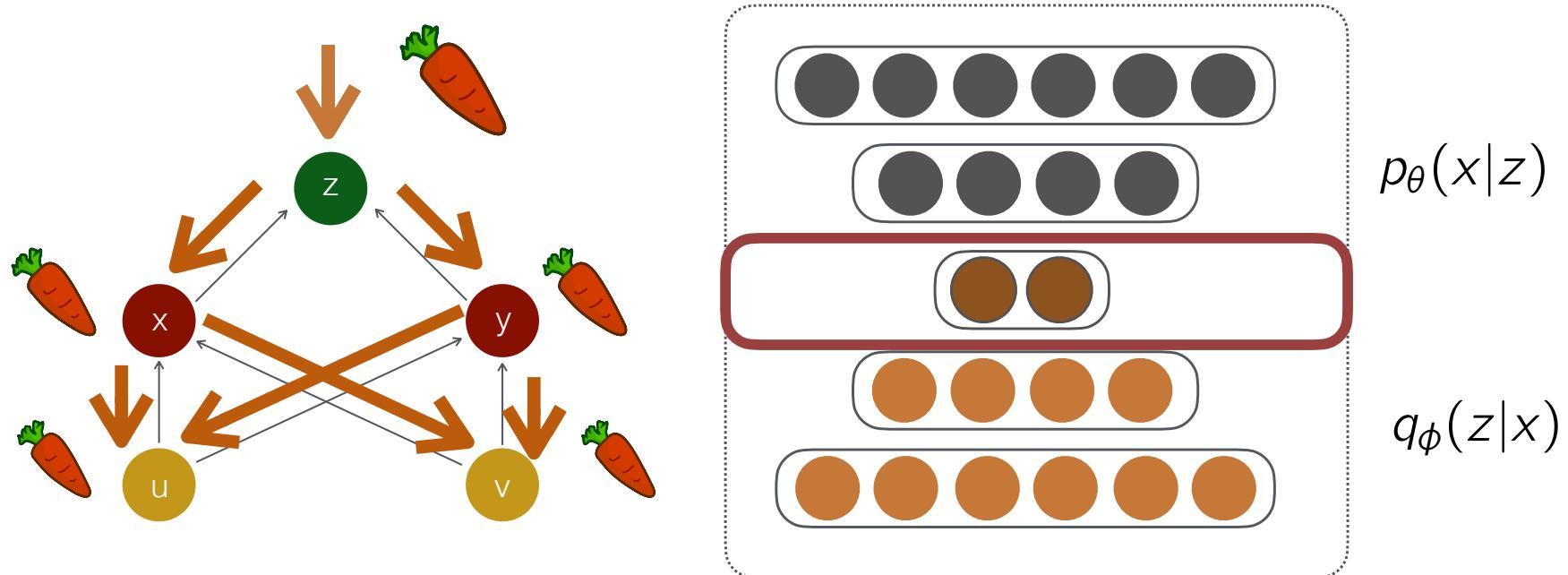
AutoEncoders Variacionales

- Si queremos implementar esta idea, el encoder aprende una distribución $q(z|x)$ y el decoder aprende otra distribución $p(x|z)$.
- ¿Cómo entrenamos la máquina?



BP sobre Capas Estocásticas?

- Como la representación latente es una variable aleatoria. Si aplicamos back-propagation sobre el error de reconstrucción, los gradientes deben poder fluir a través de una **capa estocástica**.



- Hasta 2013/2014 no se sabía cómo hacer esto

BP sobre Capas Estocásticas?

- La solución (D. Kingma, 2013) se conoce como *truco de reparametrización* (*reparametrization trick*) y es extremadamente sencilla.
- Muchas variables aleatorias que depende de parámetros (e.g posición y escala) pueden escribirse como transformaciones deterministas $Z = g_\phi(x, \epsilon)$ de una v.a. auxiliar sin parámetros $\epsilon \sim p(\epsilon)$.
- Por ejemplo si $z \sim \mathcal{N}(\mu, \sigma^2)$, podemos escribir $Z = \sigma \cdot \epsilon + \mu$ con $\epsilon \sim \mathcal{N}(0, 1)$ sin parámetros.



Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

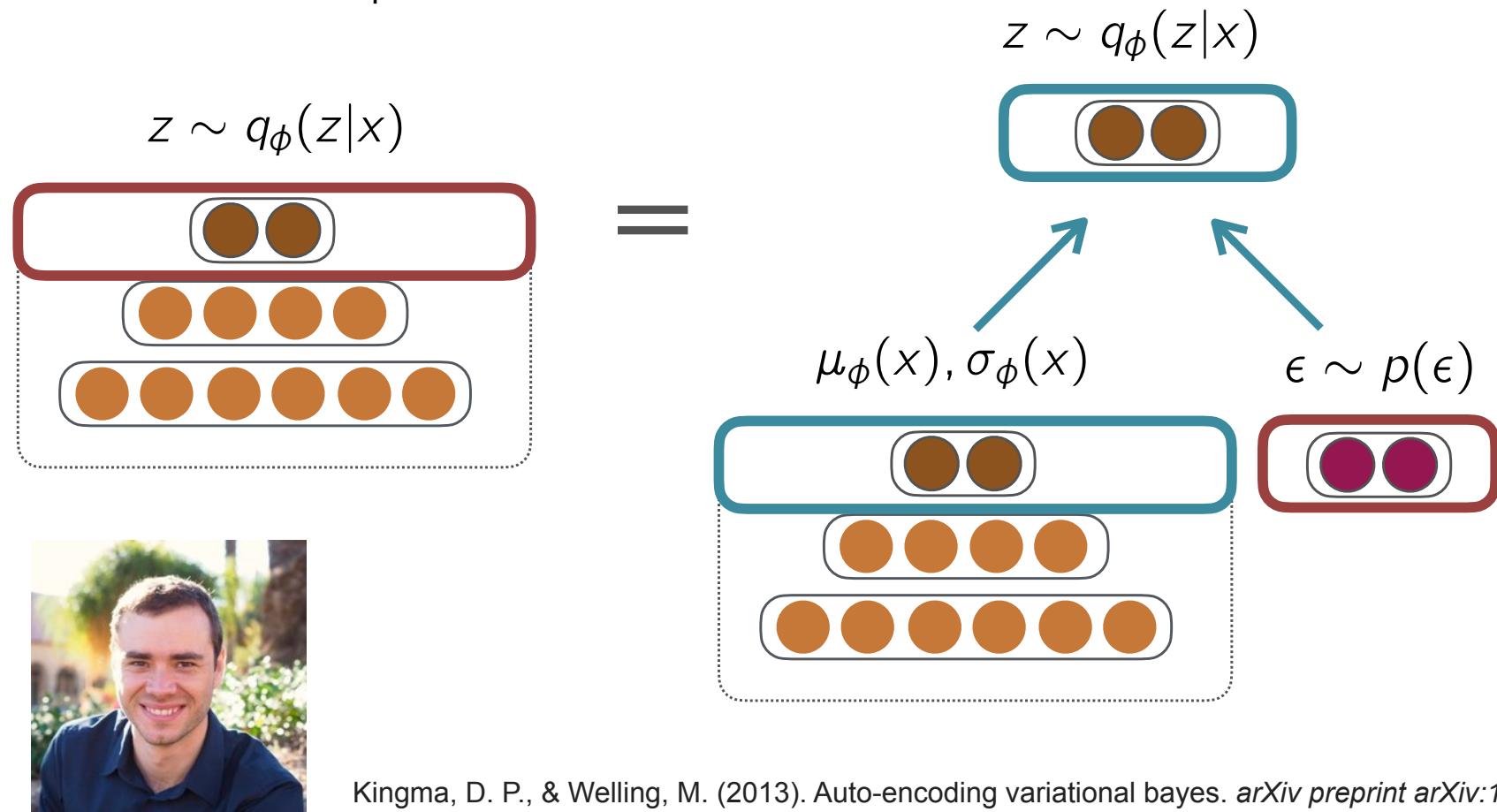
Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions is two-fold. First, we show that a reparameterization of the variational lower bound yields a lower bound estimator that can be straightforwardly optimized using standard stochastic gradient methods. Second, we show that for i.i.d. datasets with continuous latent variables per datapoint, posterior inference can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator. Theoretical advantages are reflected in experimental results.

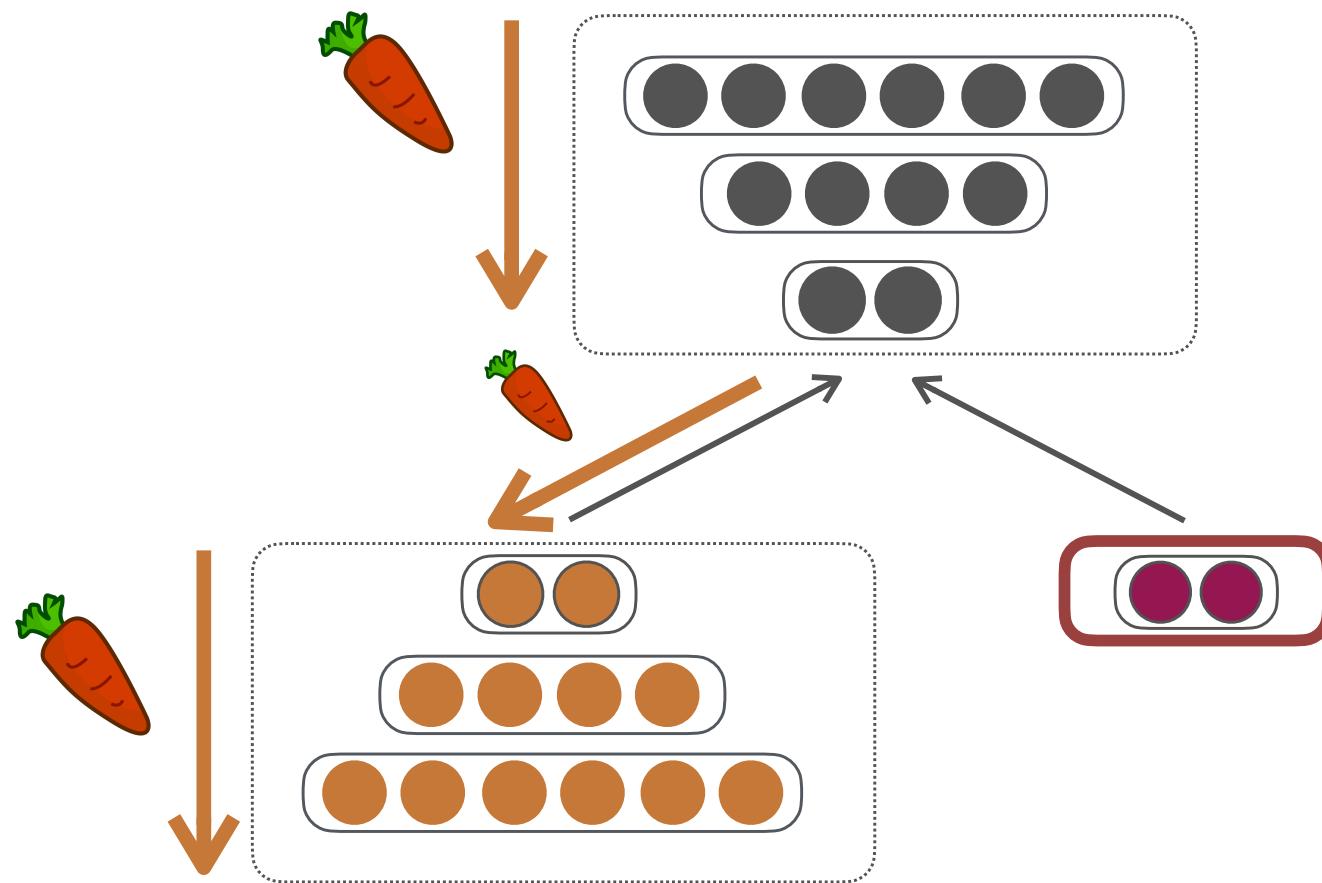
BP sobre Capas Estocásticas?

- Si la distribución que aprende el encoder admite una re-parametrización de este tipo, implementar la transformación aislando los parámetros entrenables de la parte estocástica.



Modelos Generativos

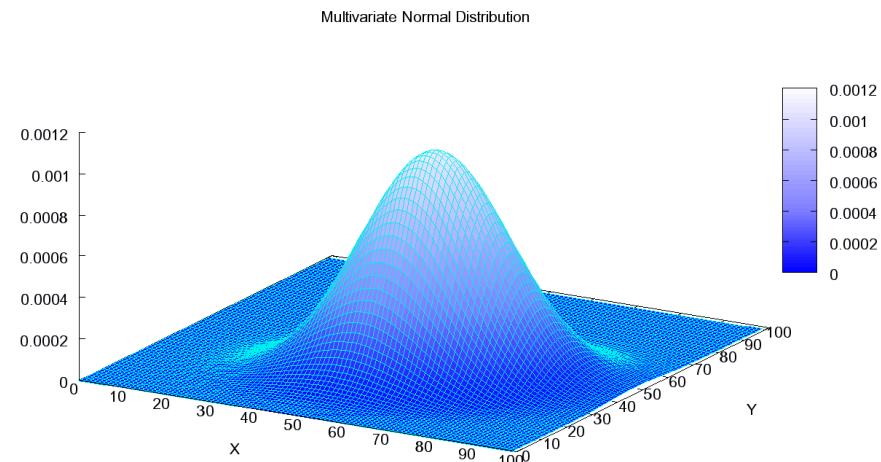
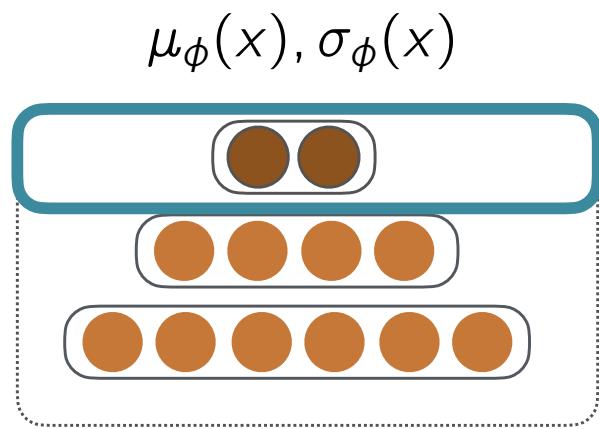
- El efecto es que cuando hacemos back-propagation, el gradiente no necesita atravesar capas estocásticas porque la v.a. auxiliar no tiene parámetros



AutoEncoders Variacionales

- El truco de re-parametrización se puede aplicar a una amplia gama de distribuciones. Sin embargo, un VAE estándar utiliza una gaussiana k-dimensional. En este modelo una red neuronal aprende la media (k-dimensional) y (la diagonal de) la matriz de covarianza.

$$p(z|x) \propto \exp \left(-\sum_{j=1}^k \frac{(z_j - \mu_j(x))^2}{2\sigma_j(x)^2} \right) \quad \Leftrightarrow \quad z|x \sim \mathcal{N}(\mu(x), \text{diag}(\sigma(x)^2))$$



AutoEncoders Variacionales

- Se han estudiado muchas variaciones de esta elección para aumentar la flexibilidad representacional del encoder, inducir la detección de grupos (clusters) en el espacio latente, o para obtener códigos discretos (i.e. índices) de las observaciones.

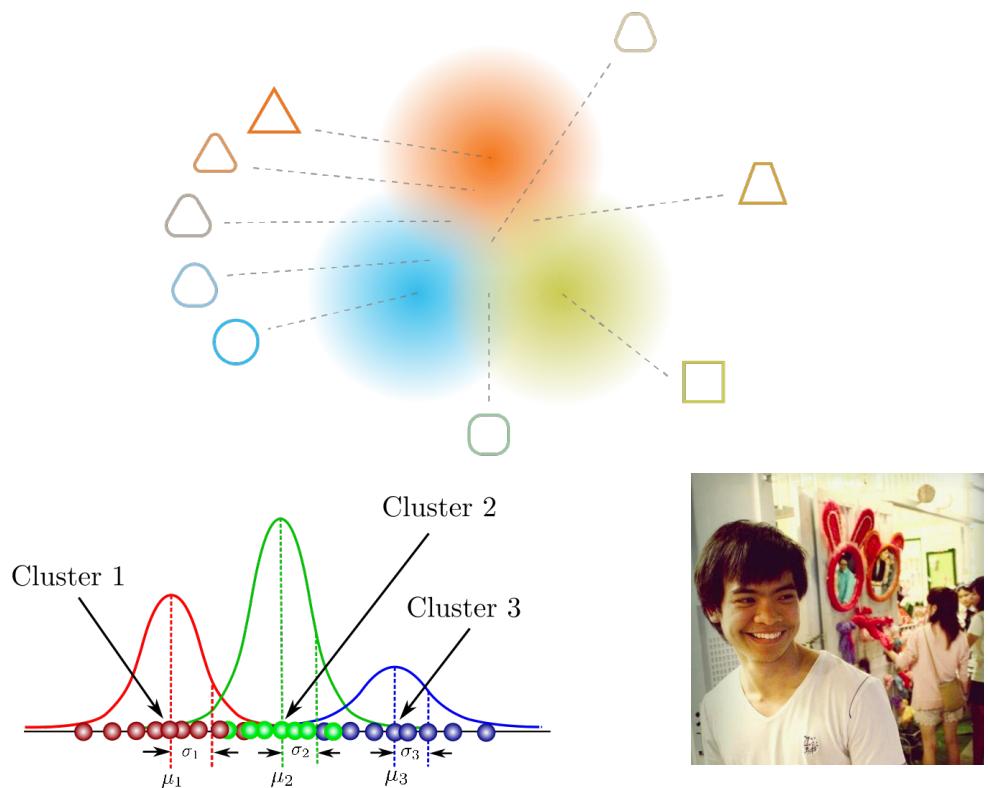
Under review as a conference paper at ICLR 2017

DEEP UNSUPERVISED CLUSTERING WITH GAUSSIAN MIXTURE VARIATIONAL AUTOENCODERS

Nat Dilokthanakul^{1,*}, Pedro A. M. Mediano¹, Marta Garnelo¹,
Matthew C. H. Lee¹, Hugh Salimbeni¹, Kai Arulkumaran² & Murray Shanahan¹
¹Department of Computing, ²Department of Bioengineering
Imperial College London
London, UK
*n.dilokthanakul14@imperial.ac.uk

ABSTRACT

We study a variant of the variational autoencoder model (VAE) with a Gaussian mixture as a prior distribution, with the goal of performing unsupervised clustering through deep generative models. We observe that the known problem of over-regularisation that has been shown to arise in regular VAEs also manifests itself in our model and leads to cluster degeneracy. We show that a heuristic called minimum information constraint that has been shown to mitigate this effect in VAEs can also be applied to improve unsupervised clustering performance with our model. Furthermore we analyse the effect of this heuristic and provide an intuition of the various processes with the help of visualizations. Finally, we demonstrate the performance of our model on synthetic data, MNIST and SVHN, showing that the obtained clusters are distinct, interpretable and result in achieving competitive performance on unsupervised clustering to the state-of-the-art results.

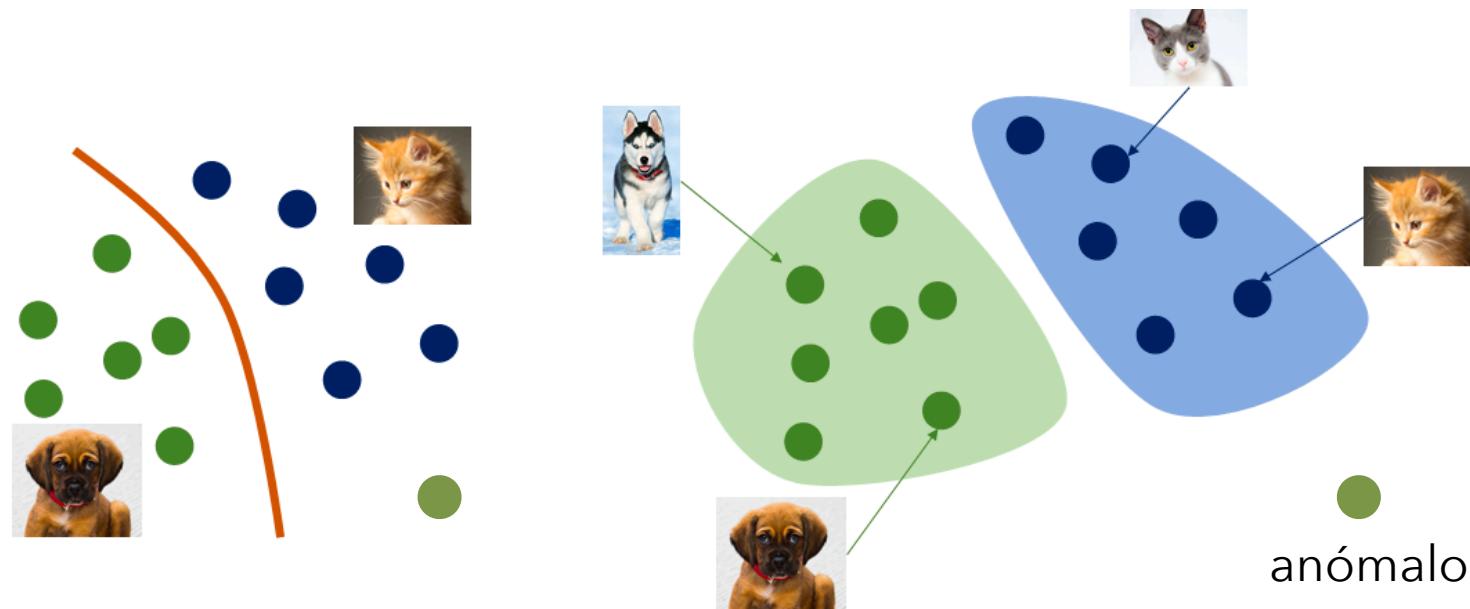


Dilokthanakul, N., Mediano, P. A., Garnelo, M., Lee, M. C., Salimbeni, H., Arulkumaran, K., & Shanahan, M. (2016). Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*.



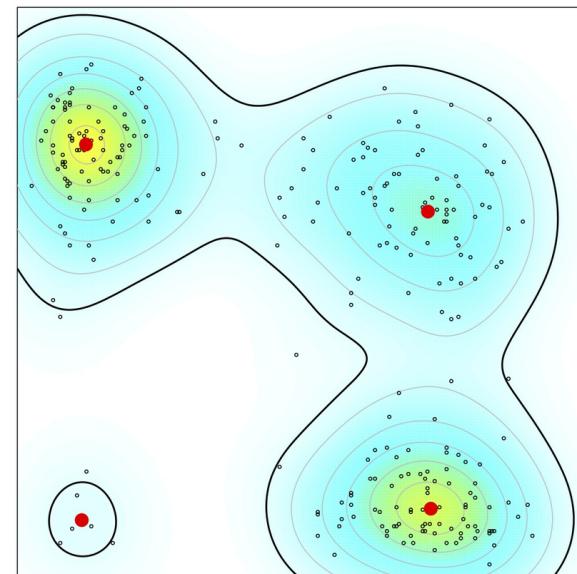
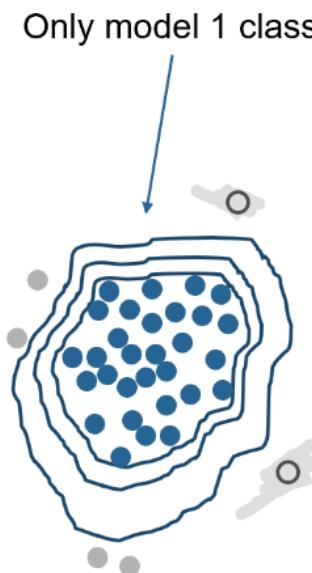
Modelos Generativos

- Ya hemos mencionado que una forma general de abordar el problema de aprendizaje *no-supervisado* consiste en aprender $p(x)$, es decir, construir un modelo probabilístico de los datos.
- Para muchos de hecho, ésta es la forma correcta de aprender desde datos *aún en el caso supervisado* (e.g. clasificadores generativos). En efecto $p(y|x)$ se puede ver como una inferencia desde $p(x,y)$.



Modelos Generativos

- A partir de este modelo nos esperamos por ejemplo poder determinar qué tan probable o inusual es un determinado escenario actual o futuro x . Esto requiere evaluar $p(x)$ o al menos poder comparar $p(x)$ con $p(x^*)$.
- Nos podría interesar también poder determinar regiones enteras del espacio de características dónde es más probable que aparezcan datos. Esto requiere encontrar las modas de $p(x)$.



Modelos Generativos

- A partir de $p(x)$ esperamos también poder hacer **inferencias** sobre x . Por ejemplo determinar $p(x_1 | x_2)$ con $x=(x_1, x_2)$.
- Pero por sobre todo, una de las posibilidades más sugestivas de estos modelos es la posibilidad de generar nuevos datos que se comporten como los que hemos observado pero no sean idénticos a éstos. Para esto es necesario poder muestrear/simular $p(x)$.



Ventajas y Desventajas de una RBM

- Porqué la RBM no nos basta?
- Recordemos primero que una de las claves en la construcción de la RBM fue la introducción de una variable auxiliar h .

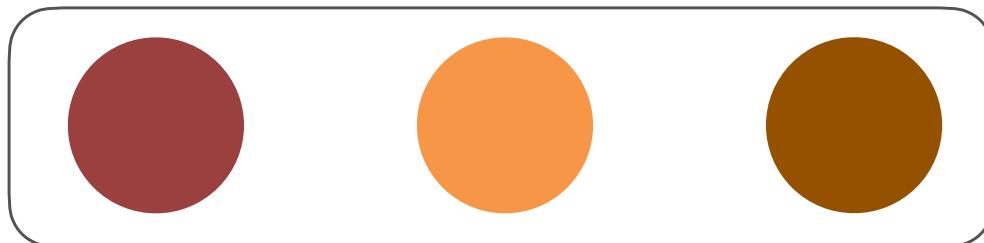
$$p(x, h) = \frac{\exp(-E(x, h))}{Z} \quad E(x, h) = - \sum_{ij} h_j W_{ji} x_i - \sum_j b_j h_j - \sum_i c_i x_i$$

- Esta representación latente h se puede interpretar como una serie de factores (e.g. tópicos) que caracterizan y explican los datos que observamos.



X

h_1 (sports) h_2 (women) h_3 (economy)



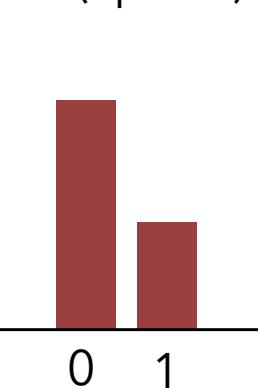
Ventajas y Desventajas de una RBM

- Una de las grandes ventajas de la RBM es que **es muy fácil y eficiente hacer inferencia**: dado x , podemos inferir rápidamente la probabilidad de que los diferentes factores estén presentes en esa observación.

$$p(h|x) = \prod_j p(h_j|x)$$



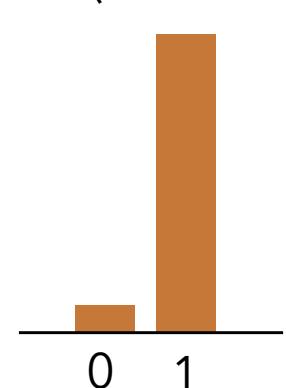
h_1 (sports)



h_2 (women)



h_3 (economy)



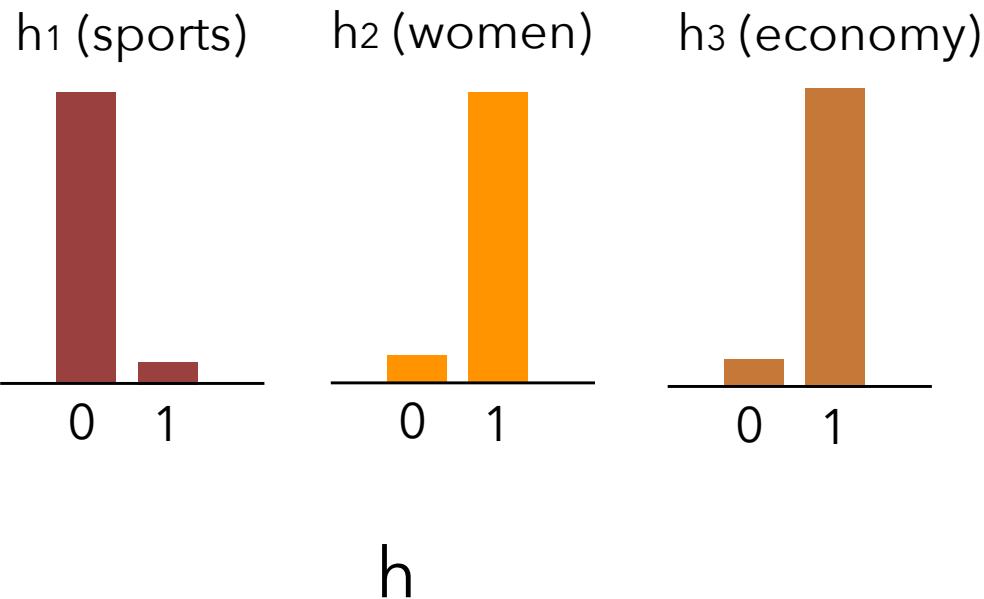
x

$p(h|x)$

Ventajas y Desventajas de una RBM

- Gracias a la reversibilidad de la factorización, es también muy fácil generar un dato (x) dada una combinación deseada de factores/tópicos.

$$p(x|h) = \prod_i p(x_i|h)$$



$p(x|h)$

Ventajas y Desventajas de una RBM

- Lamentablemente también hay desventajas:
 - **La evaluación exacta de $p(x)$ para un dato cualquiera x es computacionalmente prohibitiva** porque requiere estimar la función de partición Z .

$$p(x) = \sum_h p(x, h) = \frac{\exp(-F(x))}{Z} = \frac{\exp(-c^T x)}{Z} \prod_j \exp(\psi_j(x))$$

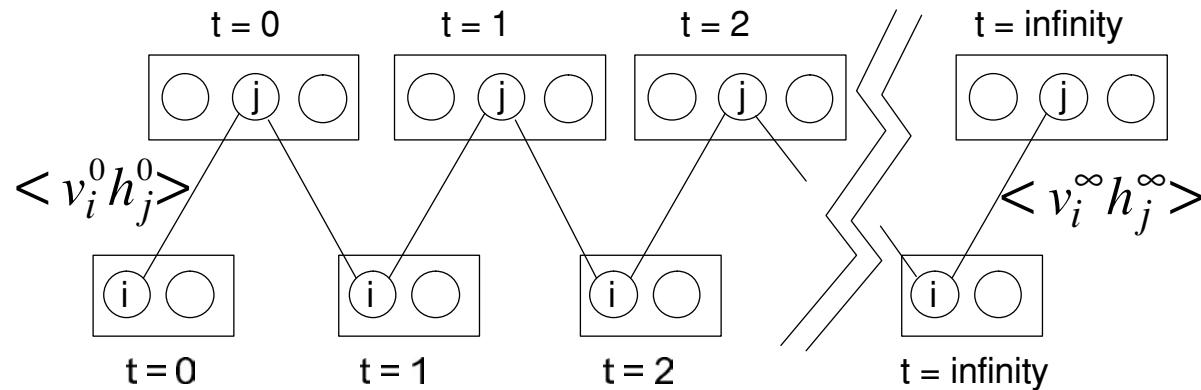
$$Z = \sum_{x,h} p(x, h) = \sum_{x,h} \exp(-E(x, h))$$

Notar que si sólo se requiere la probabilidad un dato relativa a la de otro, la constante Z no es necesaria y por lo tanto esta limitación desaparece. Notar también que el entrenamiento puede escapar al menos en parte a esta limitación ya que el gradiente de $\ln(Z)$ toma la forma de un valor esperado que se estima vía muestreo.



Ventajas y Desventajas de una RBM

- **Simular datos es factible computacionalmente, pero muy lento** ya que se basa iterar el proceso $p(h|x)$, $p(x|h)$ hasta convergencia (muestreo de Gibbs).



Notar que la raíz de esta limitación es que no existe un modelo eficientemente operable de la marginal/prior $p(h)$. El modelo se construyó para que fuese eficiente operar $p(h|x)$ y $p(x|h)$. Notar también que para el entrenamiento un muestreo de Gibbs incompleto (CD1) funciona bien.

Ventajas y Desventajas de una RBM

- **No es fácil extender la RBM** para que sea **más profunda** (tenga más capas latentes) sin perder las propiedades del modelo básico.

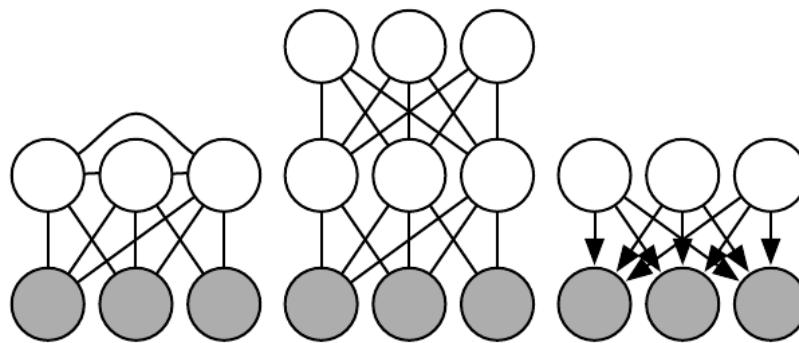
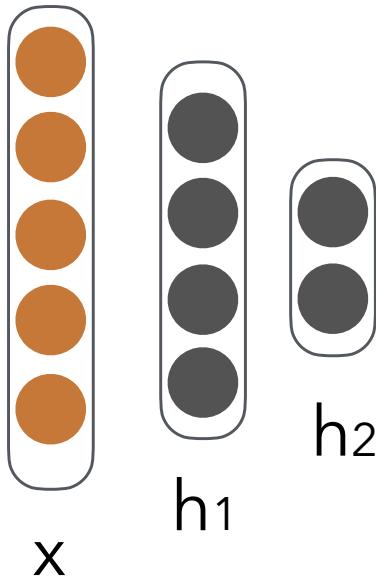
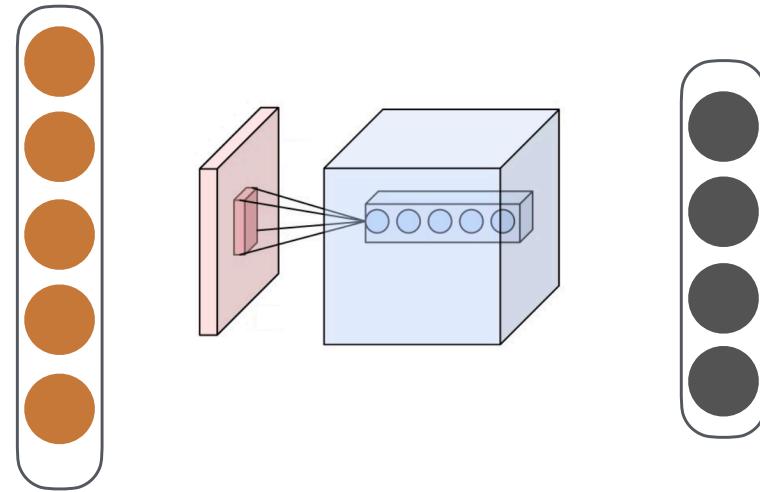


Figure 19.1: Intractable inference problems in deep learning are usually the result of interactions between latent variables in a structured graphical model. These can be due to edges directly connecting one latent variable to another, or due to longer paths that are activated when the child of a V-structure is observed. (Left) A semi-restricted

Concretamente $p(h_1, h_2 | x)$ no factoriza de ningún modo simple y es necesario introducir métodos especializados (e.g. mean field approximations)

Ventajas y Desventajas de una RBM

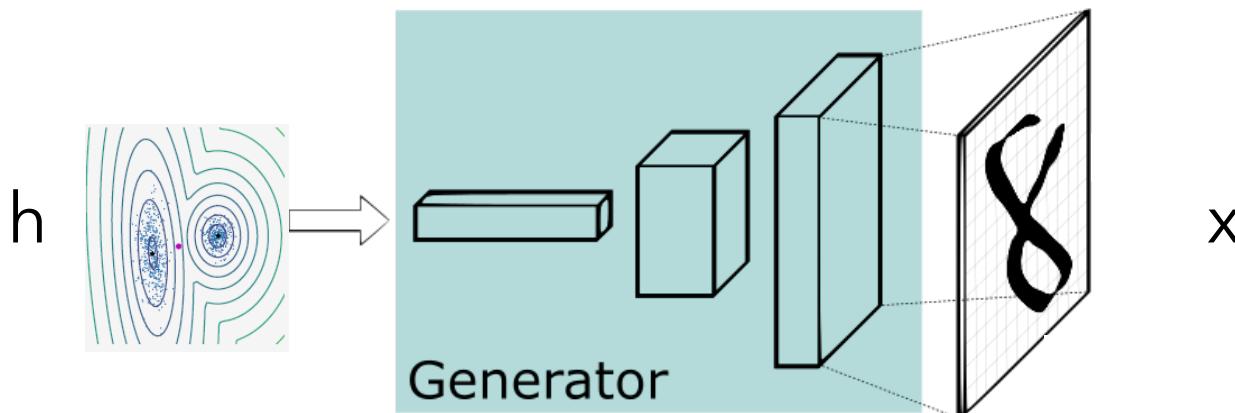
- **No es fácil extender la RBM** para que acomode **cualquier tipo de capa** adicional. Por ejemplo, no es obvio cómo podríamos entrenar el modelo si agregáramos una o más capas convolucionales entre x y h .



Esto se debe a que la RBM no se entrena con back-propagation estándar. Toda nuestra maquinaria de diferenciación automática se pierde y se requería crear un “solver” adhoc. Lamentablemente, agregar capas especializadas es necesario para generar data compleja.

Modelo Generativo Dirigido

- Un modo de **simplificar y flexibilizar el proceso generativo** consiste en aprender una distribución marginal $p(h)$ sobre la variable latente (fácil de muestrear) y representar la transformación $p(x|h)$ como un modelo dirigido que puede aproximar cualquier red neuronal (CNN, RNN).



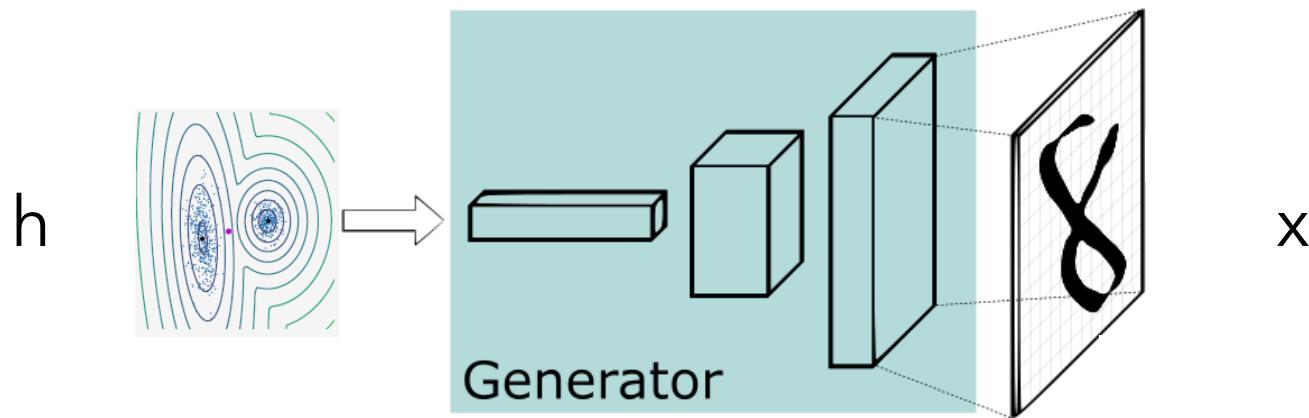
1. Primero se muestrea h desde $p(h)$
2. Luego se opera $p(x|h)$

La red neuronal aprende a transformar la representación oculta en un objeto con la estructura de los observables.

Modelo Generativo Dirigido

- Una gran dificultad de este tipo de modelos es que **la inferencia suele volverse intratable computacionalmente**. Esto ocurre porque como $p(h|x)$ no se modela explícitamente, se debe obtener desde $p(h)$ y $p(x|h)$

$$p(h|x) = \frac{p(x|h)p(h)}{p(x)} = \frac{p(x|h)p(h)}{\sum_h p(x|h)p(h)}$$

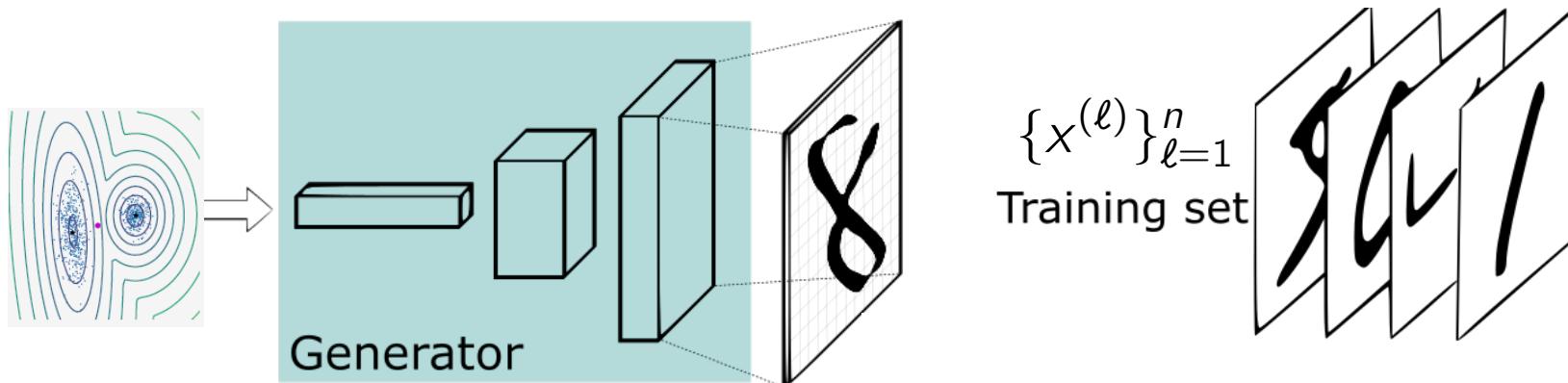


- En palabras, para acelerar y flexibilizar el proceso generativo sacrificamos la capacidad de hacer inferencia (obtener representaciones de los datos). Esto es exactamente lo que sacrificará una GAN.

Modelo Generativo Dirigido

- Para entrenar el modelo generativo, es decir $p(h)$ y $p(x|h)$, usando máxima verosimilitud (MV) sobre un conjunto de ejemplos, necesitamos poder evaluar

$$\mathcal{L} = \mathbb{E}_x \ln p(x) \approx \frac{1}{n} \sum_{\ell} p(x^{(\ell)})$$

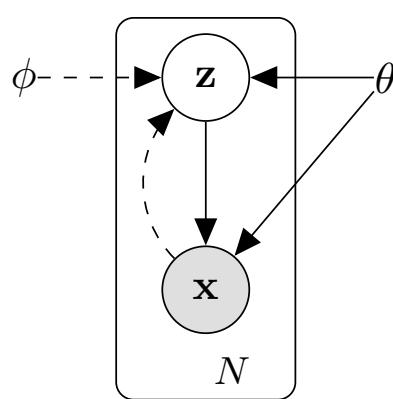


- Lamentablemente, evaluar la verosimilitud de un dato x se vuelve también intratable computacionalmente

$$p(x) = \sum_h p(x|h)p(h)$$

Modelo Generativo Dirigido

En resumen, en un modelo como este:



- Generación rápida y flexible: $h \sim p(h), x \sim p(x|h)$ 😊
- Inferencia intratable: $p(h|x)$ 😕
- Verosimilitud intratable: $p(x)$ 😕
- Entrenamiento intratable (vía MV exacto) 😕

¿Cómo superar estas dificultades?

- Se buscan ideas ...

Generative Adversarial Nets

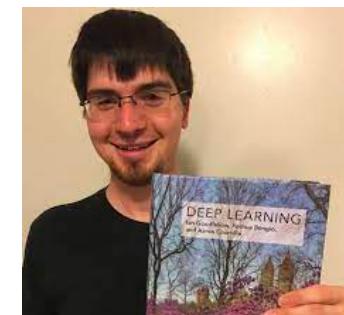
Ian J. Goodfellow, Jean Pouget-Abadie*, Mehdi Mirza, Bing Xu, David Warde-Farley,

Sherjil Ozair,[†] Aaron Courville, Yoshua Bengio[‡]

Département d'informatique et de recherche opérationnelle

Université de Montréal

Montréal, QC H3C 3J7



Ian Goodfellow, et al. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

AutoEncoders Variacionales

- Un auto-encoder variacional (VAE) es un nuevo framework para construir modelos generativos no-supervisados que permite:
 - Inferencia eficiente $p(h|x)$
 - Estimación de $p(x)$ eficiente
 - Entrenamiento eficiente y estable
 - Generación eficiente
 - Flexibilidad arquitectural

Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

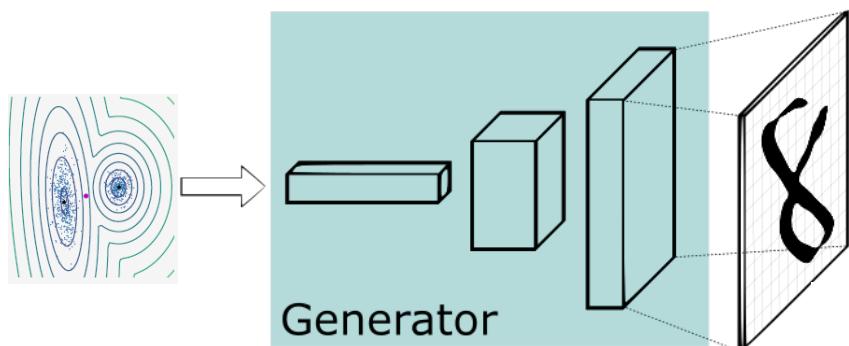
How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions is two-fold. First, we show that a reparameterization of the variational lower bound yields a lower bound estimator that can be straightforwardly optimized using standard stochastic gradient methods. Second, we show that for i.i.d. datasets with continuous latent variables per datapoint, posterior inference can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator. Theoretical advantages are reflected in experimental results.



Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

AutoEncoders Variacionales

- **Warning (notación):** volveremos a denotar la variable latente como z e incluiremos explícitamente los parámetros θ del modelo.
 1. La distribución $p(z)$ puede contener parámetros (para por ejemplo aprender diferentes modas). Escribiremos $p_\theta(z)$.
 2. Como $p(x|z)$ se implementará usando una red neuronal (sin restricciones arquitecturales) esta parte generará muchísimos parámetros entrenables. Escribiremos $p_\theta(x|z)$.
- Por ejemplo, $p_\theta(z)$ podría ser una mezcla de gaussianas y $p_\theta(x|z)$ una Bernoulli multi-dimensional (pixeles en 1=blanco ó 0=negro).



Para obtener las prob. de activación de cada pixel, **basta que la capa de salida de la red tenga neuronas sigmoidales.** El resto de la arquitectura puede ser cualquier cosa.

ELBO

- Como el modelo diseña $p_\theta(z)$ y $p_\theta(x|z)$, tendremos que $p_\theta(z|x)$ será (en general) intratable. La primera idea clave de un VAE es aproximar esta f.d.p. usando un modelo auxiliar $q_\phi(z|x)$.
- Es fácil ver que, para cualquier q_ϕ , la (log-) verosimilitud (o evidencia) de un dato de entrenamiento x se puede escribir como:

$$\begin{aligned}\ln p_\theta(x) &= \ln \sum_z p_\theta(x, z) = \ln \sum_z q_\phi(z|x) \frac{p_\theta(x, z)}{q_\phi(z|x)} \\ &= \ln \mathbb{E}_{q_\phi(z|x)} \frac{p_\theta(x, z)}{q_\phi(z|x)} \geq \mathbb{E}_{q_\phi(z|x)} \ln \frac{p_\theta(x, z)}{q_\phi(z|x)} \\ &\geq \mathbb{E}_{q_\phi(z|x)} \ln \left(\frac{p_\theta(z|x)p_\theta(x)}{q_\phi(z|x)} \right) =: \text{ELBO}_{\phi,\theta}(x)\end{aligned}$$

- La cota anterior se denominada ELBO (*evidence lower bound*) y veremos que puede evaluarse eficientemente.

Gap Variacional

- De hecho ...

$$\begin{aligned}\Delta_{\phi,\theta}(x) &= \ln p_\theta(x) - \text{ELBO}_{\phi,\theta}(x) \\ &= \ln p_\theta(x) - \mathbb{E}_{q_\phi(z|x)} \ln \left(\frac{p_\theta(z|x)p_\theta(x)}{q_\phi(z|x)} \right) \\ &= \ln p_\theta(x) - \mathbb{E}_{q_\phi(z|x)} \ln p_\theta(x) - \mathbb{E}_{q_\phi(z|x)} \ln \left(\frac{p_\theta(z|x)}{q_\phi(z|x)} \right) \\ &= \ln p_\theta(x) - \ln p_\theta(x) + \mathbb{E}_{q_\phi(z|x)} \ln \left(\frac{q_\phi(z|x)}{p_\theta(z|x)} \right)\end{aligned}$$

$$\boxed{\Delta_{\phi,\theta}(x) = \mathbf{KL} (q_\phi(z|x) \parallel p_\theta(z|x))}$$

- La ELBO estima perfectamente la evidencia cuando la aproximación $q_\phi(z|x)$ coincide (c.s.) con el verdadero *a-posteriori* $p_\theta(z|x)$.
- **Idea:** podemos “entrenar” $q_\phi(z|x)$ para minimizar este *gap variacional*.

Componentes de la ELBO

- La ELBO se puede escribir como

$$\begin{aligned}\text{ELBO}_{\phi,\theta}(x) &= \mathbb{E}_{q_\phi(z|x)} \ln \left(\frac{p_\theta(z|x)p_\theta(x)}{q_\phi(z|x)} \right) = \mathbb{E}_{q_\phi(z|x)} \ln \left(\frac{p_\theta(x|z)p_\theta(z)}{q_\phi(z|x)} \right) \\ &= \mathbb{E}_{q_\phi(z|x)} \ln p_\theta(x|z) + \mathbb{E}_{q_\phi(z|x)} \ln \left(\frac{p_\theta(z)}{q_\phi(z|x)} \right)\end{aligned}$$

- Por lo tanto, para maximizar la ELBO (reducir el gap variacional y mejorar la cota), tenemos que **minimizar la siguiente función objetivo**

$$\begin{aligned}J_{\phi,\theta}(x) &= -\mathbb{E}_{q_\phi(z|x)} \ln p_\theta(x|z) - \mathbb{E}_{q_\phi(z|x)} \ln \left(\frac{p_\theta(z)}{q_\phi(z|x)} \right) \\ &= -\mathbb{E}_{q_\phi(z|x)} \ln p_\theta(x|z) + \mathbb{E}_{q_\phi(z|x)} \ln \left(\frac{q_\phi(z|x)}{p_\theta(z)} \right) \\ &= \underbrace{-\mathbb{E}_{q_\phi(z|x)} \ln p_\theta(x|z)}_{J_{\phi,\theta}^{(1)}(x)} + \underbrace{\mathbb{E}_{q_\phi(z|x)} \ln \left(\frac{q_\phi(z|x)}{p_\theta(z)} \right)}_{J_{\phi,\theta}^{(2)}(x)}\end{aligned}$$

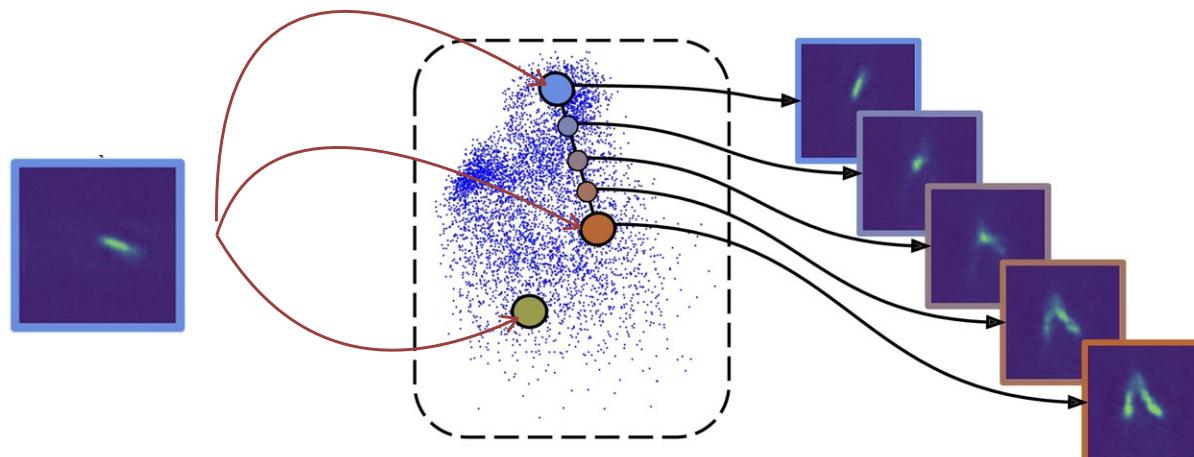


Error de Reconstrucción

- El **primer término** de la función objetivo anterior, se puede interpretar como un **error de reconstrucción**. En efecto, si simulamos $z_1^{(\ell)}, z_2^{(\ell)} \dots z_m^{(\ell)}$ a partir de $q_\phi(z|x^{(\ell)})$, tenemos que

$$J_{\phi,\theta}^{(1)}(x) = -\mathbb{E}_{q_\phi(z|x)} \ln p_\theta(x^{(\ell)}|z) \approx \frac{1}{m} \sum_j -\ln p_\theta(x^{(\ell)}|z_j^{(\ell)})$$

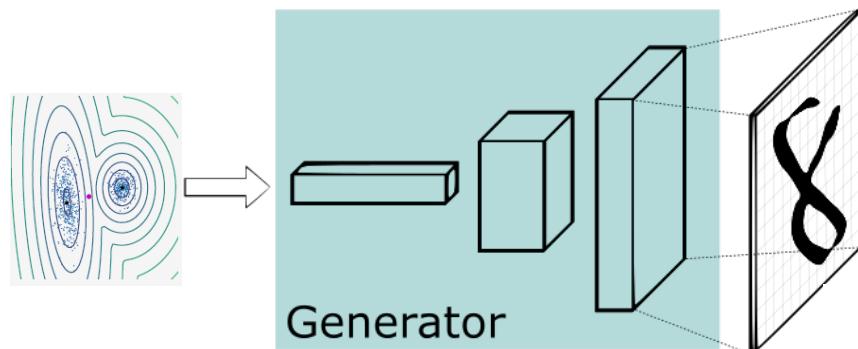
- Le pedimos al modelo que m reconstrucciones del dato (cada una diferente porque depende de una codificación diferente) aproximen bien el dato original.



Error de Reconstrucción

- Para ver que $-\ln p_\theta(x^{(\ell)}|z_j^{(\ell)})$ efectivamente es una **loss**, pensemos que estamos simulando imágenes en blanco y negro (1=blanco, 0=negro) y que implementamos el generador usando una red neuronal con neuronas de salida sigmoidales ...
- En este caso, $p_\theta(x|z)_{ij}$ es la probabilidad de que x_{ij} , el pixel en la posición (i,j) sea blanco (1). Por otro lado, $1 - p_\theta(x|z)_{ij}$ es la probabilidad de que x_{ij} sea negro (0). Por lo tanto,

$$\begin{aligned}-\ln p_\theta(x^{(\ell)}|z_j^{(\ell)}) &= - \sum_{s,t} x_{s,t}^{(\ell)} \ln p_\theta(x_{s,t}^{(\ell)}|z_j^{(\ell)}) \\ &\quad - \sum_{s,t} (1 - x_{s,t}^{(\ell)}) \ln(1 - p_\theta(x_{s,t}^{(\ell)}|z_j^{(\ell)}))\end{aligned}$$



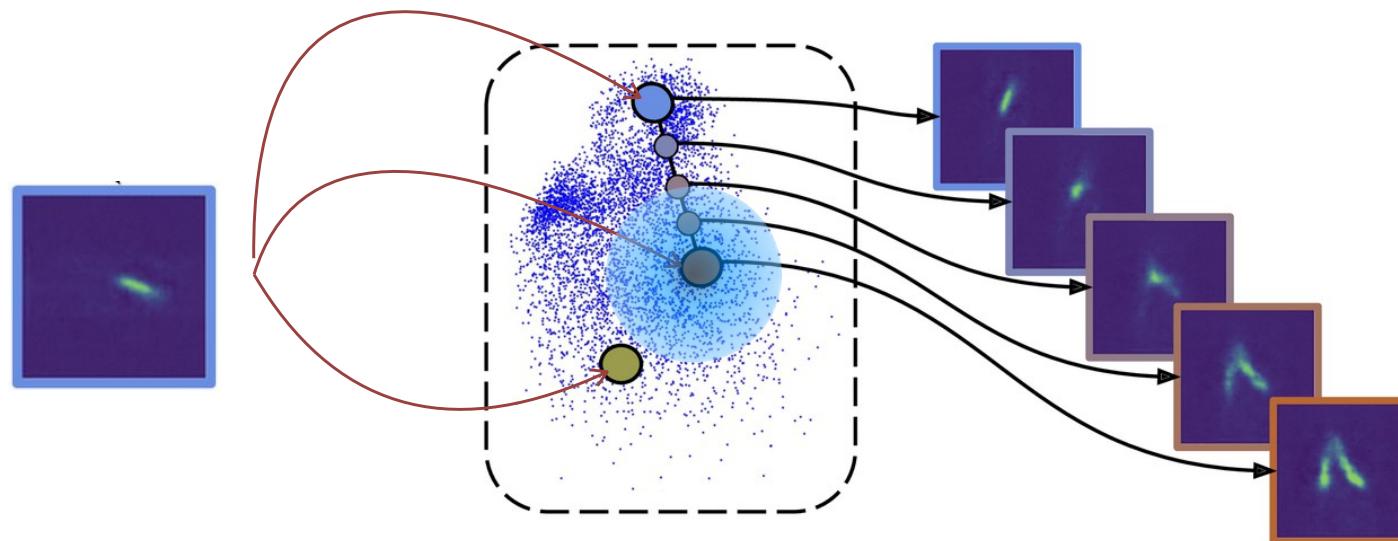
$$-\ln p_\theta(x^{(\ell)}|z_j^{(\ell)}) = \text{BCE}(\hat{x}^{(\ell)}, x^{(\ell)})$$

con $\hat{x}^{(\ell)} = p_\theta(x^{(\ell)}|z_j^{(\ell)})$

Regularizador

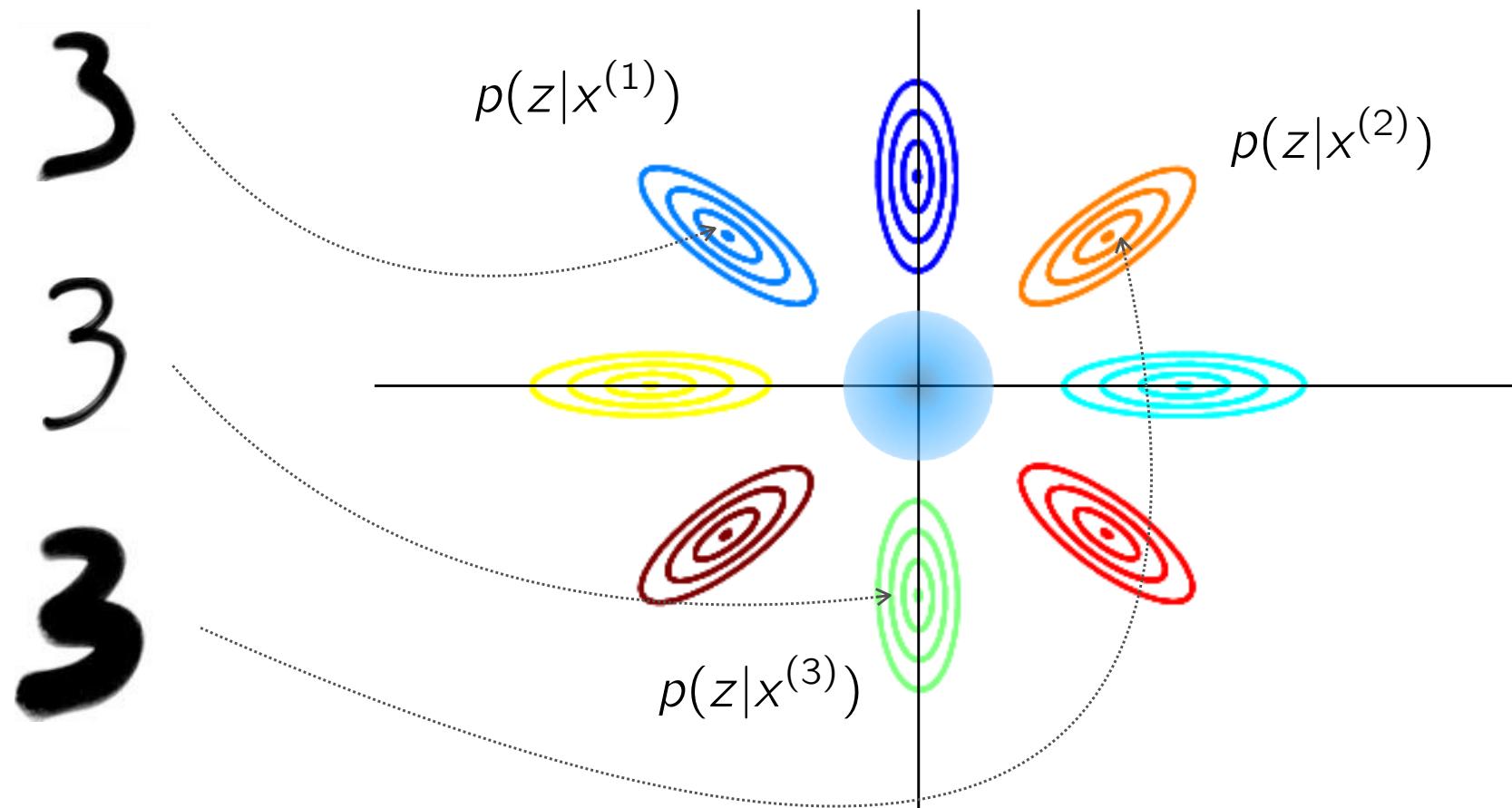
$$J_{\phi,\theta}(x) = \underbrace{-\mathbb{E}_{q_\phi(z|x)} \ln p_\theta(x|z)}_{J_{\phi,\theta}^{(1)}(x)} + \underbrace{KL(q_\phi(z|x) \parallel p_\theta(z))}_{J_{\phi,\theta}^{(2)}(x)}$$

- El segundo término de la función objetivo obtenida, se puede interpretar como un **regularizador** que fuerza al modelo a no codificar excesivamente lejos del prior $p_\theta(z)$, una región donde preferimos representar los datos.



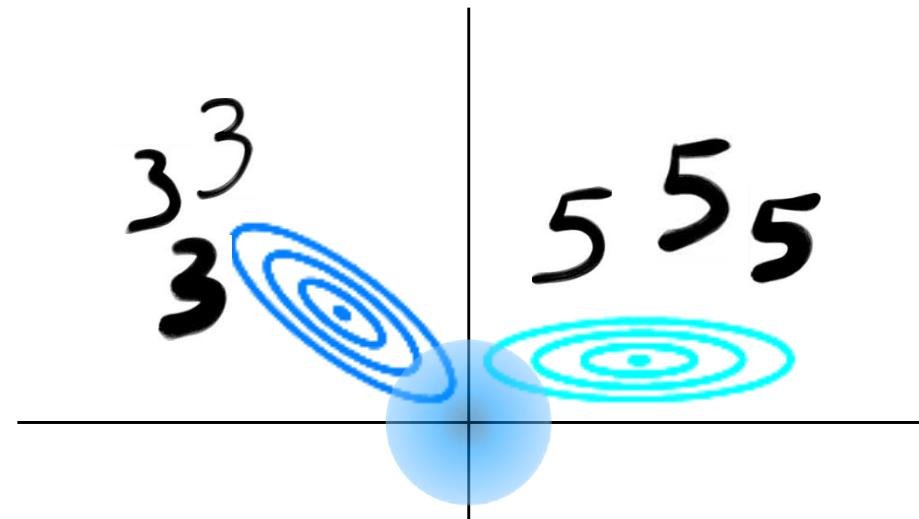
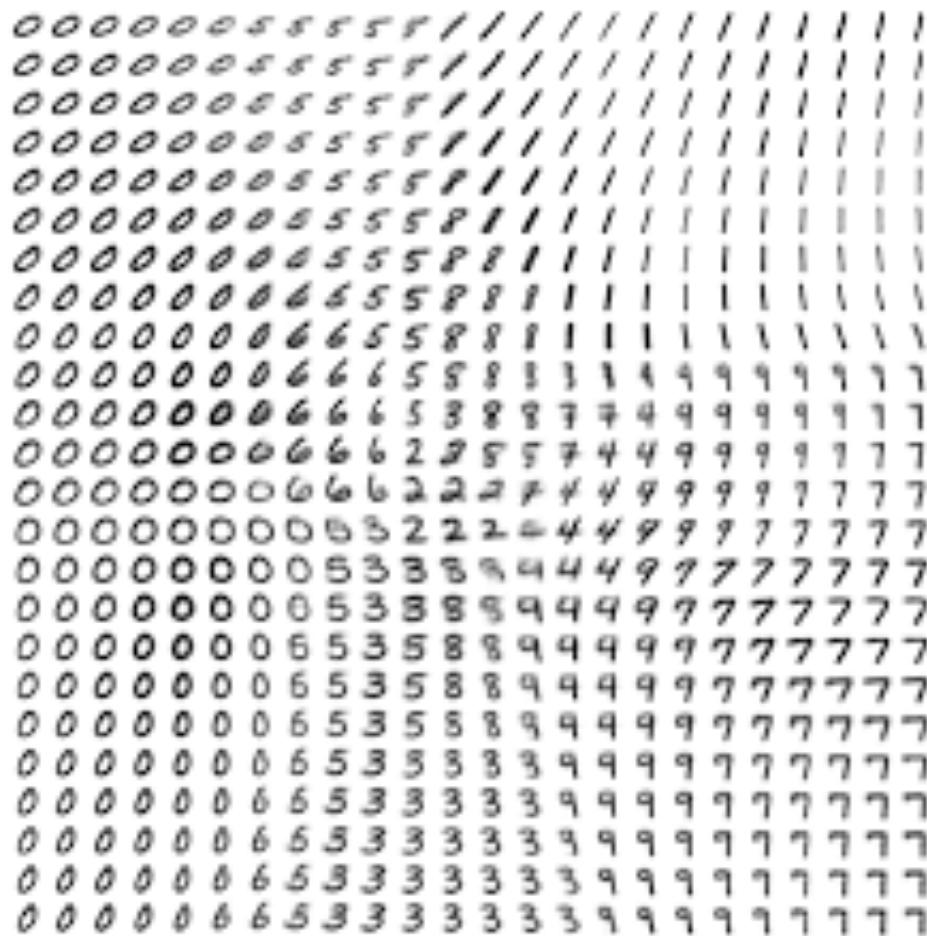
Regularizador

- Sin el regularizador en la función objetivo, el modelo podría memorizar cada dato, mapeándolo a una región propia del espacio latente. El regularizador fuerza una transición más suave de $p(z|x^{(1)})$ a $p(z|x^{(2)})$



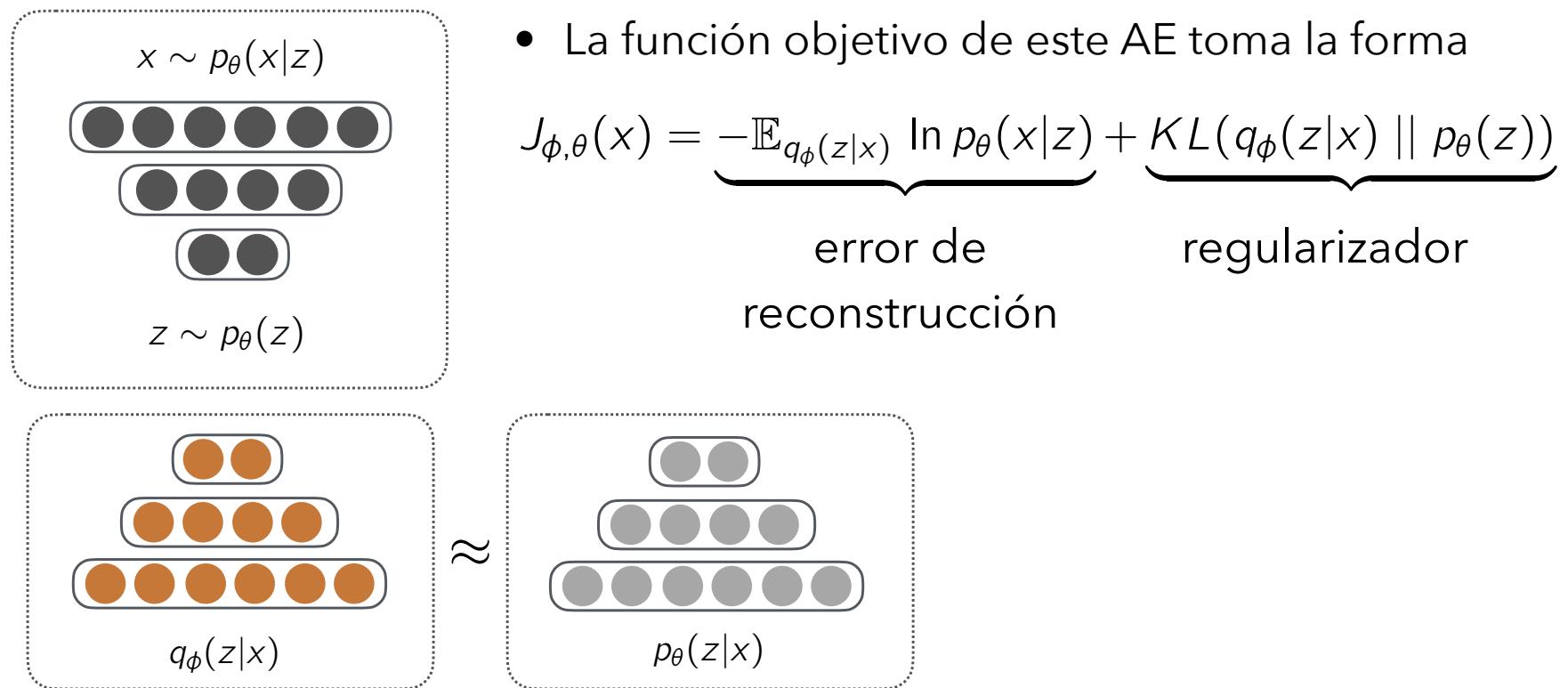
AutoEncoders Variacionales

- El regularizador fuerza una transición más suave de $p(z|x^{(1)})$ a $p(z|x^{(2)})$



AutoEncoders Variacionales

- Como el modelo auxiliar $p(z|x)$ y del modelo generativo $p(z|x)$ se entran de modo simultáneo, el framework anterior se puede interpretar como un AE estocástico donde $p(z|x)$ es el encoder y $p(z|x)$ es el decoder.



Gradientes

- Un requisito fundamental para que el sistema anterior se pueda entrenar mediante backpropagation ordinario es que la f.o.

$$J_{\phi,\theta}(x) = -\mathbb{E}_{q_\phi(z|x)} \ln p_\theta(x|z) + KL(q_\phi(z|x) || p_\theta(z))$$

permita obtener gradientes estables. La segunda contribución clave de un VAE es un método para permitir esto.

- Los gradientes con respecto a θ no son un problema ..

$$J_{\phi,\theta}(x) = \mathbb{E}_{q_\phi(z|x)} G_{\phi,\theta}(x, z)$$

$$\Rightarrow \nabla_\theta J_{\phi,\theta}(x) = \mathbb{E}_{q_\phi(z|x)} \nabla_\theta G_{\phi,\theta}(x, z)$$

$$\Rightarrow \nabla_\theta J_{\phi,\theta}(x) \approx \frac{1}{m} \sum_{j=1}^m \nabla_\theta G_{\phi,\theta}(x, z_j)$$

con $z_1, z_2, \dots, z_m \sim q_\phi(z|x)$



Gradientes

- El problema es el gradiente respecto de ϕ

$$J_{\phi,\theta}(x) = \mathbb{E}_{q_\phi(z|x)} G_{\phi,\theta}(x, z) \Rightarrow \nabla_\phi J_{\phi,\theta}(x) \neq \mathbb{E}_{q_\phi(z|x)} \nabla_\phi G_{\phi,\theta}(x, z)$$

- Hasta 2013/2014, el método convencional era usar el siguiente (score function estimator)

$$\begin{aligned} \nabla_\phi J_{\phi,\theta}(x) &= \nabla_\phi \mathbb{E}_{q_\phi(z|x)} G_{\phi,\theta}(x, z) = \nabla_\phi \int G_{\phi,\theta}(x, z) q_\phi(z|x) dz \\ &= \int \nabla_\phi G_{\phi,\theta}(x, z) q_\phi(z|x) dz + \int G_{\phi,\theta}(x, z) \nabla_\phi q_\phi(z|x) dz \\ &= \mathbb{E}_{q_\phi(z|x)} \nabla_\phi G_{\phi,\theta}(x, z) + \int q_\phi(z|x) G_{\phi,\theta}(x, z) \nabla_\phi \ln q_\phi(z|x) dz \end{aligned}$$

$$\therefore \nabla_\phi J_{\phi,\theta}(x) = \mathbb{E}_{q_\phi(z|x)} \nabla_\phi G_{\phi,\theta}(x, z) + \mathbb{E}_{q_\phi(z|x)} G_{\phi,\theta}(x, z) \nabla_\phi \ln q_\phi(z|x)$$

Gradientes

- Score function estimator

$$\nabla_{\phi} J_{\phi, \theta}(x) = \mathbb{E}_{q_{\phi}(z|x)} \nabla_{\phi} G_{\phi, \theta}(x, z) + \mathbb{E}_{q_{\phi}(z|x)} G_{\phi, \theta}(x, z) \nabla_{\phi} \ln q_{\phi}(z|x)$$

$$\nabla_{\phi} J_{\phi, \theta}(x) \approx \frac{1}{m} \sum_{j=1}^m \nabla_{\phi} G_{\phi, \theta}(x, z_j) + \frac{1}{m} \sum_{j=1}^m G_{\phi, \theta}(x, z_j) \nabla_{\phi} \ln q_{\phi}(z_j|x)$$

Este estimador puede exhibir alta varianza y generar entrenamientos muy inestables. Lo que hace un VAE para resolver esto es una reparametrización de $q(z|x)$. Concretamente, la idea es escribir z como transformación determinista $Z = g_{\phi}(x, \epsilon)$ de una v.a. auxiliar sin parámetros $\epsilon \sim p(\epsilon)$

- Por ejemplo si $z \sim \mathcal{N}(\mu, \sigma^2)$, podemos escribir $Z = \sigma \cdot \epsilon + \mu$ con $\epsilon \sim \mathcal{N}(0, 1)$ sin parámetros.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.



Gradientes

- Con este truco

$$\begin{aligned}\nabla_{\phi} J_{\phi, \theta}(x) &= \nabla_{\phi} \mathbb{E}_{q_{\phi}(z|x)} G_{\phi, \theta}(x, z) \\ &= \nabla_{\phi} \mathbb{E}_{p(\epsilon)} G_{\phi, \theta}(x, z(\epsilon)) \\ &= \mathbb{E}_{p(\epsilon)} \nabla_{\phi} G_{\phi, \theta}(x, z(\epsilon))\end{aligned}$$

Por lo tanto

$$\mathbb{E}_{p(\epsilon)} \nabla_{\phi} G_{\phi, \theta}(x, z(\epsilon)) \approx \frac{1}{m} \sum_{j=1}^m \nabla_{\theta} G_{\phi, \theta}(x, z(\epsilon_j))$$

con $\epsilon_1, \epsilon_2, \dots, \epsilon_m \sim p(\epsilon)$

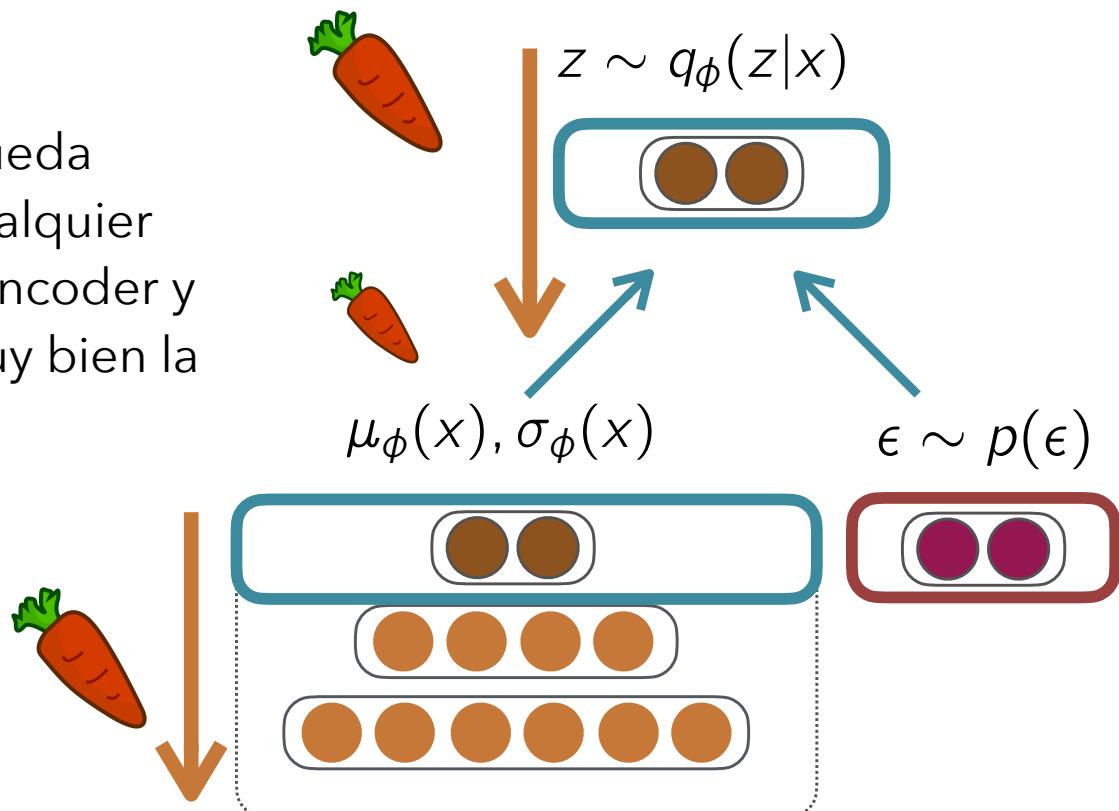
Si el gradiente se estima en un mini-batch relativamente grande (digamos $B=100$ ejemplos), los experimentos sugieren que basta usar $m=1$.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.



Flexibilidad de un VAE

- El truco de reparametrización permite que el gradiente “pase” por la capa estocástica y que el sistema completo se pueda entrenar vía back-propagation.
- Esto hace que un VAE pueda utilizar esencialmente cualquier arquitectura tanto para encoder y decoder (basta elegir muy bien la distribución $q(z|x)$).



Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Demos

The screenshot shows a blog post titled "Building Autoencoders in Keras". The post is dated Sat 14 May 2016 and was written by Francois Chollet. It covers various types of autoencoders including simple, sparse, deep fully-connected, deep convolutional, and variational autoencoders. A note at the bottom states that code examples have been updated to the Keras 2.0 API on March 14, 2017, requiring Keras version 2.0.0 or higher.

The Keras Blog

Keras is a Deep Learning library for Python, that is simple, modular, and extensible.

Archives Github Documentation Google Group

Building Autoencoders in Keras

This post was written in early 2016. It is therefore badly outdated.

In this tutorial, we will answer some common questions about autoencoders, and we will cover code examples of the following models:

- a simple autoencoder based on a fully-connected layer
- a sparse autoencoder
- a deep fully-connected autoencoder
- a deep convolutional autoencoder
- an image denoising model
- a sequence-to-sequence autoencoder
- a variational autoencoder

Sat 14 May 2016
By [Francois Chollet](#)
In [Tutorials](#).

Note: all code examples have been updated to the Keras 2.0 API on March 14, 2017. You will need Keras version 2.0.0 or higher to run them.

<https://blog.keras.io/building-autoencoders-in-keras.html>

<https://machinelearningmastery.com/autoencoder-for-classification/>

