DTU Course 02456 Deep learning
5 Un- and semi-supervised learning
2020 Updates

Ole Winther

Dept for Applied Mathematics and Computer Science
Technical University of Denmark (DTU)

November 15, 2020

Objectives of 2020 update unsupervised learning

- Flows
- Self-supervised learning
- Self-training
- Distribution Augmentation
- Flat minima

Part 1:
Flows

Motivation - modelling high dimensional distributions

- Popular probabilistic deep generative modelling:
  - Latent variable models - Deep Boltzmann Machines [Salakhutdinov et al], Variational Autoencoders [Kingma and Welling, Rezende and Mohamed].

Motivation - modelling high dimensional distributions

- Popular probabilistic deep generative modelling:
  - Latent variable models - Deep Boltzmann Machines [Salakhutdinov et al], Variational Autoencoders [Kingma and Welling, Rezende and Mohamed].
  - Autoregressive models - Recurrent Neural Networks, NADE [Uria et al], MADE [Germain et al], WaveNet [Oord et al], PixelCNN [Oord et al]

$$p(\mathbf{x}) = p(x_1) \prod_{i=2}^{d} p(x_i | x_{<i})$$

Motivation - modelling high dimensional distributions

- Popular probabilistic deep generative modelling:
  - Latent variable models - Deep Boltzmann Machines [Salakhutdinov et al], Variational Autoencoders [Kingma and Welling, Rezende and Mohamed].
  - Autoregressive models - Recurrent Neural Networks, NADE [Uria et al], MADE [Germain et al], WaveNet [Oord et al], PixelCNN [Oord et al]

$$p(\mathbf{x}) = p(x_1)\prod_{i=2}^{d} p(x_i|x_{<i})$$

  - Flow models - Normalizing flows [Rezende and Mohamed],RealNVP [Dinh et al], GLOW [Kingma and Dhariwal], MAF [Papamakarios et al].

$$\mathbf{z} \sim p(\mathbf{z}) \text{ and } \mathbf{x} = g(\mathbf{z}) \;\Rightarrow\; p(\mathbf{x}) = \left|\frac{d\mathbf{z}}{d\mathbf{x}}\right| p(\mathbf{z})\Big|_{\mathbf{z}=g^{-1}(\mathbf{x})}$$

Motivation - modelling high dimensional distributions

- Popular probabilistic deep generative modelling:
  - Latent variable models - Deep Boltzmann Machines [Salakhutdinov et al], Variational Autoencoders [Kingma and Welling, Rezende and Mohamed].
  - Autoregressive models - Recurrent Neural Networks, NADE [Uria et al], MADE [Germain et al], WaveNet [Oord et al], PixelCNN [Oord et al]

$$p(\mathbf{x}) = p(x_1) \prod_{i=2}^{d} p(x_i | x_{<i})$$

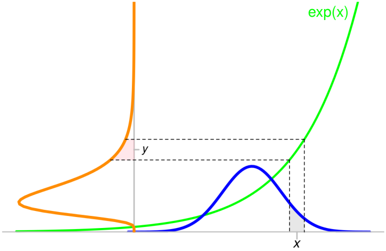
  - Flow models - Normalizing flows [Rezende and Mohamed],RealNVP [Dinh et al], GLOW [Kingma and Dhariwal], MAF [Papamakarios et al].

$$\mathbf{z} \sim p(\mathbf{z}) \text{ and } \mathbf{x} = g(\mathbf{z}) \;\Rightarrow\; p(\mathbf{x}) = \left| \frac{d\mathbf{z}}{d\mathbf{x}} \right| p(\mathbf{z}) \Big|_{\mathbf{z}=g^{-1}(\mathbf{x})}$$

- Very popular non-probabilistic generative modelling:
  - Generative adversarial networks (GAN) [Goodfellow et al]

Change of variable formula

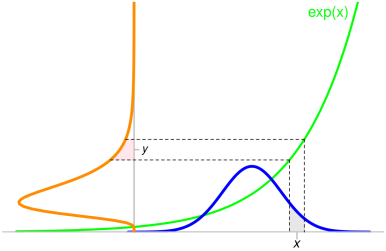
- The probability volume should be unchanged under transformation: (image source)



$$p_Y(y)dy = p_X(x)dx$$

Change of variable formula

- The probability volume should be unchanged under transformation: (image source)



$$p_Y(y)dy = p_X(x)dx$$

- Example $y = \exp x$ or $x(y) = \log y$

$$p_Y(y) = \left|\frac{d}{dy}x(y)\right| p_X(x(y)) = \left|\frac{d}{dy}\log y\right| p_X(x(y)) = \frac{1}{y}p_X(\log(y))$$

Flows

- Flows - get more flexible distribution by transformations:

$$\mathbf{z} \sim p(\mathbf{z}) \text{ and } \mathbf{x} = g(\mathbf{z}) \;\Rightarrow\; p(\mathbf{x}) = \left|\frac{d\mathbf{z}}{d\mathbf{x}}\right| p(\mathbf{z})\Big|_{\mathbf{z}=g^{-1}(\mathbf{x})}$$

- $\left|\frac{d\mathbf{z}}{d\mathbf{x}}\right|$ is the determinant of Jacobian $J_{ij} = \frac{dz_i}{dx_j}$.

Flows

- Flows - get more flexible distribution by transformations:

$$\mathbf{z} \sim p(\mathbf{z}) \text{ and } \mathbf{x} = g(\mathbf{z}) \;\Rightarrow\; p(\mathbf{x}) = \left| \frac{d\mathbf{z}}{d\mathbf{x}} \right| p(\mathbf{z}) \Big|_{\mathbf{z}=g^{-1}(\mathbf{x})}$$

- $\left| \frac{d\mathbf{z}}{d\mathbf{x}} \right|$ is the determinant of Jacobian $J_{ij} = \frac{dz_i}{dx_j}$.
- Restrictions
  - $\mathbf{x}$ and $\mathbf{z}$ should have same dimensionality
  - Transformation should be invertible: $f = g^{-1}$: $\mathbf{z} = f(\mathbf{x})$ and $\mathbf{x} = g(\mathbf{z})$
  - Most obvious for continuous data: $\mathbf{x} = f(\mathbf{z})$.

Flows

- Flows - get more flexible distribution by transformations:

$$\mathbf{z} \sim p(\mathbf{z}) \text{ and } \mathbf{x} = g(\mathbf{z}) \;\Rightarrow\; p(\mathbf{x}) = \left|\frac{d\mathbf{z}}{d\mathbf{x}}\right| p(\mathbf{z})\Big|_{\mathbf{z}=g^{-1}(\mathbf{x})}$$

- $\left|\frac{d\mathbf{z}}{d\mathbf{x}}\right|$ is the determinant of Jacobian $J_{ij} = \frac{dz_i}{dx_j}$.
- Restrictions
  - $\mathbf{x}$ and $\mathbf{z}$ should have same dimensionality
  - Transformation should be invertible: $f = g^{-1}$: $\mathbf{z} = f(\mathbf{x})$ and $\mathbf{x} = g(\mathbf{z})$
  - Most obvious for continuous data: $\mathbf{x} = f(\mathbf{z})$.
- Strength: sequence of transformations

$$f = f_1 \circ f_2 \ldots \circ f_L$$

$$f^{-1} = f_L^{-1} \circ f_{L-1}^{-1} \ldots \circ f_1^{-1}$$

$$\log\left|\frac{d\mathbf{z}}{d\mathbf{x}}\right| = \sum_{i=1}^{L} \log\left|\frac{d\mathbf{h}_i}{d\mathbf{h}_{i-1}}\right|$$

with $\mathbf{h}_0 = \mathbf{x}$ and $\mathbf{h}_i = f_1(\mathbf{h}_{i-1})$

Flows - examples

- Example - normalizing flows

$$g(\mathbf{z}) = \mathbf{z} + \mathbf{u}F(\mathbf{w}^T\mathbf{z} + b)$$

  F is a scalar function,

- Can be stacked and has inverse Jacobian:

$$\frac{dg(\mathbf{z})}{d\mathbf{z}} = \mathbf{I} + \mathbf{u}\mathbf{w}^T F'(\mathbf{w}^T\mathbf{z} + b)$$

  w easy computed determinant (matrix inversionn lemma).

Flows - examples

- Example - normalizing flows

$$g(\mathbf{z}) = \mathbf{z} + \mathbf{u}F(\mathbf{w}^T\mathbf{z} + b)$$

  F is a scalar function,

- Can be stacked and has inverse Jacobian:

$$\frac{dg(\mathbf{z})}{d\mathbf{z}} = \mathbf{I} + \mathbf{u}\mathbf{w}^T F'(\mathbf{w}^T\mathbf{z} + b)$$

  w easy computed determinant (matrix inversionn lemma).

- Example - affine transformations RealNVP and GLOW:

Description	Function	Reverse Function	Log-determinant
Actnorm. See Section 3.1.	$\forall i,j: \mathbf{y}_{i,j} = \mathbf{s} \odot \mathbf{x}_{i,j} + \mathbf{b}$	$\forall i,j: \mathbf{x}_{i,j} = (\mathbf{y}_{i,j} - \mathbf{b})/\mathbf{s}$	$h \cdot w \cdot \mathtt{sum}(\log\lvert\mathbf{s}\rvert)$
Invertible 1×1 convolution. $\mathbf{W}: [c \times c]$. See Section 3.2.	$\forall i,j: \mathbf{y}_{i,j} = \mathbf{W}\mathbf{x}_{i,j}$	$\forall i,j: \mathbf{x}_{i,j} = \mathbf{W}^{-1}\mathbf{y}_{i,j}$	$h \cdot w \cdot \log\lvert\det(\mathbf{W})\rvert$ or $h \cdot w \cdot \mathtt{sum}(\log\lvert\mathbf{s}\rvert)$ (see eq. (10))
Affine coupling layer. See Section 3.3 and (Dinh et al., 2014)	$\mathbf{x}_a, \mathbf{x}_b = \mathtt{split}(\mathbf{x})$ $(\log \mathbf{s}, \mathbf{t}) = \mathtt{NN}(\mathbf{x}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{y}_a = \mathbf{s} \odot \mathbf{x}_a + \mathbf{t}$ $\mathbf{y}_b = \mathbf{x}_b$ $\mathbf{y} = \mathtt{concat}(\mathbf{y}_a, \mathbf{y}_b)$	$\mathbf{y}_a, \mathbf{y}_b = \mathtt{split}(\mathbf{y})$ $(\log \mathbf{s}, \mathbf{t}) = \mathtt{NN}(\mathbf{y}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{x}_a = (\mathbf{y}_a - \mathbf{t})/\mathbf{s}$ $\mathbf{x}_b = \mathbf{y}_b$ $\mathbf{x} = \mathtt{concat}(\mathbf{x}_a, \mathbf{x}_b)$	$\mathtt{sum}(\log(\lvert\mathbf{s}\rvert))$

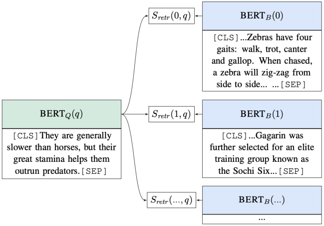
Part 2:
Self-supervised learning

Self-supervised versus unsupervised

- Unsupervised: Model $p(\mathbf{x})$ or something related to it.
- Self-supervised: Invent supervised task using only $\mathbf{x}$
- Example 1: Masked language models like BERT

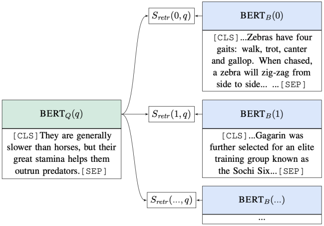
Self-supervised versus unsupervised

- Unsupervised: Model $p(\mathbf{x})$ or something related to it.
- Self-supervised: Invent supervised task using only $\mathbf{x}$
- Example 1: Masked language models like BERT
- Example 2: Inverse cloze task

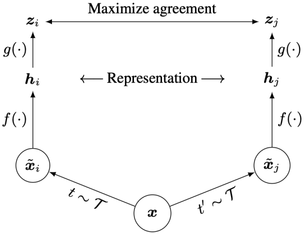


Self-supervised versus unsupervised

- Unsupervised: Model $p(\mathbf{x})$ or something related to it.
- Self-supervised: Invent supervised task using only $\mathbf{x}$
- Example 1: Masked language models like BERT
- Example 2: Inverse cloze task



- Example 3: SIMCLR



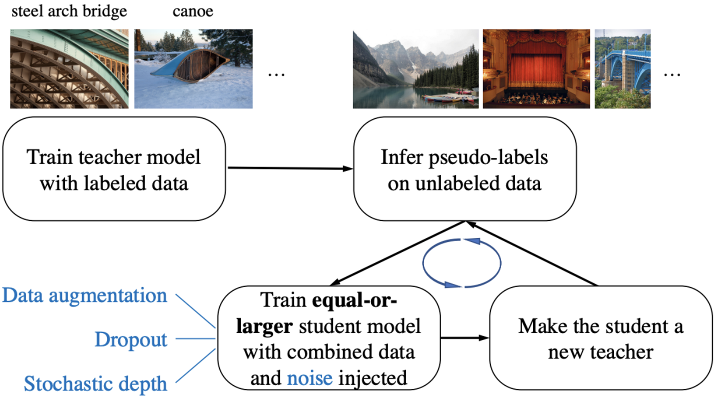
Part 3:
Self-training/Noisy student

Self-training and noisy student

- Self-training: Iteratively expand labeled set by using model to label previously unlabeled data.

Self-training and noisy student

- Self-training: Iteratively expand labeled set by using model to label previously unlabeled data.
- Noisy student

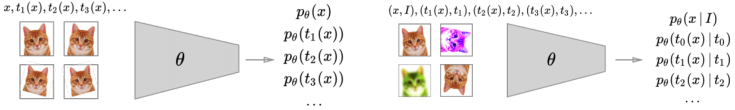


- Noisy student for audio

Part 4:
Distribution augmentation

Distribution augmentation

- DistAug

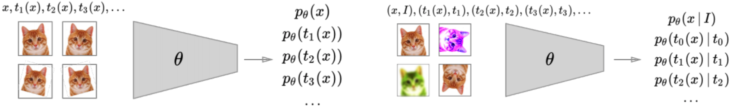


(a) Data augmentation

(b) DistAug

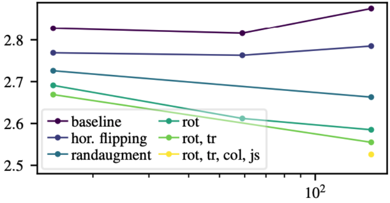
Distribution augmentation

- DistAug



(a) Data augmentation (b) DistAug

- CIFAR10 results



DistAug type	Dropout	BPD
-	60%	2.88
horizontal flipping	40%	2.79
randaugment	25%	2.66
rot	40%	2.59
rot, tr	40%	2.56
rot, tr, col	10%	2.58
rot, tr, col, js	5%	2.53

- js = jigsaw

Part 5:
Flat minima

Distribution augmentation

- Sharpness-aware minimization for efficiently improving generalization (SAM):

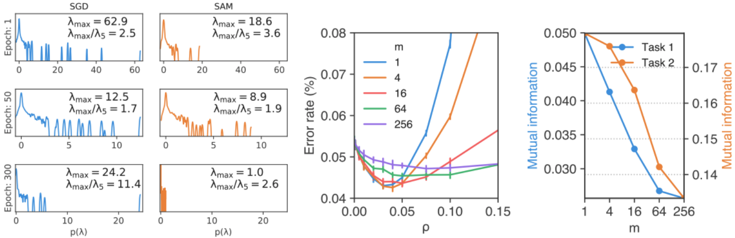
$$\mathcal{L}^{\mathrm{SAM}}(\mathbf{w}) = \max_{||\epsilon||_p < \rho} \mathcal{L}(\mathbf{w} + \epsilon)$$

Distribution augmentation

- Sharpness-aware minimization for efficiently improving generalization (SAM):

$$\mathcal{L}^{\mathrm{SAM}}(\mathbf{w}) = \max_{||\epsilon||_p < \rho} \mathcal{L}(\mathbf{w} + \epsilon)$$

- Hessian spectrum
- *m* sharpness: ϵ optimized on subset of size *m*
- Test performance as function of ball size ρ:



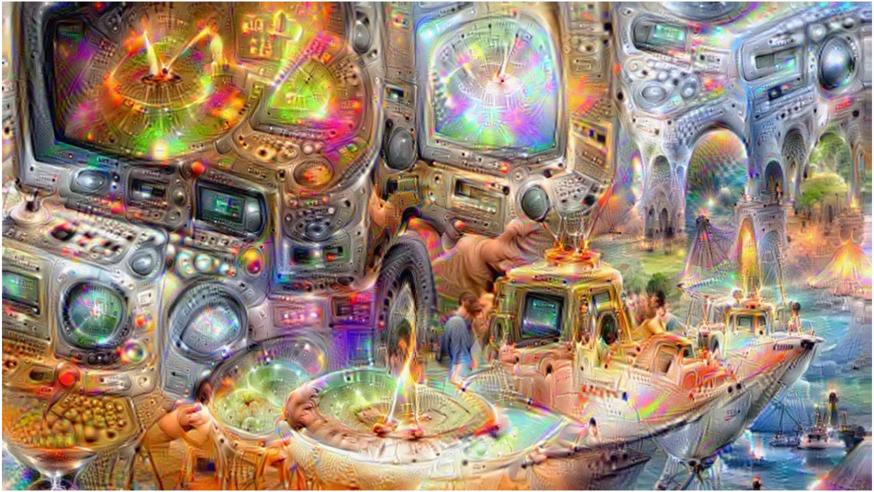
- Still open how important flat minima is for generalization.

Quiz/Exercises

- Flows: How is $\left|\frac{d\mathbf{z}}{d\mathbf{x}}\right| = \left|\frac{df(\mathbf{z})}{d\mathbf{x}}\right|$ and $\left|\frac{d\mathbf{x}}{d\mathbf{z}}\right| = \left|\frac{dg(\mathbf{z})}{d\mathbf{z}}\right|$ related?
- Flows: Calculate the Jacobian for a sequence of transformations and show that the log determinant becomes a sum of the individual log determinants.
- Flows: Normalizing flows - how would you invert $g(\mathbf{z})$?
- Flows: GLOW (the table) - what makes the transformations invertiable and the Jacobian determinant cheap to compute?
- Self-supervised learning: Come up with some possible self-supervised tasks.
- Self-supervised learning: Speculate on why self-supervised learning might sometimes work better than unsupervised.

Quiz/Exercises II

- Self-supervised learning: Explain difference to unsupervised. What category would you put masked language model in?
- Self-training/ : Give some intutition why this might work.
- Self-training/noisy student: Give an example where it might not work at all.
- Distribution augmentation: Discuss why modeling $p(t(\mathbf{x}|t)$ is better than $p(t(\mathbf{x})$.
- Flat minima: Discuss intuition behind why flat minima might be better than sharp minima.



Thanks!
Ole Winther