Journal Club 28/11/17

Joseph Boyd JOSEPH.BOYD@MINES-PARISTECH.FR

May 11, 2018

Journal of Machine Learning Research 17 (2016) 1-35 Submitted 5/15; Published 4/16

Domain-Adversarial Training of Neural Networks

Yaroslav Ganin GANIN@SKOLTECH.RU
Evgeniya Ustinova EVGENIYA.USTINOVA@SKOLTECH.RU
Skolkovo Institute of Science and Technology (Skoltech)
Skolkovo, Moscow Region, Russia

Hana Ajakan HANA.AJAKAN.1@ULAVAL.CA
Pascal Germain PASCAL.GERMAIN@IFT.ULAVAL.CA
Département d'informatique et de génie logiciel, Université Laval
Québec, Canada, G1V 0A6

Hugo Larochelle HUGO.LAROCHELLE@USHERBROOKE.CA
Département d'informatique, Université de Sherbrooke
Québec, Canada, J1K 2R1

François Laviolette FRANCOIS.LAVIOLETTE@IFT.ULAVAL.CA
Mario Marchand MARIO.MARCHAND@IFT.ULAVAL.CA
Département d'informatique et de génie logiciel, Université Laval
Québec, Canada, G1V 0A6

Victor Lempitsky LEMPITSKY@SKOLTECH.RU
Skolkovo Institute of Science and Technology (Skoltech)
Skolkovo, Moscow Region, Russia

Two papers in one...

Combines work done in:

Two papers in one...

Combines work done in:

- Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., & Marchand, M. (2014). Domain-adversarial neural networks. arXiv preprint arXiv:1412.4446.

(original, theoretical groundings, shallow network only)

Two papers in one...

Combines work done in:

- Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., & Marchand, M. (2014). Domain-adversarial neural networks. arXiv preprint arXiv:1412.4446.

(original, theoretical groundings, shallow network only)

- Ganin, Y., & Lempitsky, V. (2015, June). Unsupervised domain adaptation by backpropagation. In International Conference on Machine Learning (pp. 1180-1189).

(extension to deep networks, amenability to implementation, state-of-the-art results)

Unsupervised domain adaptation

- Unsupervised domain adaptation from source S to target T:

$$S = \{(\mathbf{x}_i, y_i)\} \sim \mathcal{D}_S, T = \{\mathbf{x}_i\} \sim \mathcal{D}_T$$

Unsupervised domain adaptation

- Unsupervised domain adaptation from source S to target T:

$$S = \{(\mathbf{x}_i, y_i)\} \sim \mathcal{D}_S, T = \{\mathbf{x}_i\} \sim \mathcal{D}_T$$

- The target is *unlabelled* (unsupervised), or partially labelled (semi-supervised), but implicitly has the same class ontology.

Unsupervised domain adaptation

- Unsupervised domain adaptation from source S to target T:

$$S = \{(\mathbf{x}_i, y_i)\} \sim \mathcal{D}_S, T = \{\mathbf{x}_i\} \sim \mathcal{D}_T$$

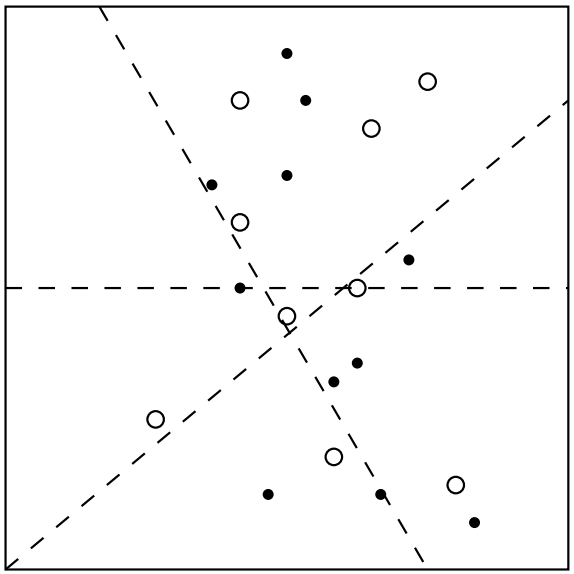
- The target is *unlabelled* (unsupervised), or partially labelled (semi-supervised), but implicitly has the same class ontology.

- Aim: create a classifier that performs well on both S and T.

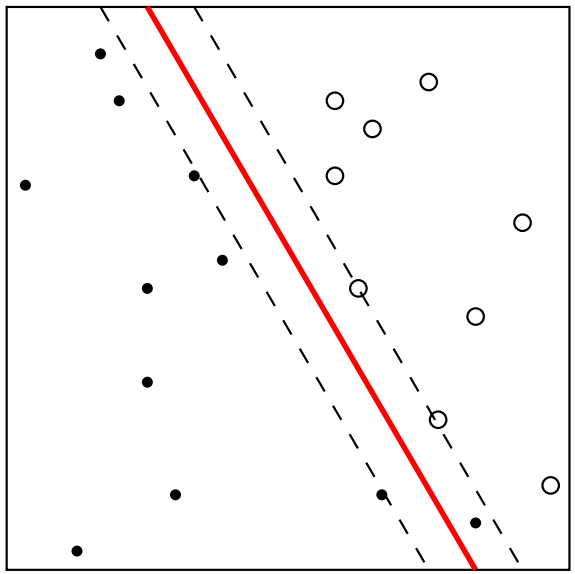
$\mathcal{H}$-divergence

The $\mathcal{H}$-divergence (Ben-David et al. (2010)) between D_S^X and D_T^X is for a hypothesis class $\mathcal{H}$ (e.g. all linear models):

$$d_{\mathcal{H}}(D_S^X, D_T^X) = 2 \sup_{h \in \mathcal{H}} \left| P_{\mathbf{x} \sim D_S^X}\left(h(\mathbf{x}) = 1\right) - P_{\mathbf{x} \sim D_T^X}\left(h(\mathbf{x}) = 1\right) \right|$$



(a) Low divergence



(b) High divergence

Approximating $\mathcal{H}$-divergence

A consistent empirical estimate is given for samples, $S \sim D_S^X$ of size n and $T \sim D_T^X$ if size n':

$$\hat{d}_{\mathcal{H}}(S,T) = 2\left(1 - \min_{h \in \mathcal{H}} \left[\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[h(\mathbf{x}_i) = 0] + \frac{1}{n'}\sum_{i=n+1}^{N} \mathbb{1}[h(\mathbf{x}_i) = 1]\right]\right)$$

Approximating $\mathcal{H}$-divergence

A consistent empirical estimate is given for samples, $S \sim D_S^X$ of size n and $T \sim D_T^X$ if size n':

$$\hat{d}_{\mathcal{H}}(S,T) = 2\left(1 - \min_{h \in \mathcal{H}} \left[\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[h(\mathbf{x}_i) = 0] + \frac{1}{n'}\sum_{i=n+1}^{N} \mathbb{1}[h(\mathbf{x}_i) = 1]\right]\right)$$

We can approximate this by forming,

$$U = \{(\mathbf{x}, 0) : \mathbf{x} \in S\} \cup \{(\mathbf{x}, 1) : \mathbf{x} \in T\},$$

training a classifier, and estimating its generalisation error.

To summarise...

- $\mathcal{H}$-divergence quantifies the separation of two distributions.

To summarise...

- $\mathcal{H}$-divergence quantifies the separation of two distributions.
- We can estimate the $\mathcal{H}$ divergence empirically.

To summarise...

- $\mathcal{H}$-divergence quantifies the separation of two distributions.
- We can estimate the $\mathcal{H}$ divergence empirically.
- We can approximate the $\mathcal{H}$ as the generalisation error of a domain discriminator.

DANN Model

We can formalise a neural network as a learning model of the form,

$$E(\theta_f, \theta_d, \theta_y) = \left[\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_y^i(G_y(G_f(\mathbf{x}_i; \theta_f); \theta_y), y_i) + \lambda R(\theta_f, \theta_d) \right]$$

where G_f acts as a feature extractor, G_y as a classifier. R is the (negative) loss function of a domain discriminator:

$$R(\theta_f, \theta_d) = -\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_d^i(\theta_d, d_i) - \frac{1}{n'} \sum_{i=n+1}^{N} \mathcal{L}_d^i(\theta_d, d_i)$$

where $\mathcal{L}_d^i(\theta_d, d^i) = \mathcal{L}(G_d(G_f(\theta_f); \theta_d), d_i), d_i \in \{0, 1\}$.

Optimisation

Formally, we have,

$$(\theta_f, \theta_y) = \arg\min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \hat{\theta}_d)$$

$$\theta_d = \arg\max_{\theta_d} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d)$$

Optimisation

Formally, we have,

$$(\theta_f, \theta_y) = \arg\min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \hat{\theta}_d)$$

$$\theta_d = \arg\max_{\theta_d} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d)$$

Crucially, select θ_f to:

- minimise G_y error (a good classifier for source data)

Optimisation

Formally, we have,

$$(\theta_f, \theta_y) = \arg\min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \hat{\theta}_d)$$

$$\theta_d = \arg\max_{\theta_d} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d)$$

Crucially, select θ_f to:

- minimise G_y error (a good classifier for source data)
- minimise R:

$$\arg\min_{\theta_f} R = \arg\min_{\theta_f} -\epsilon_{\theta_d}$$

$$\implies \text{maximise domain discrimination error}$$

$$\implies \text{minimise divergence (adaptation)}$$

DANN architecture

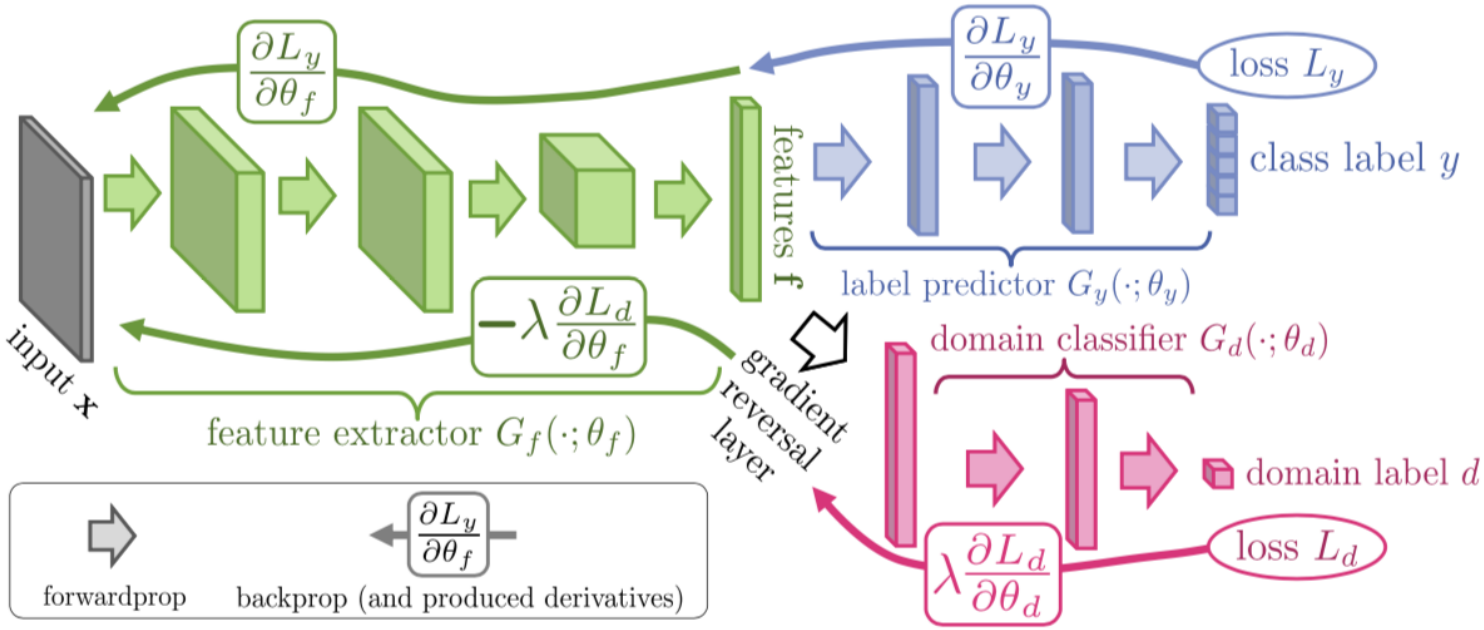


Figure: Generalised DANN architecture

- This promotes the emergence of *domain-invariant* features.
- The resulting classifier performs well on both domains.

Optimisation

Formally, we have,

$$\theta_f \leftarrow \theta_f - \mu\left(\frac{\partial \mathcal{L}_y^i}{\partial \theta_f} - \lambda \frac{\partial \mathcal{L}_d^i}{\partial \theta_f}\right)$$

$$\theta_y \leftarrow \theta_f - \mu \frac{\partial \mathcal{L}_y^i}{\partial \theta_y}$$

$$\theta_d \leftarrow \theta_f - \mu \frac{\partial \mathcal{L}_d^i}{\partial \theta_d}$$

We therefore find a saddle point between the adversarial objectives.

Gradient reversal layer

The GLR is achieved with the pseudo-function,

$$R_\lambda(\mathbf{x}) = \mathbf{x}$$
$$\frac{\partial R_\lambda}{\partial \mathbf{x}} = -\lambda \mathbf{I}$$

Gradient reversal layer

The GLR is achieved with the pseudo-function,

$$R_\lambda(\mathbf{x}) = \mathbf{x}$$
$$\frac{\partial R_\lambda}{\partial \mathbf{x}} = -\lambda \mathbf{I}$$

- This spares us having to modify the optimisation algorithm.

Gradient reversal layer

The GLR is achieved with the pseudo-function,

$$R_\lambda(\mathbf{x}) = \mathbf{x}$$
$$\frac{\partial R_\lambda}{\partial \mathbf{x}} = -\lambda \mathbf{I}$$

- This spares us having to modify the optimisation algorithm.
- In packages such as `Caffe` and `TensorFlow`, it is by contrast easy to define a function's gradient arbitrarily.

Experimentation

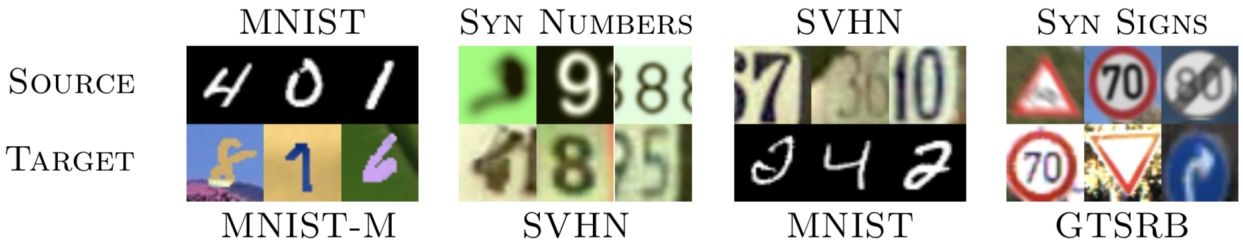
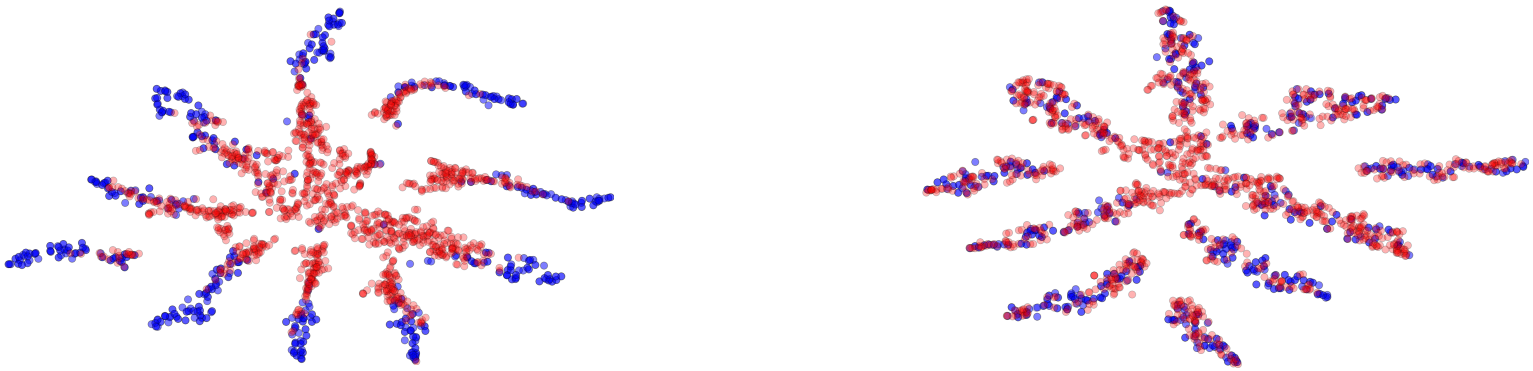


Figure: M-MNIST constructed by randomly sampling patch from natural image, then inverting the pixels in the positions of the MNIST digit



(a) Non-adapted

(b) Adapted

Figure: T-SNE plot of pre- and post-adaptation

Auto-Encoding Variational Bayes

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

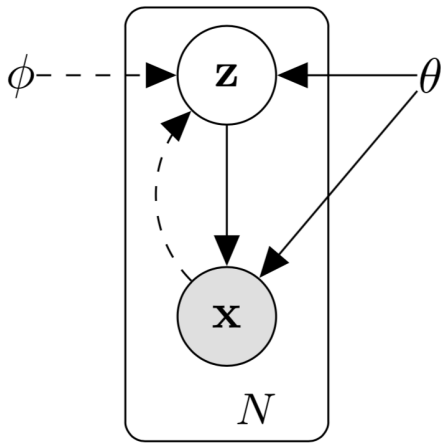
Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

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.

Auto-Encoding Variational Bayes (2014)

An auto-encoding variational Bayes (AEVB) model postulates a set of latent factors, $\mathbf{z}$ generating observations $\mathbf{x}$ (unsupervised).



AEVB proposes a *recognition model*, $q_\phi(\mathbf{z}|\mathbf{x})$ that approximates the true, generative posterior, $p_\theta(\mathbf{z}|\mathbf{x})$ (typically intractable).

Generative vs. discriminative models

- Generative models construct a posterior distribution by application of Bayes' theorem:

$$\underbrace{p(z|x)}_{\text{posterior}} = \frac{\overbrace{p(x|z)}^{\text{likelihood}}\overbrace{p(z)}^{\text{prior}}}{\underbrace{p(x)}_{\text{marginal likelihood}}}$$

- Discriminative models model the posterior distribution directly.

AEVB – variational lower bound

Consider the marginal (log-)likelihood on a single (i.i.d) observation:

$$\log p_\theta(\mathbf{x}^{(i)}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}^{(i)})\big]$$

AEVB – variational lower bound

Consider the marginal (log-)likelihood on a single (i.i.d) observation:

$$\begin{aligned}\log p_\theta(\mathbf{x}^{(i)}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}^{(i)})\right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}^{(i)}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x}^{(i)})}\right]\end{aligned}$$

AEVB – variational lower bound

Consider the marginal (log-)likelihood on a single (i.i.d) observation:

$$
\begin{aligned}
\log p_\theta(\mathbf{x}^{(i)}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}^{(i)})\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}^{(i)}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x}^{(i)})}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}^{(i)}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x}^{(i)})} \cdot \frac{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\right]
\end{aligned}
$$

AEVB – variational lower bound

Consider the marginal (log-)likelihood on a single (i.i.d) observation:

$$
\begin{aligned}
\log p_\theta(\mathbf{x}^{(i)}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}^{(i)})\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}^{(i)}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x}^{(i)})}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}^{(i)}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x}^{(i)})} \cdot \frac{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}{p_\theta(\mathbf{z}|\mathbf{x}^{(i)})}\right] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}^{(i)}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\right]
\end{aligned}
$$

AEVB – variational lower bound

Consider the marginal (log-)likelihood on a single (i.i.d) observation:

$$
\begin{aligned}
\log p_\theta(\mathbf{x}^{(i)}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}^{(i)})\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}^{(i)}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x}^{(i)})}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}^{(i)}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x}^{(i)})} \cdot \frac{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}{p_\theta(\mathbf{z}|\mathbf{x}^{(i)})}\right] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}^{(i)}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\right] \\
&= \underbrace{D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}|\mathbf{x}^{(i)})\right)}_{\geq 0} + \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)})
\end{aligned}
$$

Variational lower bound

We thus obtain the *variational lower bound* on the marginal likelihood:

$$\log p_\theta(\mathbf{x}^{(i)}) \geq \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)})$$

Therefore, if we maximise $\mathcal{L}$, we will maximise (a lower bound of) the likelihood. The bound can be further decomposed:

$$\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) = -D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})\big) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\big]$$

To summarise...

- We have supposed there is an underlying process of latent variables $\mathbf{z}$ generating the observations $\mathbf{x}$.

To summarise...

- We have supposed there is an underlying process of latent variables $\mathbf{z}$ generating the observations $\mathbf{x}$.
- We are attempting to *unsupervisedly* fit generative parameters θ by maximising the likelihood of the observed data $\mathbf{x}$.

To summarise...

- We have supposed there is an underlying process of latent variables $\mathbf{z}$ generating the observations $\mathbf{x}$.
- We are attempting to *unsupervisedly* fit generative parameters θ by maximising the likelihood of the observed data $\mathbf{x}$.
- We have replaced the intractable posterior with a recognition model $q_\phi(\mathbf{z}|\mathbf{x})$.

The reparameterisation trick

Rather than $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$, *reparameterise* as,

$$\mathbf{z} = g(\mathbf{x}, \boldsymbol{\epsilon})$$

where $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$.

The reparameterisation trick

Rather than $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$, *reparameterise* as,

$$\mathbf{z} = g(\mathbf{x}, \boldsymbol{\epsilon})$$

where $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$.

- This will be a differentiable function (in our case an *encoding network*).

The reparameterisation trick

Rather than $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$, *reparameterise* as,

$$\mathbf{z} = g(\mathbf{x}, \boldsymbol{\epsilon})$$

where $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$.

- This will be a differentiable function (in our case an *encoding network*).
- We will use this reparameterised posterior to sample $\mathbf{z}$.

Stochastic gradient variational Bayes estimator

The Stochastic gradient variational Bayes (SGVB) estimator is therefore,

$$\widetilde{\mathcal{L}}(\theta, \phi; \mathbf{x}^{(i)}) = \underbrace{-D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})\right)}_{\text{analytic solution}} + \underbrace{\frac{1}{L}\sum_{l=1}^{L} \log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})}_{\text{Monte Carlo approximation}}$$

where $\mathbf{z}^{(i,l)} = g(\mathbf{x}^{(i)}, \boldsymbol{\epsilon}^{(l)})$ and $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$. Thus, the AEVB algorithm consists of gradient descent and sampling $\mathbf{z}$.

Choosing a prior

$$p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$$

- Apt as by this we are only assuming the latent variables are emitted independently, and according to parsimonious assumptions.
- The form is convenient, as it is *conjugate* to the recognition model.

Choosing the recognition model (posterior)

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_E, \boldsymbol{\sigma}_E^2 \mathbf{I})$$

Hence, the covariance matrix is diagonal. The mean and variance are computed by an *encoder network* in the following way:

$$\begin{aligned}
\mathbf{h}_E &= \tanh(\mathbf{W}_E^{(1)}\mathbf{x} + \mathbf{b}_E^{(1)}) \\
\boldsymbol{\mu}_E &= \mathbf{W}_E^{(2)}\mathbf{h}_E + \mathbf{b}_E^{(2)} \\
\boldsymbol{\sigma}_E^2 &= \exp(\mathbf{W}_E^{(3)}\mathbf{h}_E + \mathbf{b}_E^{(3)})
\end{aligned}$$

Choosing a likelihood

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_D, \boldsymbol{\sigma}_D^2 \mathbf{I})$$

The Gaussian parameters come from a *decoder network*:

$$\mathbf{z}^{(l)} = \boldsymbol{\mu}_E + \boldsymbol{\sigma}_E \odot \boldsymbol{\epsilon}^{(l)} \qquad \text{(reparameterised sample)}$$

$$\mathbf{h}_D = \tanh(\mathbf{W}_D^{(1)} \mathbf{z}^{(l)} + \mathbf{b}_D^{(1)}) \qquad \text{(hidden state)}$$

$$\boldsymbol{\mu}_D = \mathbf{W}_D^{(2)} \mathbf{h}_D + \mathbf{b}_D^{(2)}$$

$$\boldsymbol{\sigma}_D^2 = \exp(\mathbf{W}_D^{(3)} \mathbf{h}_D + \mathbf{b}_D^{(3)})$$

where $\boldsymbol{\epsilon}^{(l)}$ is sampled from $\mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})$.

Variational auto-encoder

With the choice of distributions above, and the underlying neural network for generating the parameters, we have a *variational auto-encoder*, with *generative parameters*,

$$\boldsymbol{\theta} = \{\mathbf{W}_D^{(1)}, \mathbf{W}_D^{(2)}, \mathbf{W}_D^{(3)}, \mathbf{b}_D^{(1)}, \mathbf{b}_D^{(2)}, \mathbf{b}_D^{(3)}\}$$

and *variational parameters*,

$$\boldsymbol{\phi} = \{\mathbf{W}_E^{(1)}, \mathbf{W}_E^{(2)}, \mathbf{W}_E^{(3)}, \mathbf{b}_E^{(1)}, \mathbf{b}_E^{(2)}, \mathbf{b}_E^{(3)}\}.$$

Variational auto-encoder

With the choice of distributions above, and the underlying neural network for generating the parameters, we have a *variational auto-encoder*, with *generative parameters*,

$$\boldsymbol{\theta} = \{\mathbf{W}_D^{(1)}, \mathbf{W}_D^{(2)}, \mathbf{W}_D^{(3)}, \mathbf{b}_D^{(1)}, \mathbf{b}_D^{(2)}, \mathbf{b}_D^{(3)}\}$$

and *variational parameters*,

$$\boldsymbol{\phi} = \{\mathbf{W}_E^{(1)}, \mathbf{W}_E^{(2)}, \mathbf{W}_E^{(3)}, \mathbf{b}_E^{(1)}, \mathbf{b}_E^{(2)}, \mathbf{b}_E^{(3)}\}.$$

Thus, $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ will be inferred to generate the mean and variance parameters that maximise the probabilities (that maximise the variational bound).

Synthesising images

We can sample images in the following way:

Synthesising images

We can sample images in the following way:

1. Since $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ (diagonal covariance), choose a probability from the inverse CDF for each dimension of the latent space.

Synthesising images

We can sample images in the following way:

1. Since $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ (diagonal covariance), choose a probability from the inverse CDF for each dimension of the latent space.
2. Feed the obtained $\mathbf{z}$ through the decoder net.

Synthesising images

We can sample images in the following way:

1. Since $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ (diagonal covariance), choose a probability from the inverse CDF for each dimension of the latent space.
2. Feed the obtained $\mathbf{z}$ through the decoder net.
3. Reconstruct the obtained $\boldsymbol{\mu}_D$ as an image.

Synthesising images

We can sample images in the following way:

1. Since $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ (diagonal covariance), choose a probability from the inverse CDF for each dimension of the latent space.
2. Feed the obtained $\mathbf{z}$ through the decoder net.
3. Reconstruct the obtained $\boldsymbol{\mu}_D$ as an image.

When we do this for $\dim(\mathbf{z}) = 2$, we can plot the whole manifold

Two-dimensional manifold

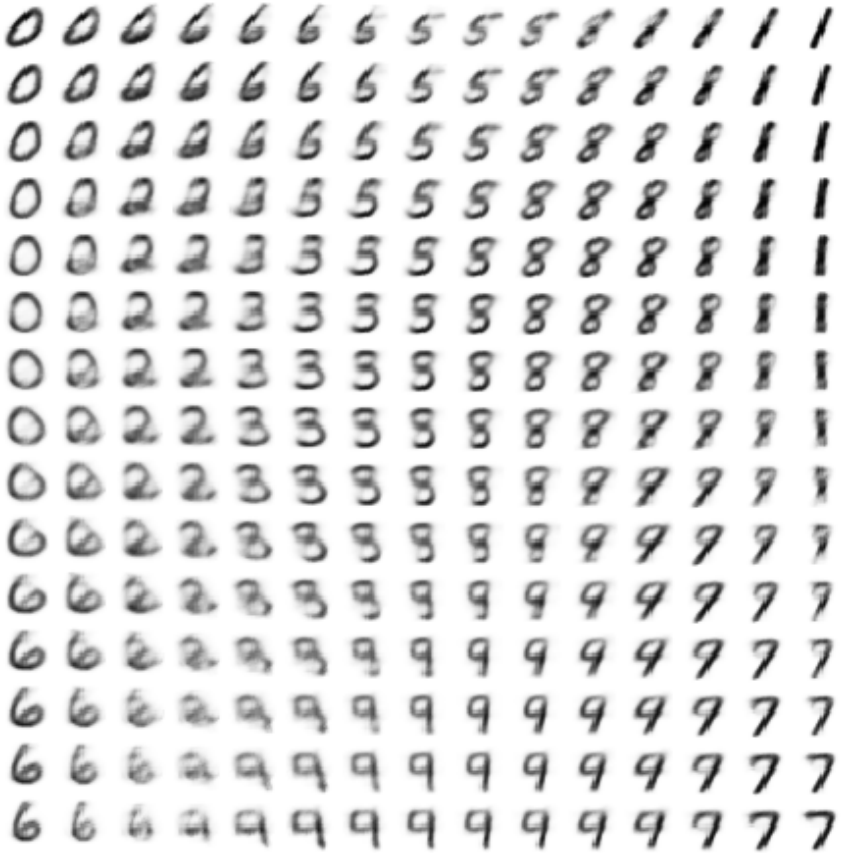


Figure: Plot of two-dimensional manifold. Produced with `fchollet/keras/examples/variational_autoencoder_deconv.py`

20-dimensional random samples (cell data)

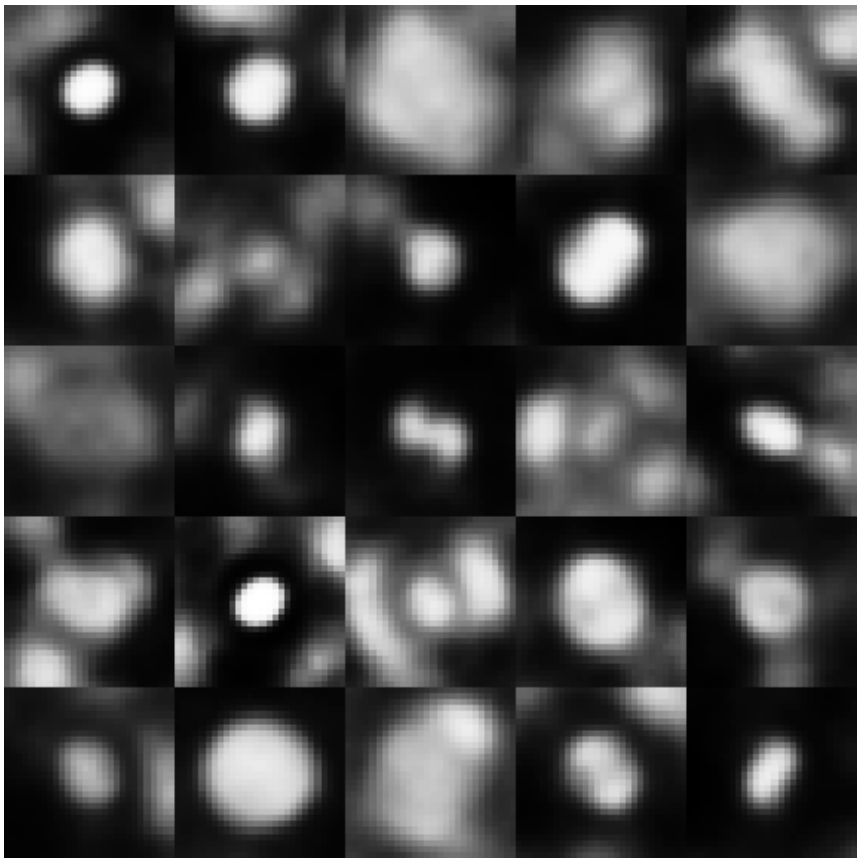


Figure: Random samples from VAE trained on (downsampled) annotated morphological cell line screen data (20 dimensional latent space).