

# Notes on INFO8010 - Deep Learning

Yuanzhe (Roger) Li

2020-04

## Contents

<a href="#">Lecture 4: Computer Vision</a> . . . . .	1
<a href="#">Lecture 5: Training Neural Networks</a> . . . . .	1
<a href="#">Lecture 6: Recurrent Neural Networks</a> . . . . .	2
<a href="#">Lecture 7: Auto-encoders and generative models</a> . . . . .	3
<a href="#">Lecture 8: Generative Adversarial Networks</a> . . . . .	4
Resrouces . . . . .	5

This document contains notes and additional readings for self-study.

## Lecture 4: Computer Vision

- Misc.
  - On cross-entropy
    - \* [\(Wiki page\) Cross Entropy](#)
    - \* [A Gentle Introduction to Cross-Entropy for Machine Learning](#)
- Classification
  - Image augmentation
  - Use pre-trained models for fine tuning and transfer learning
  - Large networks trained for classification are heavily re-used for object detection and semantic segmentation tasks.
- Object Detection
  - YOLO for object detection
    - \* [EPFL EE-559, 8-3: Object Detection](#)
  - R-CNN
    - \* [Dive into Deep Learning - 13.8. Region-based CNNs \(R-CNNs\)](#)
  - Takeaways
    - \* One-stage detectors (YOLO, SSD, RetinaNet, etc) are *fast* for inference *not as accurate*.
    - \* Two-stage detectors (Fast R-CNN, Faster R-CNN, R-FCN, Light head R-CNN, etc) are usually *slower* but are *more accurate*.
    - \* Both depend on engineering decisions.
- Segmentation
  - Task: partitioning an image into regions of different semantic categories at *pixel level*.
  - Fully convolutional network(FCN) and transposed convolution
    - \* [CS231n, Lecture 11, 2018](#).
  - Mask R-CNN
    - \* Object detection combined with mask prediction enables instance segmentation.
    - \* [Dive into Deep Learning - 13.8.4 Mask R-CNN](#)

## Lecture 5: Training Neural Networks

- Optimizers
  - Gradient descent
    - \* GD, SGD, mini-batch SGD

- \* Rely on assumptions on 1) the magnitude of the local curvature to set the step size, and 2) *isotropy* in gradient so the step size makes sense in all directions
- [Wolfe conditions](#) ensures that both the loss function decreases sufficiently and the slope reduces sufficiently. However, line search will be too expensive for DL, and might lead to local minimum / overfitted solution.
- [Momentum](#)
  - \* Use momentum to add inertia in the choice of the step direction
  - \* [Nesterov momentum](#)
- Adaptive learning rate: without the assumption of isotropic gradient
  - \* Per-parameter methods: [AdaGrad](#), [RMSProp](#), [Adam](#)
  - \* [Scheduling](#)
- Some additional reading on optimization: [\(Sebastian Ruder\) An overview of gradient descent optimization algorithms](#)
- Initialization
  - Principles
    - \* Break symmetry
    - \* Control variance of activation across layers during forward and backward pass
  - Xavier initialization
- Normalization
  - Batch normalization
  - Layer normalization

## Lecture 6: Recurrent Neural Networks

- Types of tasks
  - Classification: sequence to classes
  - Synthesis: real values to sequence
  - Translation: sequence to sequence
- Temporal convolutions
- Recurrent neural networks
  - Structure
    - \* maintain a recurrent state updated at each time step (a function of state the previous step, input of the current step, and weights),  $\mathbf{h}_t = \phi(\mathbf{x}_t, \mathbf{h}_{t-1}; \theta)$
    - \* Predictions can be computed at any step from the recurrent state  $y_t = \psi(\mathbf{h}_t; \theta)$
    - \* Elman networks apply non-linear activation functions as  $\phi$  and  $\psi$
  - **Stacked RNN**
    - \* Since RNNs can be viewed as layers producing sequences of activations, and can be stacked
  - **Bidirectional RNNs**. RNNs can be made *bidirectional*. run the same single direction RNN twice from both end and concatenate the states.
  - Gating
    - \* Similar to the skip connections in ResNet, RNN cells can include pass-throughs so recurrent state does not go repeatedly through a squashing non-linearity.
    - \* *forget gate*: current state update be a per-component weighted average of its previous value and a full update, with the weighting depending on input and the previous state.
  - LSTM is able to learn long-term dependencies, and the core idea is to use cell state and erase/update/output gates for cell state information.
    - \* See [Understanding LSTM Networks](#) by Colah
  - GRU (gated recurrent unit) uses two (instead of three as in LSTM) gates (update/reset), and it performs similarly to LSTM but with fewer parameters (although LSTM is strictly stronger).
  - Gradient
    - \* Note that gated units prevent gradients from vanishing, but not from exploding, which can be solved using **gradient norm clipping** (scaling of the norm).
    - \* Orthogonal initialization (of the weight matrix) will guarantee that activations will neither vanish nor explode.
- Applications
  - Sentiment analysis
    - \* Document-level modeling for sentiment analysis (= text classification), with stacked, bidirectional and gated recurrent networks. [\(Duyu Tang et al, 2015\)](#)

- Language models
  - \* Language as Markov Chain  $p(\mathbf{w}_t | \mathbf{w}_{1:t-1})$
  - \* An instance of sequence synthesis where predictions are computed at all time steps
  - \* Text generation ([Max Woolf 2018](#))
- Sequence synthesis
- Neural machine translation ([Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation](#))
- Text-to-speech synthesis
- Beyond sequences
  - Neural computers
  - Programs as neural nets
  - Graph neural network
- Reference
  - [Kyunghyun Cho, “Natural Language Understanding with Distributed Representation”, 2015](#)

## Lecture 7: Auto-encoders and generative models

- Auto-encoders (AE)
  - An **auto-encoder** is a composite function made of
    - \* *encoder*  $f$  from the original space  $\mathcal{X}$  to a latent space  $\mathcal{Z}$
    - \* *decoder*  $g$  to map back to  $\mathcal{X}$
    - \* such that  $g \circ f$  is close to the identity on the data, i.e.  $\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} [\|\mathbf{x} - g \circ f(\mathbf{x})\|^2] \approx 0$
    - \* Training an auto-encoder consists of minimizing this loss function to find the best parameterization of  $f$  and  $g$ .
  - Interpolation on latent space can be made to get an intuition of the learned latent representation.
  - Denoising auto-encoders
    - \* The goal is to optimize  $h = g \circ f : \mathcal{X} \rightarrow \mathcal{X}$  such that a perturbation  $\tilde{\mathbf{x}}$  is restored to  $\mathbf{x}$ .
    - \* A weakness of denoising auto-encoder is that the posterior  $p(\mathbf{x} | \tilde{\mathbf{x}})$  may be multi-modal.
- Generative models
  - a probabilistic model that can be used to simulate the data,  $\mathbf{x} \sim p(\mathbf{x}; \theta)$ .
  - Applications
    - \* Super-resolution, Compression, text-to-speech
    - \* Proteomics, drug discovery, astronomy
    - \* Planning, exploration, model-based RL
  - The decoder  $g$  can be assessed by introducing a density model  $q$  over the latent space  $\mathcal{Z}$  for sampling and mapping back into the data space  $\mathcal{X}$ . (e.g., Gaussian  $q(\mathbf{z}) = \mathcal{N}(\hat{\mu}, \hat{\Sigma})$ )
  - Sampled and generated results are not satisfactory because the density model  $p$  on the latent space is too simple and inadequate.
- Variational inference (VI)
  - A prescribed latent variable model that defines a joint probability  $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x} | \mathbf{z})p(\mathbf{z})$
  - Bayes rule gives  $p(\mathbf{z} | \mathbf{x}) = \frac{p(\mathbf{x} | \mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}$ , which is intractable for integrating over  $\mathbf{x}$ .
  - VI turns the posterior inference into an optimization problem that minimize the KL divergence between  $p(\mathbf{z} | \mathbf{x})$  and the approximation  $q(\mathbf{z} | \mathbf{x}; \nu)$ 
    - \* See slides [pp. 44 - pp.47](#) for details of the KL divergence, *evidence lower bound objective* (ELBO), and the optimization setups.
    - \* ELBO encourages distributions to place their mass on configurations of latent variables that explain the observed data, and close to the prior.
- Variational auto-encoders
  - Variational auto-encoder is a deep latent model where
    - \*  $p(\mathbf{x} | \mathbf{z}; \theta)$  is parameterized with a **generative network**  $\text{NN}_\theta$  (decoder) that takes input  $\mathbf{z} \in \mathcal{Z}$  and outputs parameters  $\phi = \text{NN}_\theta(\mathbf{z})$  to the data distribution, i.e.

$$\mu, \sigma = \text{NN}_\theta(\mathbf{z}), \quad p(\mathbf{x} | \mathbf{z}; \theta) = \mathcal{N}(\mathbf{x}; \mu, \sigma^2 \mathbf{I})$$

- \* The approximate posterior  $q(\mathbf{z} | \mathbf{x}; \varphi)$  is parameterized with an **inference network**  $\text{NN}_\varphi$  (encoder)

that takes as input  $\mathbf{x}$  and outputs parameters  $\nu = \text{NN}_\varphi(x)$  to the approximate posterior. E.g.

$$\mu, \sigma = \text{NN}_\varphi(\mathbf{x}), \quad q(\mathbf{z}|\mathbf{x}; \varphi) = \mathcal{N}(\mathbf{z}; \mu, \sigma^2 \mathbf{I})$$

- We use variational inference to jointly optimize the generative and inference networks.
  - \* Doing so involves Monte Carlo integration (for computing gradients of the ELBO w.r.t.  $\theta$ ) and reparameterization trick + Monte Carlo (for computing gradients of ELBO w.r.t.  $\varphi$ )

## Lecture 8: Generative Adversarial Networks

- Generative adversarial networks (GANs)
  - Two-player game
    - \* Generator network  $g(\cdot; \theta) : \mathcal{Z} \rightarrow \mathcal{X}$  with prior  $p(\mathbf{z})$  on latent space, thereby inducing a generative distribution

$$\mathbf{x} \sim q(\mathbf{x}; \theta) \iff \mathbf{z} \sim p(\mathbf{z}), \mathbf{x} = g(\mathbf{z}; \theta)$$

- \* Discriminative network  $d(\cdot; \phi) : \mathcal{X} \rightarrow [0, 1]$  as a classifier to distinguish between true samples  $\mathbf{x} \sim p(\mathbf{x})$  and generated samples  $\mathbf{x} \sim q(\mathbf{x}; \theta)$
- Objective of GANs
  - \* Use cross-entropy loss for  $d$  and we have the *value function* (log-likelihood)

$$V(\phi, \theta) = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} [\log d(\mathbf{x}; \theta)] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [1 - d(g(\mathbf{z}; \theta); \phi)]$$

- \* The ultimate goal is  $\theta^* = \arg \min_{\theta} \max_{\phi} V(\phi, \theta)$
  - \* Mathematically  $\theta^*$  is minimum  $\iff p(\mathbf{x}) = q(\mathbf{x}; \theta)$ , that is, the corresponding generative model can perfectly reproduce the true data distribution.
- Learning process
  - \* Alternating SGD
 
$$\theta \leftarrow \theta - \gamma \nabla_{\theta} V(\phi, \theta); \quad \phi \leftarrow \phi - \gamma \nabla_{\phi} V(\theta, \phi)$$
  - \* For each step of  $\theta$  we can take  $k$  steps on  $\phi$  to make the classifier near optimal
  - \* Computing  $\nabla_{\theta}$  requires backprop through  $d$  before computing the partial derivatives w.r.t.  $g$ 's internals.
- Open problems
  - \* Training standard GAN often results in pathological behaviors due to
    - Oscillations without convergence
    - Vanishing gradients
    - Mode collapse:  $g$  models well on a small sub-population concentrating on a few modes of the data distribution.
    - Performance difficult to assess in practice.

- Wasserstein GANs
  - Original GAN as defined above suffers from vanishing gradients, especially when initially  $\mathbf{x} \sim q(\mathbf{x}; \theta)$  can be so bad that the response of  $d$  saturates, meaning  $d$  is nearly perfect there fore  $V(\phi, \theta) = \dots(\text{defined above})$  has near-zero gradients which halts the optimization.
    - \* One of the reason for the setback is that Jensen-Shannon divergence poorly accounts for the metric structure of the space.
  - Wasserstein GAN uses Wasserstein-1 distance

$$\theta^* = \arg \min_{\theta} W_1(p(\mathbf{x}) || q(\mathbf{x}; \theta)) = \arg \min_{\theta} \max_{\phi: \|d(\cdot; \phi)\|_L \leq 1} \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} [d(\cdot; \phi)] - \mathbb{E}_{\mathbf{x} \sim q(\mathbf{x}; \theta)} [d(\mathbf{x}; \theta)]$$

- \* In this case,  $d : \mathcal{X} \rightarrow \mathbb{R}$  is a critic function that satisfies 1-Lipschitzness.
  - \* See [Arjovsky et al \(2017\)](#) and [Gulrajani et al \(2017\)](#) for details.
- As a result, Wasserstein GANs benefit from
  - \* A meaningful loss metric
  - \* Improved stability (no mode collapse is observed)

- Convergence of GANs
  - TODO: this section (from pp. 39 - 54) is skipped for now, with some brief notes as follows:

- \* GANs suffer from [saddle point](#), see [Ferenc Huszár, GANs are Broken in More than One Way, 2017](#) for details
- \* The loss function of Vanilla GANs and Wasserstein GANs can be unified to analyse the convergence.
- \* Reference: [Mescheder et al, 2018. Which Training Methods for GANs do actually Converge?](#)
- State of the art
- Applications

## Resources

- [EPFL EE-559 – Deep Learning](#) - EE-559 “Deep Learning”, taught by François Fleuret in the School of Engineering of the École Polytechnique Fédérale de Lausanne, Switzerland.
- [Dive into Deep Learning](#): An interactive deep learning book with code, math, and discussions, based on the NumPy interface.
- [Notes of deep learning specialization](#), good for reviewing the fundamentals of DL.